The Journal of Machine Learning Research
Volume 12
Print-Archive Edition

Pages 1149–2334

Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 12
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 2011.

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 12, 2011

Journal of Machine Learning Research 12 (2011) 1149-1184 Submitted 10/09; Revised 2/10; Published 3/11

Laplacian Support Vector Machines Trained in the Primal

Stefano Melacci MELA@DII.UNISI.IT
Department of Information Engineering
University of Siena
Siena, 53100, ITALY

Mikhail Belkin MBELKIN@CSE.OHIO-STATE.EDU
Department of Computer Science and Engineering
Ohio State University
Columbus, OH 43210, USA

Editor: Sathiya Keerthi

Abstract

In the last few years, due to the growing ubiquity of unlabeled data, much effort has been spent by the machine learning community to develop better understanding and improve the quality of classifiers exploiting unlabeled data. Following the manifold regularization approach, Laplacian Support Vector Machines (LapSVMs) have shown the state of the art performance in semi-supervised classification. In this paper we present two strategies to solve the *primal* LapSVM problem, in order to overcome some issues of the original *dual* formulation. In particular, training a LapSVM in the primal can be efficiently performed with preconditioned conjugate gradient. We speed up training by using an early stopping strategy based on the prediction on unlabeled data or, if available, on labeled validation examples. This allows the algorithm to quickly compute approximate solutions with roughly the same classification accuracy as the optimal ones, considerably reducing the training time. The computational complexity of the training algorithm is reduced from $O(n^3)$ to $O(kn^2)$, where n is the combined number of labeled and unlabeled examples and k is empirically evaluated to be significantly smaller than n. Due to its simplicity, training LapSVM in the primal can be the starting point for additional enhancements of the original LapSVM formulation, such as those for dealing with large data sets. We present an extensive experimental evaluation on real world data showing the benefits of the proposed approach.

Keywords: Laplacian support vector machines, manifold regularization, semi-supervised learning, classification, optimization

1. Introduction

In semi-supervised learning one estimates a target classification/regression function from a few labeled examples together with a large collection of unlabeled data. In the last few years there has been a growing interest in the semi-supervised learning in the scientific community. Many algorithms for exploiting unlabeled data in order to enhance the quality of classifiers have been recently proposed, see, for example, Chapelle et al. (2006) and Zhu and Goldberg (2009). The general principle underlying semi-supervised learning is that the marginal distribution, which can be estimated from unlabeled data alone, may suggest a suitable way to adjust the target function. The two commons assumption on such distribution that, explicitly or implicitly, are made by many of semi-supervised learning algorithms are the *cluster assumption* (Chapelle et al., 2003) and the *manifold assumption* (Belkin et al., 2006). The cluster assumption states that two points are likely to have the same class label if they can be connected by a curve through a high density region. Consequently, the separation boundary between classes should lie in the lower density region of the space. For example, this intuition underlies the Transductive Support Vector Machines (Vapnik, 2000) and its different implementations, such as TSVM (Joachims, 1999) or S^{3}VM (Demiriz and Bennett, 2000; Chapelle et al., 2008). The manifold assumption states that the marginal probability distribution underlying the data is supported on or near a low-dimensional manifold, and that the target function should change smoothly along the tangent direction. Many graph based methods have been proposed in this direction, but the most of them only perform transductive inference (Joachims, 2003; Belkin and Niyogi, 2003; Zhu et al., 2003), that is classify the unlabeled data given in training. Laplacian Support Vector Machines (LapSVMs) (Belkin et al., 2006) provide a natural out-of-sample extension, so that they can classify data that becomes available after the training process, without having to retrain the classifier or resort to various heuristics.

In this paper, we focus on the LapSVM algorithm, that has been shown to achieve state of the art performance in semi-supervised classification. The original approach used to train LapSVM in Belkin et al. (2006) is based on the dual formulation of the problem, in a traditional SVM-like fashion. This dual problem is defined on a number of dual variables equal to l, the number of labeled points. If the total number of labeled and unlabeled points is n, the relationship between the l variables and the final n coefficients is given by a linear system of n equations and variables. The overall cost of the process is $O(n^3)$.

Motivated by the recent interest in solving the SVM problem in the primal (Keerthi and DeCoste, 2005; Joachims, 2006; Chapelle, 2007; Shalev-Shwartz et al., 2007), we present a solution to the primal LapSVM problem that can significantly reduce training times and overcome some issues of the original training algorithm. Specifically, the contributions of this paper are the following:

1. We propose two methods for solving the LapSVM problem in the primal form (not limited to the linear case), following the ideas presented by Chapelle (2007) for SVMs and pointing out some important differences resulting from an additional regularization term. Our Matlab library can be downloaded from:
   `http://sourceforge.net/projects/lapsvmp/`

   First, we show how to solve the problem using the Newton's method and compare the result with the supervised (SVM) case. Interestingly, it turns out that the advantages of the Newton's method for the SVM problem are lost in LapSVM due to the intrinsic norm regularizer, and the complexity of this solution is still $O(n^3)$, same as in the original dual formulation.

   The second method is preconditioned conjugate gradient, which seems better suited to the LapSVM optimization problem. We see that preconditioning by the kernel matrix comes at no additional cost, and each iteration has complexity $O(n^2)$. Empirically, we establish that only a small number of iterations is necessary for convergence. Complexity can be further reduced if the kernel matrix is sparse, increasing the scalability of the algorithm.

2. We note that the quality of an approximate solution of the traditional dual form and the resulting approximation of the target optimal function are hard to relate due to the change of variables when passing to the dual problem. Training LapSVMs in the primal overcomes this issue, and it allows us to directly compute approximate solutions by controlling the number of conjugate gradient iterations.

3. An approximation of the target function with roughly the same classification accuracy as the optimal one can be achieved with a small number of iterations due to the influence of the intrinsic norm regularizer of LapSVMs on the training process. We investigate those effects, showing that they make common stopping conditions for iterative gradient based algorithms hard to tune, often leading to either a premature stopping of the iteration or to a large amount of unnecessary iterations, which do not improve classification accuracy. Instead we suggest a criterion dependent on the *output* of the classifier on the training data for terminating the iteration of our algorithm. This criterion exploits the stability of the prediction on the unlabeled data, or the classification accuracy on validation data (if available). A number of experiments on several data sets support these types of criteria, showing that accuracy similar to that of the optimal solution can be obtained in significantly reduced training time.

4. The primal solution of the LapSVM problem is based on an L_2 hinge loss, that establishes a direct connection to the Laplacian Regularized Least Square Classifier (LapRLSC) (Belkin et al., 2006). We discuss the similarities between primal LapSVM and LapRLSC and we show that the proposed fast solution can be straightforwardly applied to LapRLSC.

The rest of the paper is organized as follows. In Section 2 we recall the basic approach of manifold regularization. Section 2.1 describes the LapSVM algorithm in its original formulation while in Section 3 we discuss in detail the proposed solutions in the primal form. The quality of an approximate solution and the data based early stopping criterion are discussed in Section 4. In Section 5 a parallel with the primal solution of LapSVM and the solution for LapRLSC (Regularized Least Squares) is drawn, describing some possible future work. An extensive experimental analysis is presented in Section 6, and, finally, Section 7 concludes the paper.

2. Manifold Regularization

First, we introduce some notation that will be used in this section and in the rest of the paper. We take $n = l + u$ to be the number of m dimensional training examples $x_i \in X \subset \mathbb{R}^m$, collected in $\mathcal{S} = \{x_i, i = 1, \dots, n\}$. Examples are ordered so that the first l ones are labeled, with label $y_i \in \{-1, 1\}$, and the remaining u points are unlabeled. We put $\mathcal{S} = \mathcal{L} \cup \mathcal{U}$, where $\mathcal{L} = \{(x_i, y_i), i = 1, \dots, l\}$ is the labeled data set and $\mathcal{U} = \{x_i, i = l + 1, \dots, n\}$ is the unlabeled data set. Labeled examples are generated accordingly to the distribution P on $X \times \mathbb{R}$, whereas unlabeled examples are drawn according to the marginal distribution P_X of P. Labels are obtained from the conditional probability distribution $P(y|x)$. L is the graph Laplacian associated to $\mathcal{S}$, given by $L = D - W$, where W is the adjacency matrix of the data graph (the entry in position i, j is indicated with w_{ij}) and D is the diagonal matrix with the degree of each node (i.e., the element d_{ii} from D is $d_{ii} = \sum_{j=1}^{n} w_{ij}$). Laplacian can be expressed in the normalized form, $L = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, and iterated to a degree p greater that one. By $K \in \mathbb{R}^{n,n}$ we denote the Gram matrix associated to the n points of $\mathcal{S}$ and the i, j-th entry of such matrix is the evaluation of the kernel function $k(x_i, x_j)$, $k : X \times X \to \mathbb{R}$. The unknown target function that the learning algorithm must estimate is indicated with $f : X \to \mathbb{R}$, where f is the vector of the n values of f on training data, $f = [f(x_i), x_i \in \mathcal{S}]^T$. In a classification problem, the decision function that discriminates between classes is indicated with $y(x) = g(f(x))$, where we overloaded the use of y to denote such function.

Manifold regularization approach (Belkin et al., 2006) exploits the geometry of the marginal distribution P_X. The support of the probability distribution of data is assumed to have the geometric structure of a Riemannian manifold $\mathcal{M}$. The labels of two points that are close in the intrinsic geometry of P_X (i.e., with respect to geodesic distances on $\mathcal{M}$) should be the same or similar in sense that the conditional probability distribution $P(y|x)$ should change little between two such points. This constraint is enforced in the learning process by an intrinsic regularizer $\|f\|_I^2$ that is empirically estimated from the point cloud of labeled and unlabeled data using the graph Laplacian associated to them, since $\mathcal{M}$ is truly unknown. In particular, choosing exponential weights for the adjacency matrix leads to convergence of the graph Laplacian to the Laplace-Beltrami operator on the manifold (Belkin and Niyogi, 2008). As a result, we have

$$\|f\|_I^2 = \sum_{i=1}^{n}\sum_{j=i}^{n} w_{ij}(f(x_i)-f(x_j))^2 = f^T L f. \tag{1}$$

Consider that, in general, several natural choices of $\| \|_I$ exist (Belkin et al., 2006).

In the established regularization framework for function learning, given a kernel function $k(\cdot,\cdot)$, its associated Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ of functions $X \to \mathbb{R}$ with corresponding norm $\| \|_A$, we estimate the target function by minimizing

$$f^* = \arg\min_{f\in\mathcal{H}_k} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_A^2 + \gamma_I \|f\|_I^2 \tag{2}$$

where V is some loss function and γ_A is the weight of the norm of the function in the RKHS (or *ambient* norm), that enforces a smoothness condition on the possible solutions, and γ_I is the weight of the norm of the function in the low dimensional manifold (or *intrinsic* norm), that enforces smoothness along the sampled $\mathcal{M}$. For simplicity, we removed every normalization factor of the weights of each term in the summation. The ambient regularizer makes the problem well-posed, and its presence can be really helpful from a practical point of view when the manifold assumption holds at a lesser degree.

It has been shown in Belkin et al. (2006) that f^* admits an expansion in terms of the n points of $\mathcal{S}$,

$$f^*(x) = \sum_{i=1}^{n} \alpha_i^* k(x_i, x). \tag{3}$$

The decision function that discriminates between class $+1$ and -1 is $y(x) = sign(f^*(x))$. Figure 1 shows the effect of the intrinsic regularizer on the "clock" toy data set. The supervised approach defines the classification hyperplane just by considering the two labeled examples, and it does not benefit from unlabeled data (Figure 1(b)). With manifold regularization, the classification appears more natural with respect to the geometry of the marginal distribution (Figure 1(c)).

The intrinsic norm of Equation 1 actually performs a transduction along the manifold that enforces the values of f in nearby points with respect to geodesic distances on $\mathcal{M}$ to be the "same". From a merely practical point of view, the intrinsic regularizer can be excessively strict in some situations. Since the decision function $y(x)$ relies only on the sign of the target function $f(x)$, if f has the same sign on nearby points along $\mathcal{M}$ then the graph transduction is actually complete. Requiring that f assumes exactly the same value on a pair of nearby points could be considered as over constraining the problem. We will use this consideration in Section 4 to early stop the training algorithm.

This intuition is closely related to some recently proposed alternative formulations of the problem of Equation 2. In Tsang and Kwok (2006) the intrinsic regularizer is based on the ε-insensitive

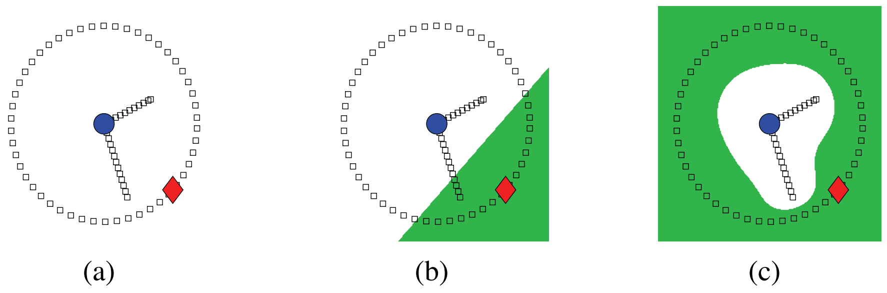


Figure 1: (a) The two class "clock" data set. One class is the circular border of the clock, the other one is the hour/minute hands. A large set of unlabeled examples (black squares) and only one labeled example per class (red diamond, blue circle) are selected. - (b) The result of a maximum margin supervised classification - (c) The result of a semi-supervised classification with intrinsic norm from manifold regularization.

loss and the problem is mapped to a Minimal Enclosing Ball (MEB) formulation. Differently, the Manifold Co-Regularization (MCR) framework (Sindhwani and Rosenberg, 2008) has been introduced to overcome the degeneration of the intrinsic regularizer to the ambient one in some restricted function spaces where it is not able to model some underlying geometries of the given data. MCR is based on multi-view learning, and it has been shown that it corresponds to adding some extra slack variables in the objective function of Equation 2 to better fit the intrinsic regularizer. Similarly, Abernethy et al. (2008) use a slack based formulation to improve the flexibility of the graph regularizer of their spam detector.

2.1 Laplacian Support Vector Machines

LapSVMs follow the principles behind manifold regularization (Equation 2), where the loss function $V(x,y,f)$ is the linear hinge loss (Vapnik, 2000), or L_1 loss. The interesting property of such function is that well classified labeled examples are not penalized by $V(x,y,f)$, independently by the value of f.

In order to train a LapSVM classifier, the following problem must be solved

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{l} \max(1 - y_i f(x_i), 0) + \gamma_A \|f\|_A^2 + \gamma_I \|f\|_I^2. \tag{4}$$

The function $f(x)$ admits the expansion of Equation 3, where an unregularized bias term b can be added as in many SVM formulations.

The solution of LapSVM problem proposed by Belkin et al. (2006) is based on the dual form. By introducing the slack variables ξ_i, the unconstrained primal problem can be written as a constrained one,

$$\min_{\alpha \in \mathbb{R}^n, \xi \in \mathbb{R}^l} \sum_{i=1}^{l} \xi_i + \gamma_A \alpha^T K \alpha + \gamma_I \alpha^T K L K \alpha$$

$$\text{subject to:} \quad y_i\left(\sum_{j=1}^{n} \alpha_i k(x_i, x_j) + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, l$$

$$\xi_i \geq 0, \quad i = 1, \ldots, l.$$

After the introduction of two sets of n multipliers β, ς, the Lagrangian L_g associated to the problem is

$$\begin{aligned} L_g(\alpha,\xi,b,\beta,\varsigma) &= \sum_{i=1}^{l}\xi_i + \frac{1}{2}\alpha^T(2\gamma_A K + 2\gamma_I KLK)\alpha - \\ &\quad - \sum_{i=1}^{l}\beta_i(y_i(\sum_{j=1}^{n}\alpha_i k(x_i,x_j)+b) - 1 + \xi_i) - \sum_{i=1}^{l}\varsigma_i\xi_i. \end{aligned}$$

In order to recover the dual representation we need to set

$$\begin{aligned} \frac{\partial L_g}{\partial b} = 0 &\implies \sum_{i=1}^{l}\beta_i y_i = 0, \\ \frac{\partial L_g}{\partial \xi_i} = 0 &\implies 1 - \beta_i - \varsigma_i = 0 \implies 0 \le \beta_i \le 1, \end{aligned}$$

where the bounds on β_i consider that $\beta_i, \varsigma_i \ge 0$, since they are Lagrange multipliers. Using the above identities, we can rewrite the Lagrangian as a function of α and β only. Assuming (as stated in Section 2) that the points in $\mathcal{S}$ are ordered such that the first l are labeled and the remaining u are unlabeled, we define with $J_L \in I\!R^{l,n}$ the matrix $[I\ 0]$ where $I \in I\!R^{l,l}$ is the identity matrix and $0 \in I\!R^{l,u}$ is a rectangular matrix with all zeros. Moreover, $Y \in I\!R^{l,l}$ is a diagonal matrix composed by the labels $y_i, i = 1,\ldots,l$. The Lagrangian becomes

$$\begin{aligned} L_g(\alpha,\beta) &= \frac{1}{2}\alpha^T(2\gamma_A K + 2\gamma_I KLK)\alpha - \sum_{i=1}^{l}\beta_i(y_i(\sum_{j=1}^{n}\alpha_i k(x_i,x_j)+b) - 1) = \\ &= \frac{1}{2}\alpha^T(2\gamma_A K + 2\gamma_I KLK)\alpha - \alpha^T K J_L^T Y\beta + \sum_{i=1}^{l}\beta_i. \end{aligned}$$

Setting to zero the derivative with respect to α establishes a direct relationships between the β coefficients and the α ones,

$$\begin{aligned} \frac{\partial L_g}{\partial \alpha} = 0 &\implies (2\gamma_A K + 2\gamma_I KLK)\alpha - K J_L^T Y\beta = 0 \\ &\implies \alpha = (2\gamma_A I + 2\gamma_I KL)^{-1} J_L^T Y\beta. \qquad (5) \end{aligned}$$

After substituting back in the Lagrangian expression, we get the dual problem whose solution leads to the optimal β^*, that is

$$\begin{aligned} &\max_{\beta \in I\!R^l} \textstyle\sum_{i=1}^{l}\beta_i - \frac{1}{2}\beta^T Q\beta \\ &\text{subject to:}\quad \textstyle\sum_{i=1}^{l}\beta_i y_i = 0 \\ &\qquad 0 \le \beta_i \le 1, \quad i = 1,\ldots,l \end{aligned}$$

where

$$Q = Y J_L K(2\gamma_A I + 2\gamma_I KL)^{-1} J_L^T Y. \qquad (6)$$

Training the LapSVM classifier requires to optimize this l variable problem, for example using a standard quadratic SVM solver, and then to solve the linear system of n equations and n variables of Equation 5 in order to get the coefficients α^* that define the target function f^*.

The overall complexity of this solution is $O(n^3)$, due to the matrix inversion of Equation 5 (and 6). Even if the l coefficients β^* are sparse, since they come from a SVM-like dual problem, the expansion of f^* will generally involves all n coefficients α^*.

3. Training in the Primal

In this section we analyze the optimization of the primal form of the non linear LapSVM problem, following the growing interest in training SVMs in the primal of the last few years (Keerthi and DeCoste, 2005; Joachims, 2006; Chapelle, 2007; Shalev-Shwartz et al., 2007). Primal optimization of a SVM has strong similarities with the dual strategy (Chapelle, 2007), and its implementation does not require any particularly complex optimization libraries. The focus of researchers has been mainly on the solution of the linear SVM primal problem, showing how it can be solved fast and efficiently. In the Modified Finite Newton method of Keerthi and DeCoste (2005) the SVM problem is optimized in the primal by a numerically robust conjugate gradient technique that implements the Newton iterations. In the works of Joachims (2006) and Shalev-Shwartz et al. (2007) a cutting plane algorithm and a stochastic gradient descent are exploited, respectively. Most of the existing results can be directly extended to the non linear case by reparametrizing the linear output function $f(x) = \langle w,x\rangle + b$ with $w = \sum_{i=1}^{l} \alpha_i x_i$ and introducing the Gram matrix K. However this may result in a loss of efficiency. Other authors (Chapelle, 2007; Keerthi et al., 2006) investigated efficient solutions for the non linear SVM case.

Primal and dual optimization are two ways different of solving the same problem, neither of which can in general be considered a “better” approach. Therefore why should a solution of the primal problem be useful in the case of LapSVM? There are three primary reasons why such a solution may be preferable. First, it allows us to efficiently solve the original problem without the need of the computations related to the variable switching. Second, it allows us to very quickly compute good *approximate* solutions, while the exact relation between approximate solutions of the dual and original problems may be involved. Third, since it allows us to directly “manipulate” the α coefficients of f without passing through the β ones, greedy techniques for incremental building of the LapSVM classifier are easier to manage (Sindhwani, 2007). We believe that studying the primal LapSVM problem is the basis for future investigations and improvements of this classifier.

We rewrite the primal LapSVM problem of Equation 4 by considering the representation of f of Equation 3, the intrinsic regularizer of Equation 1, and by indicating with k_i the i-th column of the matrix K and with 1 the vector of n elements equal to 1:

$$\min_{\alpha\in\mathbb{R}^n, b\in\mathbb{R}} \sum_{i=1}^{l} V(x_i, y_i, k_i^T\alpha + b) + \gamma_A \alpha^T K \alpha + \gamma_I (\alpha^T K + 1^T b) L (K\alpha + 1b).$$

For completeness, we included the bias b in the expansion of f. Here and in all the following derivations, L can be interchangeably used in its normalized or unnormalized version.

We use the squared hinge loss, or L_2 loss, for the labeled examples. The differentiability of such function and its properties have been investigated in Mangasarian (2002) and applied to kernel classifiers. Afterwards, it was also exploited by Keerthi and DeCoste (2005) and Chapelle (2007). L_2 loss makes the LapSVM problem continuous and differentiable in f and so in α. The optimization problem after adding the scaling constant $\frac{1}{2}$ becomes

$$\min_{\alpha \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2}\left(\sum_{i=1}^{l} \max(1 - y_i(k_i^T \alpha + b), 0)^2 + \gamma_A \alpha^T K \alpha + \gamma_I (\alpha^T K + 1^T b) L (K\alpha + 1b)\right). \tag{7}$$

We solved such convex problem by Newton's method and by preconditioned conjugate gradient, comparing their complexities and the complexity of the original LapSVM solution, and showing a parallel with the SVM case. The two solution strategies are analyzed in the following Subsections, while a large set of experimental results are collected in Section 6.

3.1 Newton's Method

The problem of Equation 7 is piecewise quadratic and the Newton's method appears a natural choice for an efficient minimization, since it builds a quadratic approximation of the function. After indicating with z the vector $z = [b, \alpha^T]^T$, each Newton's step consists of the following update

$$z^t = z^{t-1} - sH^{-1}\nabla \tag{8}$$

where t is the iteration number, s is the step size, and ∇ and H are the gradient and the Hessian of Equation 7 with respect to z. We will use the symbols ∇_α and ∇_b to indicate the gradient with respect to α and to b.

Before continuing, we introduce the further concept of *error vectors* (Chapelle, 2007). The set of error vectors $\mathcal{E}$ is the subset of $\mathcal{L}$ with the points that generate a L_2 hinge loss value greater than zero. The classifier does not penalize all the remaining labeled points, since the f function on that points produces outputs with the same sign of the corresponding label and with absolute value greater then or equal to it. In the classic SVM framework, error vectors correspond to support vectors at the optimal solution. In the case of LapSVM, all points are support vectors in the sense that they all generally contribute to the expansion of f.

We have

$$\nabla = \begin{bmatrix} \nabla_b \\ \nabla_\alpha \end{bmatrix} = \begin{pmatrix} 1^T I_{\mathcal{E}}(K\alpha + 1b) - 1 I_{\mathcal{E}} y + \gamma_I 1^T L (K\alpha + 1b) \\ K I_{\mathcal{E}}(K\alpha + 1b) - K I_{\mathcal{E}} y + \gamma_A K\alpha + \gamma_I K L (K\alpha + 1b) \end{pmatrix} \tag{9}$$

where $y \in \{-1, 0, 1\}^n$ is the vector that collects the l labels y_i of the labeled training points and a set of u zeros. The matrix $I_{\mathcal{E}} \in \mathbb{R}^{n,n}$ is a diagonal matrix where the only elements different from 0 (and equal to 1) along the main diagonal are in positions corresponding to points of $\mathcal{S}$ that belong to $\mathcal{E}$ at the current iteration. Note that if the graph Laplacian is not normalized, we have $1^T L = 0^T$ and, equivalently, $L1 = 0$.

The Hessian H is

$$H = \begin{pmatrix} \nabla_b^2 & \nabla_b(\nabla_\alpha) \\ \nabla_\alpha(\nabla_b) & \nabla_\alpha^2 \end{pmatrix} = \begin{pmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T L 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T L K \\ K I_{\mathcal{E}} 1 + \gamma_I K L 1 & K I_{\mathcal{E}} K + \gamma_A K + \gamma_I K L K \end{pmatrix} =$$

$$= \begin{pmatrix} -\gamma_A & 1^T \\ 0 & K \end{pmatrix} \begin{pmatrix} 0 & 1^T \\ I_{\mathcal{E}} 1 + \gamma_I L 1 & I_{\mathcal{E}} K + \gamma_A I + \gamma_I L K \end{pmatrix}.$$

Note that the criterion function of Equation 7 is not twice differentiable everywhere, so that H is the generalized Hessian where the subdifferential in the breakpoint of the hinge function is set to

0. This leaves intact the least square nature of the problem, as in the Modified Newton's method proposed by Keerthi and DeCoste (2005) for linear SVMs. In other words, the contribute to the Hessian of the L_2 hinge loss is the same as the one of a squared loss $(y_i - f(x_i))^2$ applied to error vectors only.

Combining the last two expressions we can write $\nabla = Hz - \begin{pmatrix} 1^T \\ K \end{pmatrix} I_{\mathcal{E}} y$, and we can plug it into the Newton's update of Equation 8,

$$\begin{aligned} z^t &= z^{t-1} - sH^{-1}\nabla = (1-s)z^{t-1} + sH^{-1}\begin{pmatrix} 1^T \\ K \end{pmatrix} I_{\mathcal{E}} y = \\ &= (1-s)z^{t-1} + s\begin{pmatrix} 0 & 1^T \\ I_{\mathcal{E}}1 + \gamma_I L1 & I_{\mathcal{E}}K + \gamma_A I + \gamma_I LK \end{pmatrix}^{-1} \begin{pmatrix} -\gamma_A & 1^T \\ 0 & K \end{pmatrix}^{-1} \begin{pmatrix} 1^T \\ K \end{pmatrix} I_{\mathcal{E}} y = \\ &= (1-s)z^{t-1} + s\begin{pmatrix} 0 & 1^T \\ I_{\mathcal{E}}1 + \gamma_I L1 & I_{\mathcal{E}}K + \gamma_A I + \gamma_I LK \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ I_{\mathcal{E}} y \end{pmatrix}. \end{aligned} \tag{10}$$

The step size s must be computed by solving the one-dimensional minimization of Equation 7 restricted on the ray from z^{t-1} to z^t, with exact line search or backtracking (Boyd and Vandenberghe, 2004). Convergence is declared when the set of error vectors does not change between two consecutive iterations of the algorithm. Exactly like in the case of primal SVMs (Chapelle, 2007), in our experiments setting $s = 1$ did not result in any convergence problems.

3.1.1 COMPLEXITY ANALYSIS

Updating the α coefficients with the Newton's method costs $O(n^3)$, due to the matrix inversion in the update rule, the same complexity of the original LapSVM solution based on the dual problem discussed in Section 2.1. Convergence is usually achieved in a tiny number of iterations, no more than 5 in our experiments (see Section 6). In order to reduce the cost of each iteration, a Cholesky factorization of the Hessian can be computed before performing the first matrix inversion, and it can be updated using a rank-1 scheme during the following iterations, with cost $O(n^2)$ for each update (Seeger, 2008). On the other hand, this does not allow us to simplify K in Equation 10, otherwise the resulting matrix to be inverted will not be symmetric. Since a lot of time is wasted in the product by K (that is usually dense), using the update of Cholesky factorization may not necessarily lead to a reduction of the overall training time.

It is interesting to compare the training of SVMs in the primal with the one of LapSVMs for a better insight in the Newton's method based solution. Given the set $\mathcal{E}$ at a generic iteration, SVMs only require to compute the inverse of the block of the Hessian matrix that is related to the error vectors, and the complexity of the inversion is then $O(|\mathcal{E}|^3)$ (see Chapelle, 2007). Exploiting this useful aspect, the training algorithm can be run incrementally, reducing the complexity of the whole training process. In the case of LapSVM those benefits are lost due to the presence of the intrinsic norm $f^T Lf$. The additional penalty $w_{ij}(f(x_i) - f(x_j))^2$ makes the Hessian a full matrix, making the block inversion impossible.

Finally, we are assuming that K and the matrix to invert on Equation 10 are non singular, otherwise the final expansion of f will not be unique, even if the optimal value of the criterion function of Equation 7 will be.

3.2 Preconditioned Conjugate Gradient

Instead of performing a costly Newton's step, the vector z for which $\nabla = 0$ can be computed by Conjugate Gradient (CG) descent. In particular if we look at Equation 9, we can write $\nabla = Hz - c$ and, consequently, we have to solve the system $Hz = c$,

$$Hz = c \Longrightarrow \begin{pmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T L 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T L K \\ K I_{\mathcal{E}} 1 + \gamma_I K L 1 & K I_{\mathcal{E}} K + \gamma_A K + \gamma_I K L K \end{pmatrix} z = \begin{pmatrix} 1^T I_{\mathcal{E}} y \\ K I_{\mathcal{E}} y \end{pmatrix}. \tag{11}$$

The convergence rate of CG is related to the condition number of H (Shewchuk, 1994). In the most general case, the presence of the terms $KI_{\mathcal{E}}K$ and KLK leads to a not so well conditioned system and to a slow convergence rate.

In order to overcome this issue, Preconditioned Conjugate Gradient (PCG) can be exploited (Shewchuk, 1994). Given a preconditioner P, the algorithm indirectly solves the system of Equation 11 by solving $\hat{H}z = \hat{c}$, where $\hat{H} = P^{-1}H$ and $\hat{c} = P^{-1}c$. P is selected so that the condition number of $\hat{H}z = \hat{c}$ is improved with respect to the initial system, leading to a faster convergence rate of the iterative method. Moreover, P^{-1} must be easily computable for PCG to be efficient. In the specific case of LapSVM, we can follow a similar strategy to the one investigated by Chapelle (2007), due to the quadratic form of the intrinsic regularizer. In particular, we can factorize Equation 11 as

$$\begin{pmatrix} 1 & 0^T \\ 0 & K \end{pmatrix} \begin{pmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T L 1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T L K \\ I_{\mathcal{E}} 1 + \gamma_I L 1 & I_{\mathcal{E}} K + \gamma_A I + \gamma_I L K \end{pmatrix} z = \begin{pmatrix} 1 & 0^T \\ 0 & K \end{pmatrix} \begin{pmatrix} 1^T I_{\mathcal{E}} y \\ I_{\mathcal{E}} y \end{pmatrix}, \tag{12}$$

and select as a preconditioner the symmetric matrix $P = \begin{pmatrix} 1 & 0^T \\ 0 & K \end{pmatrix}$. We can see that P is a factor of H and c, hence the terms $\hat{H}$ and $\hat{c}$ (and, consequently, the preconditioned gradient $\hat{\nabla}$, given by $\hat{\nabla} = P^{-1}\nabla = \hat{H}z - \hat{c}$) can be trivially computed without explicitly performing any matrix inversions. The condition number of the preconditioned system is sensibly decreased with respect to the one of Equation 11, since $KI_{\mathcal{E}}K$ and KLK are reduced to $I_{\mathcal{E}}K$ and LK. Note that $\hat{H}$ is not symmetric, and it would not possible, for instance, to simply remove the factor P in both sides of Equation 12 and solve it by standard CG. For those reasons, PCG is appropriate for an efficient optimization of our problem. As in the Newton's method, we are assuming that K is non singular, otherwise a small ridge can be added to fix it.

The iterative solution of the LapSVM problem by means of PCG is reported in Algorithm 1. For an easier comparison with the standard formulation of PCG, consider that the vectors of residual of the original and preconditioned systems corresponds to $-\nabla$ and $-\hat{\nabla}$, respectively. Nevertheless, due to our choice of P, we do not need to compute ∇ first, and then $\hat{\nabla} = P^{-1}\nabla$. We can exchange the order of those operations to avoid the matrix inversion, that is, first compute $\hat{\nabla}$ and then $\nabla = P\hat{\nabla}$. Hence, P^{-1} never appears in Algorithm 1.

Classic rules for the update of the conjugate direction at each step are discussed by Shewchuk (1994). After several iterations the conjugacy of the descent directions tends to get lost due to round-off floating point error, so a restart of the preconditioned conjugate gradient algorithm is required. The Fletcher-Reeves (FR) update is commonly used in linear optimization. Due to the piecewise nature of the problem, defined by the $I_{\mathcal{E}}$ matrix, we exploited the Polak-Ribiere (PR) formula,[1]

1. Note that in the linear case FR and PR are equivalent.

where restart can be automatically performed when the update term becomes negative. In that case, the ρ coefficient in Algorithm 1 becomes zero, and the following iteration corresponds to a steepest descent one, as when PCG starts. We experimentally evaluated that for the LapSVM problem such formula is generally the best choice, both for convergence speed and numerical stability.

Convergence is usually declared when the norm of the preconditioned gradient falls below a given threshold (Chapelle, 2007), or when the current preconditioned gradient is roughly orthogonal with the real gradient (Shewchuk, 1994). We will investigate these conditions in Section 4.

Algorithm 1 Preconditioned Conjugate Gradient (PCG) for primal LapSVMs.

```
Let $t = 0, z^t = 0, \mathcal{E} = \mathcal{L}, \hat{\nabla}^t = [-1^T y, -y^T]^T, d^t = -\hat{\nabla}^t$
repeat
    $t = t + 1$
    Find $s^*$ by line search on the line $z^{t-1} + sd^{t-1}$
    $z^t = z^{t-1} + s^* d^{t-1}$
    $\mathcal{E} = \{x_i \in \mathcal{L} \;\; s.t. \;\; (k_i \alpha^t + b^t) y_i < 1\}$
    $\hat{\nabla}^t = \hat{H}z - \hat{c} = \begin{pmatrix} 1^T I_{\mathcal{E}} 1 + \gamma_I 1^T L1 & 1^T I_{\mathcal{E}} K + \gamma_I 1^T LK \\ I_{\mathcal{E}} 1 + \gamma_I L1 & I_{\mathcal{E}} K + \gamma_A I + \gamma_I LK \end{pmatrix} z - \begin{pmatrix} 1^T I_{\mathcal{E}} y \\ I_{\mathcal{E}} y \end{pmatrix}$
    $\nabla^t = Hz - c = P\hat{H}z - P\hat{c} = P\hat{\nabla}^t$
    $\rho = \max\left(\frac{\nabla^{t^T}(\hat{\nabla}^t - \hat{\nabla}^{t-1})}{\nabla^{t-1^T}\hat{\nabla}^{t-1}}, 0\right)$
    $d^t = -\hat{\nabla}^t + \rho d^{t-1}$
until Goal condition
```

3.2.1 LINE SEARCH

The optimal step length s^* on the current direction of the PCG algorithm must be computed by backtracking or exact line search. At a generic iteration t we have to solve

$$s^* = \arg\min_{s \geq 0} obj(z^{t-1} + sd^{t-1}) \tag{13}$$

where obj is the objective function of Equation 7.

The accuracy of the line search is crucial for the performance of PCG. When minimizing a quadratic form that leads to a linear expression of the gradient, line search can be computed in closed form. In our case, we have to deal with the variations of the set $\mathcal{E}$ (and of $I_{\mathcal{E}}$) for different values of s, so that a closed form solution cannot be derived, and we have to compute the optimal s in an iterative way.

Due to the quadratic nature of Equation 13, the 1-dimensional Newton's method can be directly used, but the average number of line search iterations per PCG step can be very large, even if the cost of each of them is negligible with respect to the $O(n^2)$ of a PCG iteration. We can efficiently solve the line search problem analytically, as suggested by Keerthi and DeCoste (2005) for SVMs.

In order to simplify the notation, we discard the iteration index $t-1$ in the following description. Given the PCG direction d, we compute for each point $x_i \in \mathcal{L}$, being it an error vector or not, the step length s_i for which its state switches. The state of a given error vector switches when it leaves

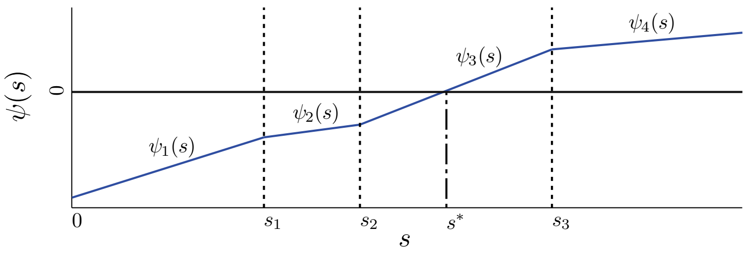


Figure 2: Example of the piecewise linear function $\psi(s)$ (blue plot). $\psi_1(s),\ldots,\psi_4(s)$ are the four linear portions of $\psi(s)$, and s_1, s_2, s_3 are the break points. The optimal step length, s^*, is the value for which $\psi(s)$ crosses zero.

the $\mathcal{E}$ set, whether the state of a point initially not in $\mathcal{E}$ switches when it becomes an error vector. We refer to the set of the former points with $\mathcal{Q}_1$ while the latter is $\mathcal{Q}_2$, with $\mathcal{L} = \mathcal{Q}_1 \cup \mathcal{Q}_2$. The derivative of Equation 13, $\psi(s) = \partial obj(z+sd)/\partial s$, is piecewise linear, and s_i are the break points of such function.

Let us consider, for simplicity, that s_i are in a non decreasing order, discarding the negative ones. Starting from $s = 0$, they define a set of intervals where $\psi(s)$ is linear and the $\mathcal{E}$ set does not change. We indicate with $\psi_j(s)$ the linear portion of $\psi(s)$ in the j-th interval. Starting with $j = 1$, if the value $s \geq 0$ where the line $\psi_j(s)$ crosses zero is within such interval, then it is the optimal step size s^*, otherwise the following interval must be checked. The convergence of the process is guaranteed by the convexity of the function obj. See Figure 2 for a basic example.

The zero crossing of $\psi_j(s)$ is given by $s = \frac{\psi_j(0)}{\psi_j(0)-\psi_j(1)}$, where the two points $(0,\psi_j(0))$ and $(1,\psi_j(1))$ determine the line $\psi_j(s)$. We indicate with $f_d(x)$ the function $f(x)$ whose coefficients are in $d = [d_b, d_\alpha^T]^T$, that is, $f_d(x_i) = k_i^T d_\alpha + d_b$, and $f_d = [f_d(x_i), x_i \in \mathcal{S}]^T$. We have

$$\begin{aligned}
\psi_j(0) &= \textstyle\sum_{x_i\in\mathcal{E}_j} (f(x_i)-y_i)f_d(x_i) + \gamma_A \alpha^T K d_\alpha + \gamma_I f_d^T L f,\\
\psi_j(1) &= \textstyle\sum_{x_i\in\mathcal{E}_j} (f(x_i)+f_d(x_i)-y_i)f_d(x_i) + \gamma_A(\alpha+d_\alpha)^T K d_\alpha + \gamma_I f_d^T L(f+f_d)
\end{aligned}$$

where $\mathcal{E}_j$ is the set of error vectors for the j-th interval.

Given $\psi_1(0)$ and $\psi_1(1)$, their successive values for increasing j can be easily computed considering that only one point (that we indicate with x_j) switches status moving from an interval to the following one. From this consideration we derived the following update rules

$$\begin{aligned}
\psi_{j+1}(0) &= \psi_j(0) + \nu_j(f(x_j)-y_j)f_d(x_j),\\
\psi_{j+1}(1) &= \psi_j(1) + \nu_j(f(x_j)+f_d(x_j)-y_i)f_d(x_j)
\end{aligned}$$

where ν_j is -1 if $x_j \in \mathcal{Q}_1$ and it is $+1$ if $r \in \mathcal{Q}_2$.

3.2.2 COMPLEXITY ANALYSIS

Each PCG iteration requires to compute the $K\alpha$ product, leading to a complexity of $O(n^2)$ to update the α coefficients. The term $LK\alpha$ can then be computed efficiently from $K\alpha$, since the matrix L is generally sparse. Note that, unlike the Newton's method and the original dual solution of the LapSVM problem, we never have to explicitly compute the LK product, always computing matrix by vector products instead. Even if L is sparse, when the number of training points is large or

L is iterated several times, a large amount of computation may be saved by avoiding such matrix by matrix product, as we will show in Section 6. Moreover, if the kernel matrix is sparse, the complexity drops to $O(n_{nz})$, where n_{nz} is the maximum number of non-zero elements between K and L. Note that the algorithm does not necessarily need to hold the whole matrix K (and L) in memory. The only requirement is a fast way to perform the product of K with the current α. On the other hand, computing each kernel function evaluation on the fly may require a large number of floating-point operations, so that some caching procedures must be devised.

Convergence of the conjugate gradient algorithm is theoretically declared in $O(n)$ steps, but a solution very close to the optimal one can be computed with far less iterations. The convergence speed is related to the condition number of the Hessian (Shewchuk, 1994), that it is composed by a sum of three contributes (Equation 11). As a consequence, their condition numbers and weighting coefficients (γ_A, γ_I) have a direct influence in the convergence speed, and in particular the condition number of the K matrix. For example, using a bandwidth of a Gaussian kernel that lead to a K matrix close to the identity allows the algorithm to converge very quickly, but the accuracy of the classifier may not be sufficient.

Finally, PCG can be efficiently seeded with an initial rough estimate of the solution ('warm" or "hot" start). For example, the solution computed for some given values of the γ_A and γ_I parameters can be a good starting point when training the classifier with some just slightly different parameter values (i.e., when cross-validating the model). Seeding is also crucial in schemes that allow the classifier to be incrementally built with reduced complexity. They have been deeply investigated by Keerthi et al. (2006) for the SVM classifier. Even if Keerthi et al. (2006) use the Newton optimization, a similar approach could be studied for LapSVMs exploiting the useful properties of the PCG algorithm.

4. Approximating the Optimal Solution

In order to reduce the training times, we want the PCG to converge as fast as possible to a good *approximation* of the optimal solution. By appropriately selecting the goal condition of Algorithm 1, we can discard iterations that may not lead to significant improvement in the classifier quality. This concept is widely used in optimization, where the early stop of the CG or PCG is exploited to approximately solve the Newton system in truncated Newton methods (see, for example, the trust region method for large-scale logistic regression of Lin et al., 2008).

The common goal conditions for the PCG algorithm and, more generally, for gradient based iterative algorithms, rely on the norm of the gradient $\|\nabla\|$ (Boyd and Vandenberghe, 2004), of the preconditioned gradient $\|\hat{\nabla}\|$ (Chapelle, 2007), on the mixed product $\sqrt{\hat{\nabla}^T\nabla}$ (Shewchuk, 1994). These values are usually normalized by the first estimate of each of them. The value of the objective function obj or its relative decrement between two consecutive iterations can also be checked, requiring some additional computations since the PCG algorithm never explicitly computes it. When one of such "stopping" values falls below the chosen threshold τ associated to it, the algorithm terminates.[2] Moreover, a maximum number t_{max} of iterations is generally specified. Tuning these parameters is crucial both for the time spent running the algorithm and the quality of the resulting solution.

2. Thresholds associated to different conditions are obviously different, but, for simplicity in the description, we will refer to a generic threshold τ.

It is really hard to find a trade-off between good approximation and low number of iterations, since τ and t_{max} are strictly problem dependent. As an example, consider that the surface of obj, the objective function of Equation 7, varies among different choices of its parameters. Increasing or decreasing the values of γ_A and γ_I can lead to a less flat or a more flat region around the optimal point. Fixing in advance the values of τ and t_{max} may cause an early stop too far from the optimal solution, or it may result in the execution of a large number of iterations without a significant improvement on the classification accuracy.

The latter situation can be particularly frequent for LapSVMs. As described in Section 2 the choice of the intrinsic norm $f^T L f$ introduces the soft constraint $f(x_i) = f(x_j)$ for nearby points x_i, x_j along the underlying manifold. This allows the algorithm to perform a graph transduction and diffuse the labels from points in $\mathcal{L}$ to the unlabeled data $\mathcal{U}$.

When the diffusion is somewhat complete and the classification hyperplane has assumed a quite stable shape around the available training data, similar to the optimal one, the intrinsic norm will keep contributing to the gradient until a balance with respect to the ambient norm (and to the L_2 loss on error vectors) is found. Due to the strictness of this constraint, it will still require some iterations (sometimes many) to achieve the optimal solution with $\|\nabla\| = 0$, even if the decision function $y(x) = sign(f(x))$ will remain substantially the same. The described common goal conditions do not "directly" take into account the decision of the classifier, so that they do not appear appropriate to early stop the PCG algorithm for LapSVMs.

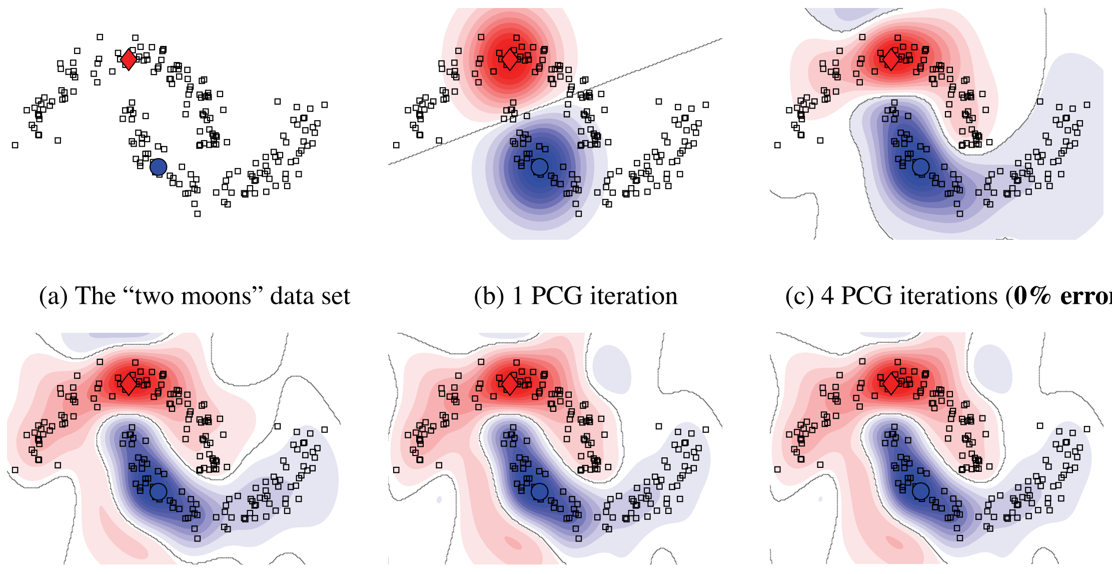
We investigate our intuition on the "two moons" data set of Figure 3(a), where we compare the decision boundary after each PCG iteration (Figure 3(b)-(e)) with the optimal solution (computed by Newton's method, Figure 3(f)). Starting with $\alpha = 0$, the first iteration exploits only the gradient of the L_2 loss on labeled points, since both the regularizing norms are zero. In the following iterations we can observe the label diffusion process along the manifold. After only 4 iterations we get a perfect classification of the data set and a separating boundary not far from the optimal one. All the remaining iterations until complete convergence are used to slightly asses the coherence along the manifold required by the intrinsic norm and the balancing with the smoothness of the function, as can be observed by looking at the function values after 25 iterations. The most of changes influences regions far from the support of P_X, and it is clear that an early stop after 4 PCG steps would be enough to roughly approximate the accuracy of optimal solution.

In Figure 4 we can observe the values of the previously described general stopping criterion for PCG. After 4 iterations they are still sensibly decreasing, without reflecting real improvements in the classifier quality. The value of the objective function obj starts to become more stable only after, say, 16 iterations, but it is still slightly decreasing even if it appears quite horizontal on the graph, due to its scale. It is clear that fixing in advance the parameters τ and t_{max} is random guessing and it will probably result in a bad trade-off between training time and accuracy.

4.1 Early Stopping Conditions

Following these considerations, we propose to early stop the PCG algorithm exploiting the predictions of the classifier on the available data.

Due to the high amount of unlabeled training points in the semi-supervised learning framework, the stability of the decision $y(x) = sign(f(x))$, $x \in \mathcal{U}$, can be used as a reference to early stop the gradient descent (*stability check*). Moreover, if labeled validation data (set $\mathcal{V}$) is available for


(a) The “two moons” data set (b) 1 PCG iteration (c) 4 PCG iterations (**0% error**)

(d) 8 PCG iterations (e) 25 PCG iterations (f) Optimal solution

Figure 3: (a) The “two moons” data set (200 points, 2 classes, 2 labeled points indicated with a red diamond and a blue circle, whereas the remaining points are unlabeled) - (b-e) A LapSVM classifier trained with PCG, showing the result after a fixed number of iterations. The dark continuous line is the decision boundary ($f(x) = 0$) and the confidence of the classifier ranges from red ($f(x) \geq 1$) to blue ($f(x) \leq -1$) - (f) The optimal solution of the LapSVM problem computed by means of Newton’s method

classifier parameters tuning, we can formulate a good stopping condition based on the classification accuracy on it (*validation check*), that can be eventually merged to the previous one (*mixed check*).

In detail, when $y(x)$ becomes quite stable between consecutive iterations or when $err(\mathcal{V})$, the error rate on $\mathcal{V}$, is not decreasing anymore, then the PCG algorithm should be stopped. Due to their heuristic nature, it is generally better to compare the predictions every θ iterations and within a certain tolerance η. As a matter of fact, $y(x)$ may slightly change also when we are very close to the optimal solution, and $err(\mathcal{V})$ is not necessarily an always decreasing function. Moreover, labeled validation data in the semi-supervised setting is usually small with respect to the whole training data, labeled and unlabeled, and it may not be enough to represent the structure of the data set.

We propose very simple implementations of such conditions, that we used to achieve the results of Section 6. Starting from these, many different and more efficient variants can be formulated, but it goes beyond the scope of this paper. They are sketched in Algorithms 2 and 3. We computed the classifier decision every $\sqrt{n}/2$ iterations and we required the classifier to improve $err(\mathcal{V})$ by one correctly classifier example at every check, due to the usually small size of $\mathcal{V}$. Sometimes this can also help to avoid a slight overfitting of the classifier.

Generating the decision $y(x)$ on unlabeled data does not require heavy additional machinery, since the $K\alpha$ product must be necessarily computed to perform every PCG iteration. Its overall cost is $O(u)$. Differently, computing the accuracy on validation data requires the evaluation of the kernel

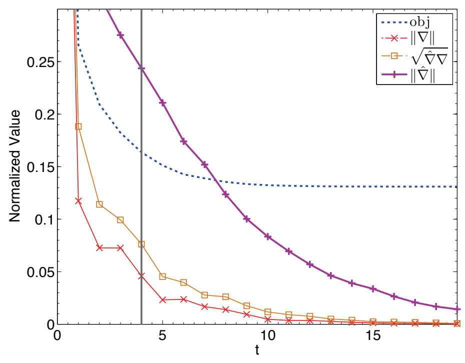


Figure 4: PCG example on the "two moons" data set. The norm of the gradient $\|\nabla\|$, of the preconditioned gradient $\|\hat{\nabla}\|$, the value of the objective function obj and of the mixed product $\sqrt{\hat{\nabla}^T\nabla}$ are displayed in function of the number of PCG iterations. The vertical line represents the number of iterations after which the error rate is 0% and the decision boundary is quite stable.

Algorithm 2 The *stability check* for PCG stopping.

```
  $d^{old} \leftarrow 0 \in \mathbb{R}^u$
  $\eta \leftarrow 1.5\%$
  $\theta \leftarrow \sqrt{n}/2$
  *Every* $\theta$ *iterations do the followings:*
  $d = [y(x_j), x_j \in \mathcal{U}, j = 1, \ldots, u]^T$
  $\tau = (100 \cdot \|d - d^{old}\|_1/u)\%$
  **if** $\tau < \eta$ **then**
     Stop PCG
  **else**
     $d^{old} = d$
  **end if**
```

Algorithm 3 The *validation check* for PCG stopping.

```
**Require:** $\mathcal{V}$
  $err\mathcal{V}^{old} \leftarrow 100\%$
  $\eta \leftarrow 100 \cdot |\mathcal{V}|^{-1}\%$
  $\theta \leftarrow \sqrt{n}/2$
  *Every* $\theta$ *iterations do the followings:*
  **if** $err(\mathcal{V}) > (err\mathcal{V}^{old} - \eta)$ **then**
     Stop PCG
  **else**
     $err\mathcal{V}^{old} = err(\mathcal{V})$
  **end if**
```

function on validation points against the n training ones, and $O(|\mathcal{V}| \cdot n)$ products, that is negligible with respect to the cost of a PCG iteration.

Please note that even if these are generally early stopping conditions, sometimes they can help in the opposite situation. For instance they can also detect that the classifier needs to move some more steps toward the optimal solution than the ones limited by the selected t_{max}.

The proposed stopping criteria could be exploited in the optimization of alternative formulations of the LapSVM problem (following the improved models of Abernethy et al., 2008 and of Tsang and Kwok, 2006), with the aim of reducing training times and getting a classifier with a roughly comparable quality to the optimal one. Even with slightly different problem formulations, our criteria are reasonably more appropriate than classical goal conditions due to their direct relationship with the stability of the classifier prediction. In particular, some additional efficient solution strategies may be devised by directly working in the primal and exploiting the ϵ-insensitive loss based intrinsic regularizer of Tsang and Kwok (2006), where manifold regularization is applied to a large-scale setting in the Minimum Enclosing Ball (MEB) framework. We note these directions for future work.

5. Laplacian Regularized Least Squares

Laplacian Regularized Least Square Classifier (LapRLSC) has many analogies with the proposed L_2 hinge loss based LapSVMs. LapRLSC uses a squared loss function to penalize wrongly classified examples, leading to the following objective function

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{l} (y_i - f(x_i))^2 + \gamma_A \|f\|_A^2 + \gamma_I \|f\|_I^2.$$

The optimal α coefficients and the optimal bias b, collected in the vector z, can be obtained by solving the linear system

$$\begin{pmatrix} |\mathcal{L}| + \gamma_I 1^T L 1 & 1^T I_{\mathcal{L}} K + \gamma_I 1^T L K \\ K I_{\mathcal{L}} 1 + \gamma_I K L 1 & K I_{\mathcal{L}} K + \gamma_A K + \gamma_I K L K \end{pmatrix} z = \begin{pmatrix} 1^T y \\ K y \end{pmatrix} \tag{14}$$

where $I_{\mathcal{L}}$ is the diagonal matrix $\in \mathbb{R}^{n,n}$ with the first l elements equal to 1 and the remaining u elements equal to zero.

Following the notation used for LapSVMs, in LapRLSCs we have a set of error vectors $\mathcal{E}$ that is actually fixed and equal to $\mathcal{L}$. As a matter of fact a LapRLSC requires the estimated function to interpolate the given targets in order to not incur in a penalty. In a hypothetic situation where all the labeled examples always belong to $\mathcal{E}$ during the training of a LapSVM classifier in the primal, then the solution will be the same of LapRLSC.

Solving the least squares problem of LapRLSC can be performed by matrix inversion, after factoring and simplifying the previously defined matrix P in Equation 14. Otherwise the proposed PCG approach and the early stopping conditions can be directly used. In this case the classic instruments for linear optimization apply, and the required line search of Equation 13 can be computed in closed form without the need of an iterative process,

$$s^* = -\frac{\nabla^T d}{d^T H d}$$

where ∇ and H are no more functions of $\mathcal{E}$.

As shown by Belkin et al. (2006); Sindhwani and Rosenberg (2008) and in the experimental section of this paper, LapRLSC, LapSVM and primal LapSVM allow us to achieve similar classification performances. The interesting property of the LapSVM problem is that the effect of the regularization terms at a given iteration can be decoupled by the one of the loss function on labeled points, since the gradient of the loss function for correctly classified points is zero and do not disturb classifier design. This characteristic can be useful as a starting point for the study of some alternative formulations of the intrinsic norm regularizer.

6. Experimental Results

We ran a wide set of experiments to analyze the proposed solution strategies of the primal LapSVM problem. In this section we describe the selected data sets, our experimental protocol and the details on the parameter selection strategy. Then we show the main result of the proposed approach, very fast training of the LapSVM classifier with reduced complexity by means of early stopped PCG. We compare the quality of the L_2 hinge loss LapSVMs trained in the primal by Newton's method with respect to the L_1 hinge loss dual formulation and LapRLSCs. Finally, we describe the convergence speed and the impact on performances of our early stopping conditions.

As a baseline reference for the performances in the supervised setting, we selected two popular regularized classifiers, Support Vector Machines (SVMs) and Regularized Least Square Classifiers (RLSCs). We implemented and tested all the algorithms using Matlab 7.6 on a 2.33Ghz machine with 6GB of memory. The dual problem of LapSVM has been solved using the latest version of Libsvm (Fan et al., 2005). Multiclass classification has been performed using the one-against-all approach.

6.1 Data Sets

We selected eight popular data sets for our experiments. Most of them data sets has been already used in previous works to evaluate several semi-supervised classification algorithms (Sindhwani et al., 2005; Belkin et al., 2006; Sindhwani and Rosenberg, 2008), and all of them are available on the Web. G50C[3] is an artificial data set generated from two unit covariance normal distributions with equal probabilities. The class means are adjusted so that the Bayes error is 5%. The COIL20 data set is a collection of pictures of 20 different objects from the Columbia University. Each object has been placed on a turntable and at every 5 degrees of rotation a 32x32 gray scale image was acquired. The USPST data set is a collection of handwritten digits form the USPS postal system. Images are acquired at the resolution of 16x16 pixels. USPST refers to the test split of the original data set. We analyzed the COIL20 and USPST data set in their original 20 and 10-class versions and also in their 2-class versions, to discard the effects on performances of the selected multiclass strategy. COIL20(B) discriminates between the first 10 and the last 10 objects, whereas USPST(B) from the first 5 digits and the remaining ones. PCMAC is a two-class data set generated from the famous 20-Newsgroups collection, that collects posts on Windows and Macintosh systems. MNIST3VS8 is the binary version of the MNIST data set, a collection of 28x28 gray scale handwritten digit images from NIST. The goal is to separate digit 3 from digit 8. Finally, the FACEMIT data set of the Center for Biological and Computational Learning at MIT contains 19x19 gray scale, PGM format, images of faces and non-faces. The details of the described data sets are resumed in Table 1.

3. It can be downloaded from `http://people.cs.uchicago.edu/~vikass/manifoldregularization.html`.

Data Set	**Classes**	**Size**	**Attributes**
G50C	2	550	50
COIL20(B)	2	1440	1024
PCMAC	2	1946	7511
USPST(B)	2	2007	256
COIL20	20	1440	1024
USPST	10	2007	256
MNIST3VS8	2	13966	784
FACEMIT	2	31022	361

Table 1: Details of the data sets that have been used in the experiments.

6.2 Experimental Protocol

All presented results has been obtained by averaging them on different splits of the available data. In particular, a 4-fold cross-validation has been performed, randomizing the fold generation process for 3 times, for a total of 12 splits. Each fold contains the same number of per class examples as in the complete data set. For each split, we have 3 folds that are used for training the classifier and the remaining one that constitutes the test set ($\mathcal{T}$). Training data has been divided in labeled ($\mathcal{L}$), unlabeled ($\mathcal{U}$) and validation sets ($\mathcal{V}$), where the last one is only used to tune the classifier parameters. The labeled and validation sets have been randomly selected from the training data such that at least one example per class is assured to be present on each of them, without any additional balancing constraints. A small number of labeled points has been generally selected, in order to simulate a semi-supervised scenario where labeling data has a large cost. The MNIST3VS8 and FACEMIT data set are already divided in training and test data, so that the 4-fold generation process was not necessary, and just the random subdivision of training data has been performed (balancing the class labels on training and validation data). In particular, on the MNIST3VS8 collection we normalized the data vectors to unit norm, and on the FACEMIT data set we exchanged the original training and test sets, since, as a matter of fact, the latter is sensibly larger that the former. In this case our goal is just to show how we were able to handle a high amount of training data using the proposed primal solution with PCG, whereas it was not possible to do it with the original dual formulation of LapSVM. Due to the high unbalancing of such data set, we report the macro error rates for it ($1-TP/2+TN/2$, where TP and TN are the rates of true positives and true negatives). Details are collected in Table 2.

6.3 Parameters

We selected a Gaussian kernel function in the form $k(x_i,x_j) = \exp\left(-\frac{||x_i-x_j||}{2\sigma^2}\right)$ for each experiment, with the exception of the MNIST3VS8 where a polynomial kernel of degree 9 was used, as suggest by Decoste and Schölkopf (2002). The other parameters were selected by cross-validating them on the $\mathcal{V}$ set. In order to speedup this step, the values of the Gaussian kernel width and of the parameters required to build the graph Laplacian (the number of neighbors, *nn*, and the degree, *p*) for the first six data sets were fixed as specified by Sindhwani and Rosenberg (2008). For details on the selection of such parameters please refer to Sindhwani and Rosenberg (2008); Sindhwani et al. (2005). The graph Laplacian was computed by using its normalized expression. The optimal

Data Set	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{T}\rvert$
G50C	50	314	50	136
COIL20(B)	40	1000	40	360
PCMAC	50	1358	50	488
USPST(B)	50	1409	50	498
COIL20	40	1000	40	360
USPST	50	1409	50	498
MNIST3VS8	80	11822	80	1984
FACEMIT	2	23973	50	6997

Table 2: The number of data points in each split of the selected data sets, where $\mathcal{L}$ and $\mathcal{U}$ are the sets of labeled and unlabeled training points, respectively, $\mathcal{V}$ is the labeled set for cross-validating parameters whereas $\mathcal{T}$ is the out-of-sample test set.

weights of the ambient and intrinsic norms, γ_A, γ_I, were determined by varying them on the grid $\{10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ and chosen with respect to validation error. For the FACEMIT data set also the value 10^{-8} was considered, due to the high amount of training points. The selected parameter values are reported in Table 9 of Appendix A for reproducibility of the experiments.

6.4 Results

Before going into further detail, in Table 3 we report the training times of LapSVMs using the original dual formulation and the primal training approach.[4] The last column refers to LapSVMs trained using the best (in terms of accuracy) of the proposed stopping heuristics for each specific data set. As expected, training in the primal by the Newton's method requires training times similar to those for the dual formulation. On the other hand, training by PCG with the proposed early stopping conditions shows an appreciable reduction of training times for all data sets. As the size of labeled and unlabeled points increases, the improvement becomes very evident. On the MNIST3VS8 data set we go from roughly half an hour to two minutes. Both in the dual formulation of LapSVMs and in the primal one solved by means of Newton's method, a lot of time is spent in computing the LK matrix product. Even if L is sparse, the cost of this product could be quite high. Similar reductions are observed for the PCMAC data set, where the training time drops from 15 seconds to only 2 seconds when solving with PCG. Finally, the memory requirements are also reduced, since, when the PCG is used, there is no need to explicitly compute, store and invert the Hessian. To emphasize this point, we had no difficulty training the classifier on the FACEMIT data set using PCG. On the other hand, the high memory requirements of dual LapSVM and primal LapSVM solved with Newton's method, coupled with the high computational cost, made those methods impossible to runt on our machine.

We now investigate the details of the solution of the primal LapSVM problem. In order to compare the effects of the different loss functions of LapRLSCs, LapSVMs trained in the dual, and LapSVMs trained in the primal, in Table 4 the classification errors of the described techniques are reported. For this comparison, the solution of primal LapSVMs is computed by means of the Newton's method. The manifold regularization based techniques lead to comparable results, and,

4. For a fair comparison of the training algorithms, the Gram matrix and the Laplacian were precomputed.

Data Set	Laplacian SVMs		
	Dual [Original]	**Primal - Newton**	**Primal - PCG**
G50C	0.155 (0.004)	0.134 (0.006)	**0.043** (0.006)
COIL20(B)	0.311 (0.012)	0.367 (0.097)	**0.097** (0.026)
PCMAC	14.82 (0.104)	15.756 (0.285)	**1.967** (0.269)
USPST(B)	1.196 (0.015)	1.4727 (0.2033)	**0.300** (0.030)
COIL20	6.321 (0.441)	7.26 (1.921)	**3.487** (1.734)
USPST	12.25 (0.2)	17.74 (2.44)	**2.032** (0.434)
MNIST3VS8	2064.18 (3.1)	2824.174 (105.07)	**114.441** (0.235)
FACEMIT	-	-	**35.728** (0.868)

Table 3: Our main result. Training times (in seconds) of Laplacian SVMs using different algorithms (standard deviation in brackets). The time required to solve the original dual formulation and the primal solution with Newton's method are comparable, whereas solving the Laplacian SVMs problem in the primal with early stopped preconditioned conjugate gradient (PCG) offers a noticeable speedup.

as expected, all semi-supervised approaches show a sensible improvement over classical supervised classification algorithms. The error rates of primal LapSVMs and LapRLSCs are quite close, due to the described relationship of the L_2 hinge loss and the squared loss. We reported the average number of Newton's steps required to compute the solution in Table 5. In all our experiments we have observed convergence in less than 6 steps.

We compared the error rates of LapSVMs trained in the primal by Newton's method with ones of PCG training, in function of the number of gradient steps t. For this comparison, γ_A and γ_I were selected by cross-validating with the former (see Appendix A), and experiments were performed using all the described data sets. In Figure 5-7 we report the graphs in the case of the USPST, MNIST3VS8 and COIL20 data as a reference. The horizontal line on each graph represents the error rate of the non-approximated solution computed with the Newton's method. The number of iterations required to converge to a solution with the same accuracy of the non-approximated one is sensibly smaller than n. Convergence is achieved really fast, and only in the COIL20 data set we experienced a relatively slower rate with respect to the other data sets. The error surface of each binary classifier is quite flat around optimum with the selected γ_A and γ_I, leading to some round-off errors in gradient descent based techniques, stressed by the large number of classes and the one-against-all approach. Moreover labeled training examples are highly unbalanced. As a matter of fact, in the COIL20(B) data set we did not experience this behavior. Finally, in the FACEMIT data set the algorithm perfectly converges in a few iterations, showing that in this data set the most of information is contained in the labeled data (even if it is very small), and the intrinsic constraint is easily fulfilled.

In Figure 8-9 we collected the values of the gradient norm $||\nabla||$, of the preconditioned gradient norm $||\hat{\nabla}||$, of the mixed product $\sqrt{\hat{\nabla}^T\nabla}$, and of the objective function obj for each data set, normalized by their respective values at $t = 0$. The vertical line is an indicative index of the number of iterations after which the error rate on all partitions ($\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$) becomes equal to the one at the stationary point (when the gradient of the objective function is zero). The curves generally keep sen-

Data Set	**Classifier**	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{T}$
G50C	SVM	9.33 (2)	9.83 (3.46)	10.06 (2.8)
	RLSC	10.43 (5.26)	10.17 (4.86)	11.21 (4.98)
	LapRLSC	6.03 (1.32)	6.17 (3.66)	6.54 (2.11)
	LapSVM Dual (Original)	5.52 (1.15)	5.67 (2.67)	5.51 (1.65)
	LapSVM Primal (Newton)	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
COIL20(B)	SVM	16.23 (2.63)	18.54 (6.2)	15.93 (3)
	RLSC	16.22 (2.64)	18.54 (6.17)	15.97 (3.02)
	LapRLSC	8.067 (2.05)	7.92 (3.96)	8.59 (1.9)
	LapSVM Dual (Original)	8.31 (2.19)	8.13 (4.01)	8.68 (2.04)
	LapSVM Primal (Newton)	8.16 (2.04)	7.92 (3.96)	8.56 (1.9)
PCMAC	SVM	19.65 (6.91)	20.83 (6.85)	20.09 (6.91)
	RLSC	19.63 (6.91)	20.67 (6.95)	20.04 (6.93)
	LapRLSC	9.67 (0.74)	7.67 (4.08)	9.34 (1.5)
	LapSVM Dual (Original)	10.78 (1.83)	9.17 (4.55)	11.05 (2.94)
	LapSVM Primal (Newton)	9.68 (0.77)	7.83 (4.04)	9.37 (1.51)
USPST(B)	SVM	17 (2.74)	18.17 (5.94)	17.1 (3.21)
	RLSC	17.21 (3.02)	17.5 (5.13)	17.27 (2.72)
	LapRLSC	8.87 (1.88)	10.17 (4.55)	9.42 (2.51)
	LapSVM Dual (Original)	8.84 (2.2)	8.67 (4.38)	9.68 (2.48)
	LapSVM Primal (Newton)	8.72 (2.15)	9.33 (3.85)	9.42 (2.34)
COIL20	SVM	29.49 (2.24)	31.46 (7.79)	28.98 (2.74)
	RLSC	29.51 (2.23)	31.46 (7.79)	28.96 (2.72)
	LapRLSC	10.35 (2.3)	9.79 (4.94)	11.3 (2.17)
	LapSVM Dual (Original)	10.51 (2.06)	9.79 (4.94)	11.44 (2.39)
	LapSVM Primal (Newton)	10.54 (2.03)	9.79 (4.94)	11.32 (2.19)
USPST	SVM	23.84 (3.26)	24.67 (4.54)	23.6 (2.32)
	RLSC	23.95 (3.53)	25.33 (4.03)	24.01 (3.43)
	LapRLSC	15.12 (2.9)	14.67 (3.94)	16.44 (3.53)
	LapSVM Dual (Original)	14.36 (2.55)	15.17 (4.04)	14.91 (2.83)
	LapSVM Primal (Newton)	14.98 (2.88)	15 (3.57)	15.38 (3.55)
MNIST3VS8	SVM	8.82 (1.11)	7.92 (4.73)	8.22 (1.36)
	RLSC	8.82 (1.11)	7.92 (4.73)	8.22 (1.36)
	LapRLSC	1.95 (0.05)	1.67 (1.44)	1.8 (0.3)
	LapSVM Dual (Original)	2.29 (0.17)	1.67 (1.44)	1.98 (0.15)
	LapSVM Primal (Newton)	2.2 (0.14)	1.67 (1.44)	2.02 (0.22)
FACEMIT	SVM	39.8 (2.34)	38 (1.15)	34.61 (3.96)
	RLSC	39.8 (2.34)	38 (1.15)	34.61 (3.96)
	LapSVM Primal (PCG)	29.97 (2.51)	36 (3.46)	27.97 (5.38)

Table 4: Comparison of the accuracy of LapSVMs trained by solving the primal (Newton's method) or the dual problem. The average classification error (standard deviation is reported brackets) is reported. Fully supervised classifiers (SVMs, RLSCs) represent the baseline performances. $\mathcal{U}$ is the set of unlabeled examples used to train the semi-supervised classifiers. $\mathcal{V}$ is the labeled set for cross-validating parameters whereas $\mathcal{T}$ is the out-of-sample test set. Results on the labeled training set $\mathcal{L}$ are omitted since all algorithms correctly classify such a few labeled training points.

Data Set	**Newton's Steps**
G50C	1 (0)
COIL20(B)	2.67 (0.78)
PCMAC	2.33 (0.49)
USPST(B)	4.17 (0.58)
COIL20	2.67 (0.75)
USPST	4.26 (0.76)
MNIST3VS8	5 (0)

Table 5: Newton's steps required to compute the solution of the primal Laplacian SVM problem.

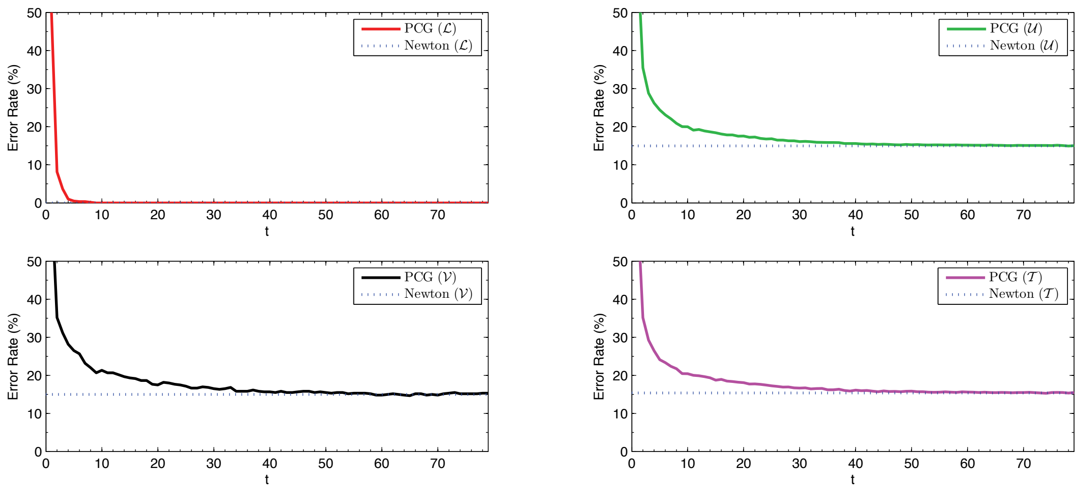


Figure 5: USPST data set: error rate on $\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$ of the Laplacian SVM classifier trained in the primal by preconditioned conjugate gradient (PCG), with respect to the number of gradient steps t. The error rate of the primal solution computed by means of Newton's method is reported as a horizontal line.

sibly decreasing even after such line, without reflecting real improvements in the classifier accuracy, and they differ by orders of magnitude among the considered data set, showing their strong problem dependency (differently from our proposed conditions). As described in Section 4, we can see how it is clearly impossible to define a generic threshold on them to appropriately stop the PCG descent (i.e., to find a good trade-off between number of iterations and accuracy). Moreover, altering the values of the classifier parameters can sensibly change the shape of the error function, requiring a different threshold every time. In those data sets where points keep entering and leaving the $\mathcal{E}$ set as t increases (mainly during the first steps) the norm of the gradient can show an instable behavior between consecutive iterations, due to the piecewise nature of the problem, making the threshold selection task ulteriorly complex. This is the case of the PCMAC and USPST(B) data set. In the MNIST data, the elements of kernel matrix non belonging to the main diagonal are very small due to the high degree of the polynomial kernel, so that the gradient and the preconditioned gradient are close.

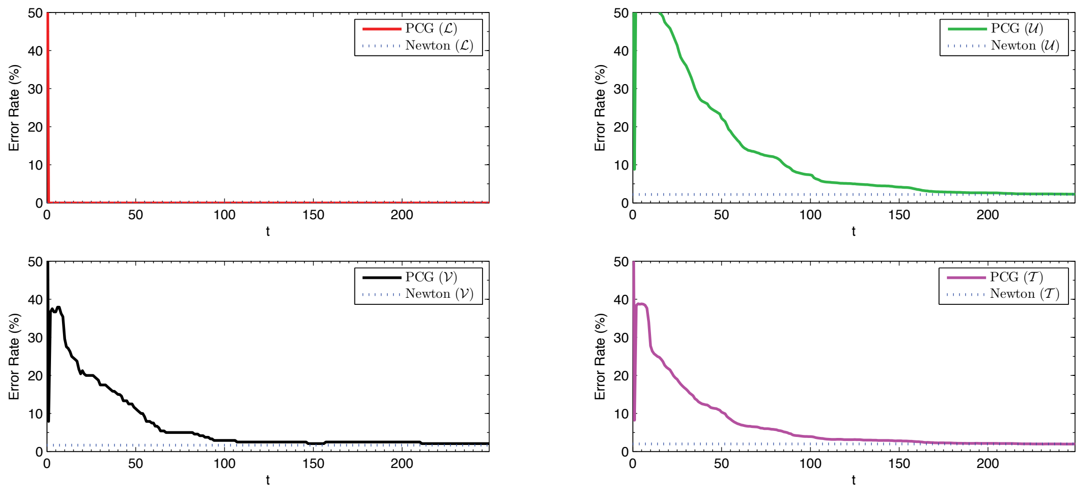


Figure 6: MNIST3VS8 data set: error rate on $\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$ of the Laplacian SVM classifier trained in the primal by preconditioned conjugate gradient (PCG), with respect to the number of gradient steps t. The error rate of the primal solution computed by means of Newton's method is reported as a horizontal line.

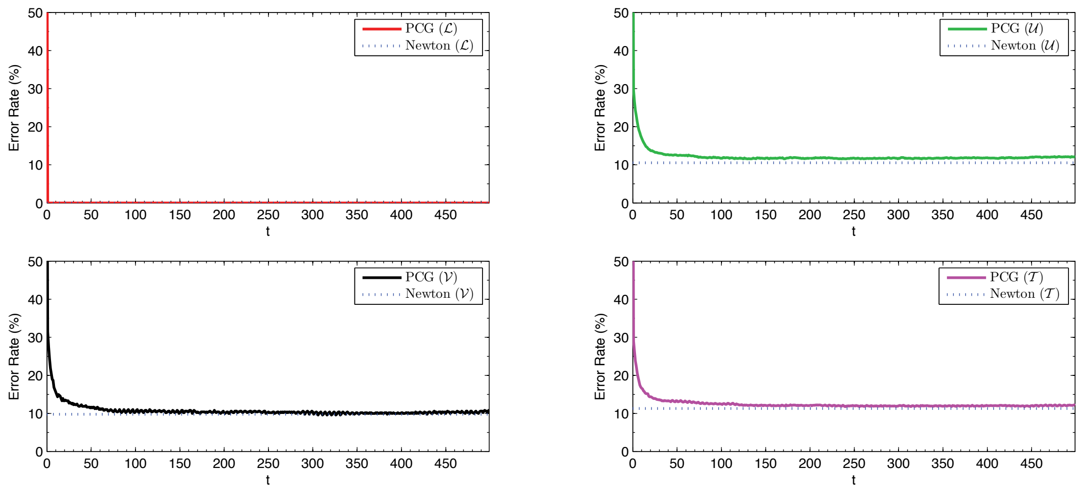


Figure 7: COIL20 data set: error rate on $\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$ of the Laplacian SVM classifier trained in the primal by preconditioned conjugate gradient (PCG), with respect to the number of gradient steps t. The error rate of the primal solution computed by means of Newton's method is reported as a horizontal line.

Using the proposed PCG goal conditions (Section 4), we cross-validated the primal LapSVM classifier trained by PCG, and the selected parameters are reported in Table 10 of Appendix A. In the USPST(B), COIL20(B), and MNIST3VS8 data sets, larger values for γ_A or γ_I are selected by the validation process, since the convergence speed of PCG is enhanced. In the other data sets, parameter values remain substantially the same of the ones selected by solving with the Newton's

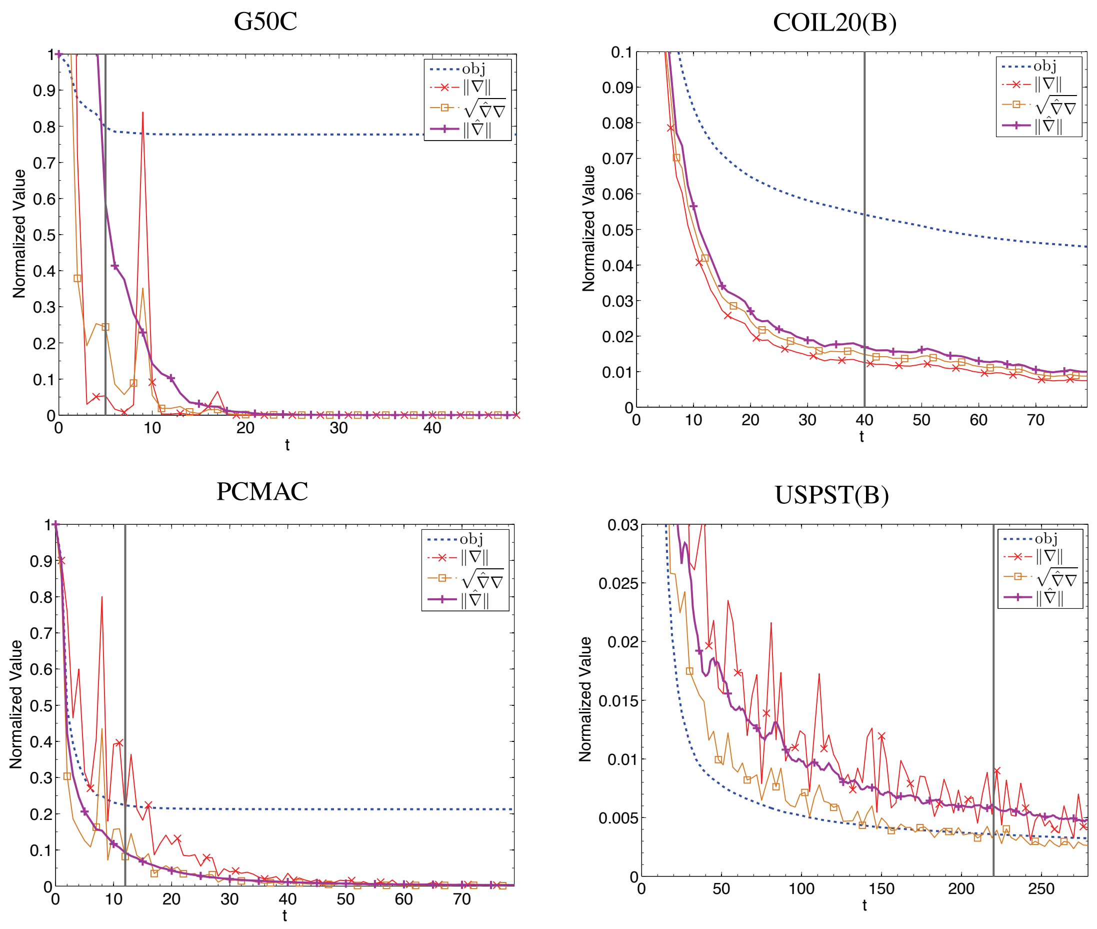


Figure 8: Details of each PCG iteration. The value of the objective function obj, of the gradient norm $\|\nabla\|$, of the preconditioned gradient norm $\|\hat{\nabla}\|$, and of the mixed product $\sqrt{\hat{\nabla}^T\nabla}$ are displayed in function of the number of PCG iterations (t). The vertical line represents the number of iterations after which the error rate on all partitions ($\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$) is roughly the same to the one at the stationary point.

method, suggesting that a reliable and fast cross-validation can be performed with PCG and the proposed early stopping heuristics.

In Table 6 the training times, the number of PCG and line search iterations are collected, whereas in Table 7 the corresponding classification error rates are reported, for a comparison with the non-approximated solution computed using Newton's method. As already stressed, the training times appreciably drop down when training a LapSVM in the primal using PCG and our goal conditions, independently by the data set. Early stopping allows us to obtain results comparable to the Newton's method or to the original two step dual formulation, showing a direct correlation between the proposed goal conditions and the quality of the classifier. Moreover, our conditions are the same for each problem or data set, overcoming all the issues of the previously described ones. In the COIL20 data set we can observe performances less close to the one of the solution computed with Newton's method. This is due to the already addressed motivations, and it also suggests that the stopping

Data Set	**Laplacian SVM**	**Training Time**	**PCG Iters**	**LS Iters**
	Dual	0.155 (0.004)	-	-
	Newton	0.134 (0.006)	-	-
G50C	PCG [Stability Check]	**0.044** (0.006)	20 (0)	1 (0)
	PCG [Validation Check]	**0.043** (0.006)	20.83 (2.89)	1 (0)
	PCG [Mixed Check]	**0.044** (0.006)	20.83 (2.89)	1 (0)
	Dual	0.311 (0.012)	-	-
	Newton	0.367 (0.097)	-	-
COIL20(B)	PCG [Stability Check]	**0.198** (0.074)	74.67 (28.4)	2.41 (1.83)
	PCG [Validation Check]	**0.097** (0.026)	37.33 (10.42)	1 (0)
	PCG [Mixed Check]	**0.206** (0.089)	78.67 (34.42)	2.38 (1.79)
	Dual	14.8203 (0.104)	-	-
	Newton	15.756 (0.285)	-	-
PCMAC	PCG [Stability Check]	**1.897** (0.040)	38.00 (0)	1.16 (0.45)
	PCG [Validation Check]	**1.967** (0.269)	39.58 (5.48)	1.15 (0.44)
	PCG [Mixed Check]	**1.997** (0.258)	39.58 (5.48)	1.15 (0.44)
	Dual	1.196 (0.015)	-	-
	Newton	1.4727 (0.2033)	-	-
USPST(B)	PCG [Stability Check]	**0.300** (0.030)	58.58 (5.48)	1.74 (0.90)
	PCG [Validation Check]	**0.281** (0.086)	55.42 (17.11)	1.68 (0.90)
	PCG [Mixed Check]	**0.324** (0.059)	63.33 (12.38)	1.70 (0.89)
	Dual	6.321 (0.441)	-	-
	Newton	7.26 (1.921)	-	-
COIL20	PCG [Stability Check]	**3.297** (1.471)	65.47 (30.35)	2.53 (1.90)
	PCG [Validation Check]	**1.769** (0.299)	34.07 (6.12)	3.37 (2.22)
	PCG [Mixed Check]	**3.487** (1.734)	69.53 (35.86)	2.48 (1.87)
	Dual	12.25 (0.2)	-	-
	Newton	17.74 (2.44)	-	-
USPST	PCG [Stability Check]	**1.953** (0.403)	41.17 (8.65)	3.11 (1.73)
	PCG [Validation Check]	**2.032** (0.434)	42.91 (9.38)	3.13 (1.73)
	PCG [Mixed Check]	**2.158** (0.535)	45.60 (11.66)	3.12 (1.72)
	Dual	2064.18 (3.1)	-	-
	Newton	2824.174 (105.07)	-	-
MNIST3VS8	PCG [Stability Check]	**114.441** (0.235)	110 (0)	5.58 (2.79)
	PCG [Validation Check]	**124.69** (0.335)	110 (0)	5.58 (2.79)
	PCG [Mixed Check]	**124.974** (0.414)	110 (0)	5.58 (2.79)
	PCG [Stability Check]	**35.728** (0.868)	3 (0)	1 (0)
FACEMIT	PCG [Validation Check]	**35.728** (0.868)	3 (0)	1 (0)
	PCG [Mixed Check]	**35.728** (0.868)	3 (0)	1 (0)

Table 6: Training time comparison among the Laplacian SVMs trained in the dual (Dual), LapSVM trained in the primal by means of Newton's method (Newton) and by means of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). Average training times (in seconds) and their standard deviations, the number of PCG iterations, and of Line Search (LS) iterations (per each PCG one) are reported.

Data Set	Laplacian SVM	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{T}$
G50C	Newton	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
	PCG [Stability Check]	6.13 (1.46)	6.17 (3.46)	7.27 (2.87)
	PCG [Validation Check]	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
	PCG [Mixed Check]	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
COIL20(B)	Newton	8.16 (2.04)	7.92 (3.96)	8.56 (1.9)
	PCG [Stability Check]	8.81 (2.23)	8.13 (3.71)	8.84 (1.93)
	PCG [Validation Check]	8.32 (2.28)	8.96 (4.05)	8.45 (1.58)
	PCG [Mixed Check]	8.84 (2.28)	8.13 (3.71)	8.84 (1.96)
PCMAC	Newton	9.68 (0.77)	7.83 (4.04)	9.37 (1.51)
	PCG [Stability Check]	9.65 (0.78)	7.83 (4.04)	9.42 (1.50)
	PCG [Validation Check]	9.67 (0.76)	7.83 (4.04)	9.40 (1.50)
	PCG [Mixed Check]	9.67 (0.76)	7.83 (4.04)	9.40 (1.50)
USPST(B)	Newton	8.72 (2.15)	9.33 (3.85)	9.42 (2.34)
	PCG [Stability Check]	9.11 (2.14)	10.50 (4.36)	9.70 (2.55)
	PCG [Validation Check]	9.10 (2.17)	10.50 (4.36)	9.75 (2.59)
	PCG [Mixed Check]	9.09 (2.17)	10.50 (4.36)	9.70 (2.55)
COIL20	Newton	10.54 (2.03)	9.79 (4.94)	11.32 (2.19)
	PCG [Stability Check]	12.42 (2.68)	10.63 (4.66)	12.92 (2.14)
	PCG [Validation Check]	13.07 (2.73)	12.08 (4.75)	13.52 (2.12)
	PCG [Mixed Check]	12.43 (2.69)	10.42 (4.63)	12.87 (2.20)
USPST	Newton	14.98 (2.88)	15 (3.57)	15.38 (3.55)
	PCG [Stability Check]	15.60 (3.45)	15.67 (3.60)	16.11 (3.95)
	PCG [Validation Check]	15.40 (3.38)	15.67 (3.98)	15.94 (4.04)
	PCG [Mixed Check]	15.45 (3.53)	15.50 (3.92)	15.94 (4.08)
MNIST3VS8	Newton	2.2 (0.14)	1.67 (1.44)	2.02 (0.22)
	PCG [Stability Check]	2.11 (0.06)	1.67 (1.44)	1.93 (0.2)
	PCG [Validation Check]	2.11 (0.06)	1.67 (1.44)	1.93 (0.2)
	PCG [Mixed Check]	2.11 (0.06)	1.67 (1.44)	1.93 (0.2)
FACEMIT	PCG [Stability Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)
	PCG [Validation Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)
	PCG [Mixed Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)

Table 7: Average classification error (standard deviation is reported brackets) of Laplacian SVMs trained in the primal by means of Newton's method (Newton) and of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). $\mathcal{U}$ is the set of unlabeled examples used to train the classifiers. $\mathcal{V}$ is the labeled set for cross-validating parameters whereas $\mathcal{T}$ is the out-of-sample test set. Results on the labeled training set $\mathcal{L}$ are omitted since all algorithms correctly classify such a few labeled training points.

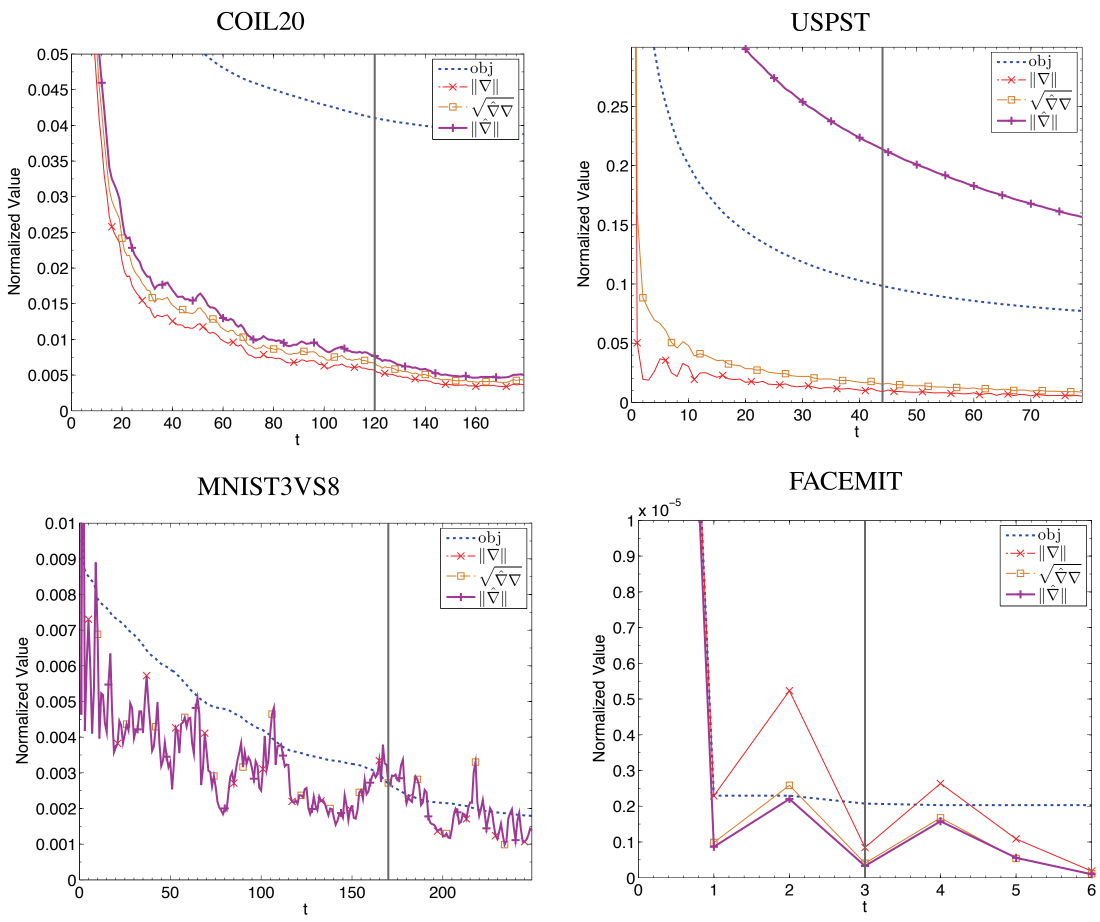


Figure 9: Details of each PCG iteration. The value of the objective function obj, of the gradient norm $\|\nabla\|$, of the preconditioned gradient norm $\|\hat{\nabla}\|$, and of the mixed product $\sqrt{\hat{\nabla}^T\nabla}$ are displayed in function of the number of PCG iterations (t). The vertical line represents the number of iterations after which the error rate on all partitions ($\mathcal{L}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{T}$) is roughly the same to the one at the stationary point.

condition should probably be checked while training in parallel the 20 binary classifiers, instead of separately checking it on each of them. A better tuning of the goal conditions or a different formulation of them can move the accuracy closer to the one of primal LapSVM trained with Newton's method, but it goes beyond to the scope of this paper.

The number of PCG iterations is noticeably smaller than n. Obviously it is function of the gap between each checking of a stopping criterion, that we set to $\sqrt{n}/2$. The number of iterations from the stability check is sometimes larger that the one from the validation check (COIL20(B), USPST, COIL20). As a matter of fact, labeled validation data is more informative than a stable, but unknown, decision on the unlabeled one. On the other hand validation data could not represent test data enough accurately. Using a mixed strategy makes sense in those cases, as can be observed in the COIL20 data set. In our experiments the mixed criterion has generally the same behavior of the most strict of the two heuristics for each specific set of data. In the FACEMIT data set complete

Data Set	**Laplacian RLSC**	**Training Time**	**PCG Iters**	$\mathcal{T}$
PCMAC	Matrix Inversion	14.21 (0.067)	-	9.34 (1.5)
	PCG [Stability Check]	**1.818** (0.016)	38 (0)	9.34 (1.46)
	PCG [Validation Check]	**1.82** (0.05)	38 (0)	9.34 (1.46)
	PCG [Mixed Check]	**1.821** (0.047)	38 (0)	9.34 (1.46)

Table 8: Training time comparison among the Laplacian RLSCs trained by solving Equation 14 with matrix inversion and by means of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). Average training times (in seconds), the number of PCG iterations, and the average classification error on test data $\mathcal{T}$ are shown. Standard deviations are reported brackets.

convergence is achieved in just a few iterations, independently by the heuristics. The number of line search iterations is usually very small and negligible with respect to the computational cost of the training algorithm.

For the sake of completeness, we show an example of the application of our early stopped PCG to LapRLSC, as described in Section 5. In Table 8 we report the training times, the PCG iterations, and the error rate (on test points) in the case of PCMAC data. The reduction of training times is significant, and positively influenced by the non iterative line search procedure.

7. Conclusions and Future Work

In this paper we described investigated in detail two strategies for solving the optimization problem of Laplacian Support Vector Machines (LapSVMs) in the primal. A very fast solution can be achieved using preconditioned conjugate gradient coupled with an early stopping criterion based on the stability of the classifier decision. Detailed experimental results on real world data show the validity of such strategy. The computational cost for solving the problem reduces from $O(n^3)$ to $O(kn^2)$, where n is the total number of training points, both labeled and unlabeled, and k is empirically evaluated to be significantly smaller than n, without the need of storing in memory the Hessian matrix and its inverse. Training times are significantly reduced on all selected benchmarks, in particular, as the amount of training data increases. This solution can be a useful starting point for applying greedy techniques for incremental classifier building or for studying the effects of a sparser kernel expansion of the classification function. Moreover, some recently proposed domain decomposition techniques for large scale RLSC (Li et al., 2007) could be investigated to solve the primal LapSVM problem, that we will address in future work.

Acknowledgments

We would like to acknowledge the NSF for supporting this research.

Appendix A.

This Appendix collects all the parameters selected using our experimental protocol, for reproducibility of the experiments (Table 9 and Table 10). Details of the cross-validation procedure are described in Section 6.

In the most of the data sets, parameter values selected using the PCG solution remain substantially the same of the ones selected by solving the primal problem with the Newton's method, suggesting that a reliable and fast cross-validation can be performed with PCG and the proposed early stopping heuristics. In the USPST(B), COIL20(B), and MNIST3VS8 data sets, larger values for γ_A or γ_I are selected when using PCG, since the convergence speed of gradient descent is enhanced.

To emphasize this behavior, the training times and the resulting error rates of the PCG solution computed using γ_A and γ_I tuned by means of the Newton's method (instead of the ones computed by PCG with each specific goal condition) are reported in Table 11 and in Table 12. Comparing these results with the ones presented in Section 6, it can be appreciated that both the convergence speed (Table 6) and the accuracy of the PCG solution (Table 7) benefit from an appropriate parameter selection. Note that the performance gaps between Newton's method and PCG of a given data set sometimes are slightly different among $\mathcal{U}$, $\mathcal{V}$, and $\mathcal{T}$. As a matter of fact, the balancing of class labels may not be exactly the same among the three sets, due to the random sampling of $\mathcal{V}$ (and $\mathcal{L}$) from non-test data, as described in Section 6.

References

J. Abernethy, O. Chapelle, and C. Castillo. Witch: A new approach to web spam detection. *Technical Report 2008-001, Yahoo! Research*, 2008.

M. Belkin and P. Niyogi. Using manifold stucture for partially labeled classification. *Advances in Neural Information Processing Systems*, pages 953–960, 2003.

M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7: 2399–2434, 2006.

S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 585–592. Cambridge, MA, USA: MIT Press, 2003.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. MIT press, 2006.

O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9:203–233, 2008.

Data Set	**Classifier**	σ	nn	p	γ_A	γ_I
	SVM	17.5	-	-	10^{-1}	-
	RLSC	17.5	-	-	1	-
G50C	LapRLSC	17.5	50	5	10^{-6}	10^{-2}
	LapSVM Dual (Original)	17.5	50	5	1	10
	LapSVM Primal (Newton)	17.5	50	5	10^{-1}	10
	SVM	0.6	-	-	10^{-6}	-
	RLSC	0.6	-	-	10^{-6}	-
COIL20(B)	LapRLSC	0.6	2	1	10^{-6}	1
	LapSVM Dual (Original)	0.6	2	1	10^{-2}	100
	LapSVM Primal (Newton)	0.6	2	1	10^{-6}	1
	SVM	2.7	-	-	10^{-6}	-
	RLSC	2.7	-	-	10^{-6}	-
PCMAC	LapRLSC	2.7	50	5	10^{-6}	10^{-2}
	LapSVM Dual (Original)	2.7	50	5	10^{-6}	10^{-4}
	LapSVM Primal (Newton)	2.7	50	5	10^{-6}	1
	SVM	9.4	-	-	10^{-6}	-
	RLSC	9.4	-	-	10^{-1}	-
USPST(B)	LapRLSC	9.4	10	2	10^{-4}	10^{-1}
	LapSVM Dual (Original)	9.4	10	2	10^{-6}	10^{-2}
	LapSVM Primal (Newton)	9.4	10	2	10^{-6}	10^{-2}
	SVM	0.6	-	-	10^{-6}	-
	RLSC	0.6	-	-	10^{-6}	-
COIL20	LapRLSC	0.6	2	1	10^{-6}	1
	LapSVM Dual (Original)	0.6	2	1	10^{-6}	10
	LapSVM Primal (Newton)	0.6	2	1	10^{-6}	1
	SVM	9.4	-	-	10^{-1}	-
	RLSC	9.4	-	-	10^{-6}	-
USPST	LapRLSC	9.4	10	2	10^{-6}	10^{-1}
	LapSVM Dual (Original)	9.4	10	2	10^{-6}	10^{-2}
	LapSVM Primal (Newton)	9.4	10	2	10^{-4}	1
	SVM	9	-	-	10^{-6}	-
	RLSC	9	-	-	10^{-6}	-
MNIST3VS8	LapRLSC	9	20	3	10^{-6}	10^{-2}
	LapSVM Dual (Original)	9	20	3	10^{-6}	10^{-2}
	LapSVM Primal (Newton)	9	20	3	10^{-6}	10^{-2}
	SVM	4.3	-	-	10^{-6}	-
FACEMIT	RLSC	4.3	-	-	10^{-6}	-
	LapSVM Primal (PCG)	4.3	6	1	10^{-6}	10^{-8}

Table 9: Parameters selected by cross-validation for supervised algorithms (SVM, RLSC) and semi-supervised ones based on manifold regularization, using different loss functions (LapRLSC, LapSVM trained in the dual formulation and in the primal one by means of Newton's method). The parameter σ is the bandwidth of the Gaussian kernel or, in the MNIST3VS8, the degree of the polynomial one.

Data Set	**Laplacian SVM**	γ_A	γ_I
G50C	Newton	10^{-1}	10
	PCG [Stability Check]	10^{-1}	10
	PCG [Validation Check]	10^{-1}	10
	PCG [Mixed Check]	10^{-1}	10
COIL20(B)	Newton	10^{-6}	1
	PCG [Stability Check]	10^{-6}	1
	PCG [Validation Check]	1	100
	PCG [Mixed Check]	10^{-6}	1
PCMAC	Newton	10^{-6}	1
	PCG [Stability Check]	10^{-4}	1
	PCG [Validation Check]	10^{-4}	1
	PCG [Mixed Check]	10^{-6}	10^{-1}
USPST(B)	Newton	10^{-6}	10^{-2}
	PCG [Stability Check]	10^{-6}	1
	PCG [Validation Check]	10^{-6}	1
	PCG [Mixed Check]	10^{-6}	1
COIL20	Newton	10^{-6}	1
	PCG [Stability Check]	10^{-6}	1
	PCG [Validation Check]	10^{-6}	1
	PCG [Mixed Check]	10^{-6}	1
USPST	Newton	10^{-4}	1
	PCG [Stability Check]	10^{-4}	1
	PCG [Validation Check]	10^{-4}	1
	PCG [Mixed Check]	10^{-4}	1
MNIST3VS8	Newton	10^{-6}	10^{-2}
	PCG [Stability Check]	10^{-6}	10^{-1}
	PCG [Validation Check]	10^{-6}	10^{-1}
	PCG [Mixed Check]	10^{-6}	10^{-1}
FACEMIT	PCG [Stability Check]	10^{-6}	10^{-8}
	PCG [Validation Check]	10^{-6}	10^{-8}
	PCG [Mixed Check]	10^{-6}	10^{-8}

Table 10: A comparison of the parameters selected by cross-validation for Laplacian SVMs trained in the primal by means of Newton's method (Newton) and preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets).

Data Set	Laplacian SVM	Training Time	PCG Iters	LS Iters
	Dual	0.155 (0.004)	-	-
	Newton	0.134 (0.006)	-	-
G50C	PCG [Stability Check]	0.044 (0.006)	20 (0)	1 (0)
	PCG [Validation Check]	0.043 (0.006)	20.83 (2.89)	1 (0)
	PCG [Mixed Check]	0.044 (0.006)	20.83 (2.89)	1 (0)
	Dual	0.311 (0.012)	-	-
	Newton	0.367 (0.097)	-	-
COIL20(B)	PCG [Stability Check]	0.198 (0.074)	74.67 (28.4)	2.41 (1.83)
	PCG [Validation Check]	0.095 (0.018)	36 (7.24)	3.26 (2.21)
	PCG [Mixed Check]	0.206 (0.089)	78.67 (34.42)	2.38 (1.79)
	Dual	14.8203 (0.104)	-	-
	Newton	15.756 (0.285)	-	-
PCMAC	PCG [Stability Check]	1.901 (0.022)	38.00 (0)	1.18 (0.45)
	PCG [Validation Check]	1.970 (0.265)	39.58 (5.48)	1.18 (0.44)
	PCG [Mixed Check]	1.969 (0.268)	39.58 (5.48)	1.18 (0.44)
	Dual	1.196 (0.015)	-	-
	Newton	1.4727 (0.2033)	-	-
USPST(B)	PCG [Stability Check]	0.496 (0.172)	95.00 (33.40)	6.56 (3.18)
	PCG [Validation Check]	0.279 (0.096)	52.25 (18.34)	6.83 (3.44)
	PCG [Mixed Check]	0.567 (0.226)	107.67 (43.88)	6.49 (3.15)
	Dual	6.321 (0.441)	-	-
	Newton	7.26 (1.921)	-	-
COIL20	PCG [Stability Check]	3.297 (1.471)	65.47 (30.35)	2.53 (1.90)
	PCG [Validation Check]	1.769 (0.299)	34.07 (6.12)	3.37 (2.22)
	PCG [Mixed Check]	3.487 (1.734)	69.53 (35.86)	2.48 (1.87)
	Dual	12.25 (0.2)	-	-
	Newton	17.74 (2.44)	-	-
USPST	PCG [Stability Check]	1.953 (0.403)	41.17 (8.65)	3.11 (1.73)
	PCG [Validation Check]	2.032 (0.434)	42.91 (9.38)	3.13 (1.73)
	PCG [Mixed Check]	2.158 (0.535)	45.60 (11.66)	3.12 (1.72)
	Dual	2064.18 (3.1)	-	-
	Newton	2824.174 (105.07)	-	-
MNIST3VS8	PCG [Stability Check]	188.775 (0.237)	165 (0)	6.78 (3.65)
	PCG [Validation Check]	207.986 (35.330)	183.33 (31.75)	6.65 (3.57)
	PCG [Mixed Check]	207.915 (35.438)	183.33 (31.75)	6.65 (3.57)
	PCG [Stability Check]	35.728 (0.868)	3 (0)	1 (0)
FACEMIT	PCG [Validation Check]	35.728 (0.868)	3 (0)	1 (0)
	PCG [Mixed Check]	35.728 (0.868)	3 (0)	1 (0)

Table 11: Training time comparison among the Laplacian SVMs trained in the dual (Dual), LapSVM trained in the primal by means of Newton's method (Newton) and by means of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). *Parameters of the classifiers were tuned using the Newton's method.* Average training times (in seconds) and their standard deviations, the number of PCG iterations, and of Line Search (LS) iterations (per each PCG one) are reported.

Data Set	Laplacian SVM	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{T}$
G50C	Newton	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
	PCG [Stability Check]	6.13 (1.46)	6.17 (3.46)	7.27 (2.87)
	PCG [Validation Check]	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
	PCG [Mixed Check]	6.16 (1.48)	6.17 (3.46)	7.27 (2.87)
COIL20(B)	Newton	8.16 (2.04)	7.92 (3.96)	8.56 (1.9)
	PCG [Stability Check]	8.81 (2.23)	8.13 (3.71)	8.84 (1.93)
	PCG [Validation Check]	8.97 (2.32)	9.17 (3.74)	8.96 (1.64)
	PCG [Mixed Check]	8.84 (2.28)	8.13 (3.71)	8.84 (1.96)
PCMAC	Newton	9.68 (0.77)	7.83 (4.04)	9.37 (1.51)
	PCG [Stability Check]	9.65 (0.76)	7.83 (4.04)	9.42 (1.43)
	PCG [Validation Check]	9.65 (0.76)	7.83 (4.04)	9.40 (1.43)
	PCG [Mixed Check]	9.65 (0.76)	7.83 (4.04)	9.40 (1.43)
USPST(B)	Newton	8.72 (2.15)	9.33 (3.85)	9.42 (2.34)
	PCG [Stability Check]	11.07 (2.27)	13.33 (4.21)	11.49 (2.55)
	PCG [Validation Check]	12.02 (2.22)	14.67 (2.99)	12.01 (2.14)
	PCG [Mixed Check]	10.81 (2.39)	12.83 (4.78)	11.31 (2.71)
COIL20	Newton	10.54 (2.03)	9.79 (4.94)	11.32 (2.19)
	PCG [Stability Check]	12.42 (2.68)	10.63 (4.66)	12.92 (2.14)
	PCG [Validation Check]	13.07 (2.73)	12.08 (4.75)	13.52 (2.12)
	PCG [Mixed Check]	12.43 (2.69)	10.42 (4.63)	12.87 (2.20)
USPST	Newton	14.98 (2.88)	15 (3.57)	15.38 (3.55)
	PCG [Stability Check]	15.60 (3.45)	15.67 (3.60)	16.11 (3.95)
	PCG [Validation Check]	15.40 (3.38)	15.67 (3.98)	15.94 (4.04)
	PCG [Mixed Check]	15.45 (3.53)	15.50 (3.92)	15.94 (4.08)
MNIST3VS8	Newton	2.2 (0.14)	1.67 (1.44)	2.02 (0.22)
	PCG [Stability Check]	3.16 (0.15)	2.5 (1.25)	2.4 (0.38)
	PCG [Validation Check]	2.89 (0.62)	2.50 (1.25)	2.37 (0.44)
	PCG [Mixed Check]	2.89 (0.62)	2.5 (1.25)	2.37 (0.44)
FACEMIT	PCG [Stability Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)
	PCG [Validation Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)
	PCG [Mixed Check]	29.97 (2.51)	36 (3.46)	27.97 (5.38)

Table 12: Average classification error (standard deviation is reported brackets) of Laplacian SVMs trained in the primal by means of Newton's method and of preconditioned conjugate gradient (PCG) with the proposed early stopping conditions (in square brackets). *Parameters of the classifiers were tuned using the Newton's method.* $\mathcal{U}$ is the set of unlabeled examples used to train the classifiers. $\mathcal{V}$ is the labeled set for cross-validating parameters whereas $\mathcal{T}$ is the out-of-sample test set. Results on the labeled training set $\mathcal{L}$ are omitted since all classifiers perfectly fit such few labeled training points.

D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46 (1):161–190, 2002.

A. Demiriz and K. Bennett. Optimization approaches to semi-supervised learning. *Complementarity: Applications, Algorithms and Extensions*, 50:1–19, 2000.

R.E. Fan, P.H. Chen, and C.J. Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 200–209, 1999.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning*, volume 20, pages 290–297, 2003.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226. ACM New York, NY, USA, 2006.

S.S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *The Journal of Machine Learning Research*, 6(1):341–361, 2005.

S.S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *The Journal of Machine Learning Research*, 7:1493–1515, 2006.

W. Li, K.-H. Lee, and K.-S. Leung. Large-scale RLSC learning without agony. In *Proceedings of the 24th International Conference on Machine learning*, pages 529–536, New York, NY, USA, 2007. ACM.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton methods for large-scale logistic regression. *The Journal of Machine Learning Research*, 9:627–650, 2008.

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

M. Seeger. Low rank updates for the Cholesky decomposition. *Department of EECS, University of California at Berkeley, Technical Report*, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the International Conference on Machine Learning*, pages 807–814, 2007.

J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. *School of Computer Science, Carnegie Mellon University, Techical Report*, 1994.

V. Sindhwani. *On Semi-supervised Kernel Methods*. PhD thesis, University of Chicago, 2007.

V. Sindhwani and D.S. Rosenberg. An RKHS for multi-view learning and manifold co-regularization. In *Proceedings of the International Conference on Machine Learning*, pages 976–983, 2008.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: From transductive to semi-supervised learning. In *Proceedings of the International Conference on Machine Learning*, volume 22, pages 825–832, 2005.

I.W. Tsang and J.T. Kwok. Large-scale sparsified manifold regularization. *Advances in Neural Information Processing Systems*, 19:1401–1408, 2006.

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

X. Zhu and A.B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan and Claypool, 2009.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, 2003.

Journal of Machine Learning Research 12 (2011) 1185-1224 Submitted 3/10; Revised 3/11; Published 4/11

The Indian Buffet Process: An Introduction and Review

Thomas L. Griffiths TOM_GRIFFITHS@BERKELEY.EDU
Department of Psychology
University of California, Berkeley
Berkeley, CA 94720-1650, USA

Zoubin Ghahramani[*] ZOUBIN@ENG.CAM.AC.UK
Department of Engineering
University of Cambridge
Cambridge CB2 1PZ, UK

Editor: David M. Blei

Abstract

The Indian buffet process is a stochastic process defining a probability distribution over equivalence classes of sparse binary matrices with a finite number of rows and an unbounded number of columns. This distribution is suitable for use as a prior in probabilistic models that represent objects using a potentially infinite array of features, or that involve bipartite graphs in which the size of at least one class of nodes is unknown. We give a detailed derivation of this distribution, and illustrate its use as a prior in an infinite latent feature model. We then review recent applications of the Indian buffet process in machine learning, discuss its extensions, and summarize its connections to other stochastic processes.

Keywords: nonparametric Bayes, Markov chain Monte Carlo, latent variable models, Chinese restaurant processes, beta process, exchangeable distributions, sparse binary matrices

1. Introduction

Unsupervised learning aims to recover the latent structure responsible for generating observed data. One of the key problems faced by unsupervised learning algorithms is thus determining the amount of latent structure—the number of clusters, dimensions, or variables—needed to account for the regularities expressed in the data. Often, this is treated as a model selection problem, choosing the model with the dimensionality that results in the best performance. This treatment of the problem assumes that there is a single, finite-dimensional representation that correctly characterizes the properties of the observed objects. An alternative is to assume that the amount of latent structure is actually potentially unbounded, and that the observed objects only manifest a sparse subset of those classes or features (Rasmussen and Ghahramani, 2001).

The assumption that the observed data manifest a subset of an unbounded amount of latent structure is often used in nonparametric Bayesian statistics, and has recently become increasingly popular in machine learning. In particular, this assumption is made in Dirichlet process mixture models, which are used for nonparametric density estimation (Antoniak, 1974; Escobar and West, 1995; Ferguson, 1983; Neal, 2000). Under one interpretation of a Dirichlet process mixture model, each datapoint is assigned to a latent class, and each class is associated with a distribution over

*. Also at the Machine Learning Department, Carnegie Mellon University, Pittsburgh PA 15213, USA.

observable properties. The prior distribution over assignments of datapoints to classes is specified in such a way that the number of classes used by the model is bounded only by the number of objects, making Dirichlet process mixture models "infinite" mixture models (Rasmussen, 2000).

Recent work has extended Dirichlet process mixture models in a number of directions, making it possible to use nonparametric Bayesian methods to discover the kinds of structure common in machine learning: hierarchies (Blei et al., 2004; Heller and Ghahramani, 2005; Neal, 2003; Teh et al., 2008), topics and syntactic classes (Teh et al., 2004) and the objects appearing in images (Sudderth et al., 2006). However, the fact that all of these models are based upon the Dirichlet process limits the kinds of latent structure that they can express. In many of these models, each object described in a data set is associated with a latent variable that picks out a single class or parameter responsible for generating that datapoint. In contrast, many models used in unsupervised learning represent each object as having multiple features or being produced by multiple causes. For instance, we could choose to represent each object with a binary vector, with entries indicating the presence or absence of each feature (e.g., Ueda and Saito, 2003), allow each feature to take on a continuous value, representing datapoints with locations in a latent space (e.g., Jolliffe, 1986), or define a factorial model, in which each feature takes on one of a discrete set of values (e.g., Zemel and Hinton, 1994; Ghahramani, 1995). Infinite versions of these models are difficult to define using the Dirichlet process.

In this paper, we summarize recent work exploring the extension of this nonparametric approach to models in which objects are represented using an unknown number of latent features. Following Griffiths and Ghahramani (2005, 2006), we provide a detailed derivation of a distribution that can be used to define probabilistic models that represent objects with infinitely many binary features, and can be combined with priors on feature values to produce factorial and continuous representations. This distribution can be specified in terms of a simple stochastic process called the *Indian buffet process*, by analogy to the *Chinese restaurant process* used in Dirichlet process mixture models. We illustrate how the Indian buffet process can be used to specify prior distributions in latent feature models, using a simple linear-Gaussian model to show how such models can be defined and used.

The Indian buffet process can also be used to define a prior distribution in any setting where the latent structure expressed in data can be expressed in the form of a binary matrix with a finite number of rows and infinite number of columns, such as the adjacency matrix of a bipartite graph where one class of nodes is of unknown size, or the adjacency matrix for a Markov process with an unbounded set of states. As a consequence, this approach has found a number of recent applications within machine learning. We review these applications, summarizing some of the innovations that have been introduced in order to use the Indian buffet process in different settings, as well as extensions to the basic model and alternative inference algorithms. We also describe some of the interesting connections to other stochastic processes that have been identified. As for the Chinese restaurant process, we can arrive at the Indian buffet process in a number of different ways: as the infinite limit of a finite model, via the constructive specification of an infinite model, or by marginalizing out an underlying measure. Each perspective provides different intuitions, and suggests different avenues for designing inference algorithms and generalizations.

The plan of the paper is as follows. Section 2 summarizes the principles behind infinite mixture models, focusing on the prior on class assignments assumed in these models, which can be defined in terms of a simple stochastic process—the Chinese restaurant process. We then develop a distribution on infinite binary matrices by considering how this approach can be extended to the case where objects are represented with multiple binary features. Section 3 discusses the role of a such a distribution in defining infinite latent feature models. Section 4 derives the distribution, making use of the Indian buffet process. Section 5 illustrates how this distribution can be used as a prior in a nonparametric Bayesian model, defining an infinite-dimensional linear-Gaussian model, deriving a sampling algorithm for inference in this model, and applying it to two simple data sets. Section 6 describes further applications of this approach, both in latent feature models and for inferring graph structures, and Section 7 discusses recent work extending the Indian buffet process and providing connections to other stochastic processes. Section 8 presents conclusions and directions for future work.

2. Latent Class Models

Assume we have N objects, with the ith object having D observable properties represented by a row vector $\mathbf{x}_i$. In a latent class model, such as a mixture model, each object is assumed to belong to a single class, c_i, and the properties $\mathbf{x}_i$ are generated from a distribution determined by that class. Using the matrix $\mathbf{X} = \left[\mathbf{x}_1^T\, \mathbf{x}_2^T\, \cdots\, \mathbf{x}_N^T\right]^T$ to indicate the properties of all N objects, and the vector $\mathbf{c} = \left[c_1\; c_2\; \cdots\; c_N\right]^T$ to indicate their class assignments, the model is specified by a prior over assignment vectors $P(\mathbf{c})$, and a distribution over property matrices conditioned on those assignments, $p(\mathbf{X}|\mathbf{c})$.[1] These two distributions can be dealt with separately: $P(\mathbf{c})$ specifies the number of classes and their relative probability, while $p(\mathbf{X}|\mathbf{c})$ determines how these classes relate to the properties of objects. In this section, we will focus on the prior over assignment vectors, $P(\mathbf{c})$, showing how such a prior can be defined without placing an upper bound on the number of classes.

2.1 Finite Mixture Models

Mixture models assume that the assignment of an object to a class is independent of the assignments of all other objects. If there are K classes, we have

$$P(\mathbf{c}|\theta) = \prod_{i=1}^{N} P(c_i|\theta) = \prod_{i=1}^{N} \theta_{c_i},$$

where θ is a multinomial distribution over those classes, and θ_k is the probability of class k under that distribution. Under this assumption, the probability of the properties of all N objects $\mathbf{X}$ can be written as

$$p(\mathbf{X}|\theta) = \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_i|c_i = k)\, \theta_k. \tag{1}$$

The distribution from which each $\mathbf{x}_i$ is generated is thus a *mixture* of the K class distributions $p(\mathbf{x}_i|c_i = k)$, with θ_k determining the weight of class k.

The mixture weights θ can be treated as a parameter to be estimated. In Bayesian approaches to mixture modeling, θ is assumed to follow a prior distribution $p(\theta)$, with a standard choice being a symmetric Dirichlet distribution. The Dirichlet distribution on multinomials over K classes has parameters $\alpha_1, \alpha_2, \ldots, \alpha_K$, and is conjugate to the multinomial (e.g., Bernardo and Smith, 1994).

1. We will use $P(\cdot)$ to indicate probability mass functions, and $p(\cdot)$ to indicate probability density functions. We will assume that $\mathbf{x}_i \in \mathbb{R}^D$, and $p(\mathbf{X}|\mathbf{c})$ is thus a density, although variants of the models we discuss also exist for discrete data.

The probability density for the parameter θ of a multinomial distribution is given by

$$p(\theta) = \frac{\prod_{k=1}^{K} \theta_k^{\alpha_k - 1}}{D(\alpha_1, \alpha_2, \ldots, \alpha_K)},$$

in which $D(\alpha_1, \alpha_2, \ldots, \alpha_K)$ is the Dirichlet normalizing constant

$$\begin{aligned} D(\alpha_1, \alpha_2, \ldots, \alpha_K) &= \int_{\Delta_K} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \, d\theta \\ &= \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K} \alpha_k)}, \end{aligned} \tag{2}$$

where Δ_K is the simplex of multinomials over K classes, and $\Gamma(\cdot)$ is the gamma, or generalized factorial, function, with $\Gamma(m) = (m-1)!$ for any non-negative integer m. In a *symmetric* Dirichlet distribution, all α_k are equal. For example, we could take $\alpha_k = \frac{\alpha}{K}$ for all k. In this case, Equation 2 becomes

$$D(\tfrac{\alpha}{K}, \tfrac{\alpha}{K}, \ldots, \tfrac{\alpha}{K}) = \frac{\Gamma(\frac{\alpha}{K})^K}{\Gamma(\alpha)},$$

and the mean of θ is the multinomial that is uniform over all classes.

The probability model that we have defined is

$$\begin{aligned} \theta \,|\, \alpha &\sim \text{Dirichlet}(\tfrac{\alpha}{K}, \tfrac{\alpha}{K}, \ldots, \tfrac{\alpha}{K}), \\ c_i \,|\, \theta &\sim \text{Discrete}(\theta) \end{aligned}$$

where Discrete(θ) is the multiple-outcome analogue of a Bernoulli event, where the probabilities of the outcomes are specified by θ (i.e., $P(c_i = k|\theta) = \theta_k$). The dependencies among variables in this model are shown in Figure 1. Having defined a prior on θ, we can simplify this model by integrating over all values of θ rather than representing them explicitly. The marginal probability of an assignment vector $\mathbf{c}$, integrating over all values of θ, is

$$\begin{aligned} P(\mathbf{c}) &= \int_{\Delta_K} \prod_{i=1}^{n} P(c_i|\theta) \, p(\theta) \, d\theta \\ &= \int_{\Delta_K} \frac{\prod_{k=1}^{K} \theta_k^{m_k + \alpha/K - 1}}{D(\frac{\alpha}{K}, \frac{\alpha}{K}, \ldots, \frac{\alpha}{K})} \, d\theta \\ &= \frac{D(m_1 + \frac{\alpha}{K}, m_2 + \frac{\alpha}{K}, \ldots, m_k + \frac{\alpha}{K})}{D(\frac{\alpha}{K}, \frac{\alpha}{K}, \ldots, \frac{\alpha}{K})} \\ &= \frac{\prod_{k=1}^{K} \Gamma(m_k + \frac{\alpha}{K})}{\Gamma(\frac{\alpha}{K})^K} \frac{\Gamma(\alpha)}{\Gamma(N + \alpha)}, \end{aligned} \tag{3}$$

where $m_k = \sum_{i=1}^{N} \delta(c_i = k)$ is the number of objects assigned to class k. The tractability of this integral is a result of the fact that the Dirichlet is conjugate to the multinomial.

Equation 3 defines a joint probability distribution for all class assignments $\mathbf{c}$ in which individual class assignments are not independent. Rather, they are *exchangeable* (Bernardo and Smith, 1994), with the probability of an assignment vector remaining the same when the indices of the objects are permuted. Exchangeability is a desirable property in a distribution over class assignments, because

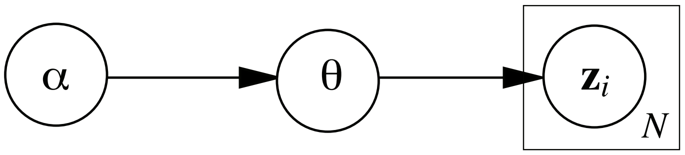


Figure 1: Graphical model for the Dirichlet-multinomial model used in defining the Chinese restaurant process. Nodes are variables, arrows indicate dependencies, and plates (Buntine, 1994) indicate replicated structures.

we have no special knowledge about the objects that would justify treating them differently from one another. However, the distribution on assignment vectors defined by Equation 3 assumes an upper bound on the number of classes of objects, since it only allows assignments of objects to up to K classes.

2.2 Infinite Mixture Models

Intuitively, defining an infinite mixture model means that we want to specify the probability of $\mathbf{X}$ in terms of infinitely many classes, modifying Equation 1 to become

$$p(\mathbf{X}|\theta) = \prod_{i=1}^{N} \sum_{k=1}^{\infty} p(\mathbf{x}_i|c_i = k)\,\theta_k,$$

where θ is an infinite-dimensional multinomial distribution. In order to repeat the argument above, we would need to define a prior, $p(\theta)$, on infinite-dimensional multinomials, and compute the probability of $\mathbf{c}$ by integrating over θ. This is essentially the strategy that is taken in deriving infinite mixture models from the Dirichlet process (Antoniak, 1974; Ferguson, 1983; Ishwaran and James, 2001; Sethuraman, 1994). Instead, we will work directly with the distribution over assignment vectors given in Equation 3, considering its limit as the number of classes approaches infinity (cf., Green and Richardson, 2001; Neal, 1992, 2000).

Expanding the gamma functions in Equation 3 using the recursion $\Gamma(x) = (x-1)\Gamma(x-1)$ and cancelling terms produces the following expression for the probability of an assignment vector $\mathbf{c}$:

$$P(\mathbf{c}) = \left(\frac{\alpha}{K}\right)^{K_+} \left(\prod_{k=1}^{K^+} \prod_{j=1}^{m_k-1} (j + \tfrac{\alpha}{K})\right) \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}, \tag{4}$$

where K_+ is the number of classes for which $m_k > 0$, and we have re-ordered the indices such that $m_k > 0$ for all $k \leq K_+$. There are K^N possible values for $\mathbf{c}$, which diverges as $K \to \infty$. As this happens, the probability of any single set of class assignments goes to 0. Since $K_+ \leq N$ and N is finite, it is clear that $P(\mathbf{c}) \to 0$ as $K \to \infty$, since $\frac{1}{K} \to 0$. Consequently, we will define a distribution over equivalence classes of assignment vectors, rather than the vectors themselves.

Specifically, we will define a distribution on *partitions* of objects. In our setting, a partition is a division of the set of N objects into subsets, where each object belongs to a single subset and the ordering of the subsets does not matter. Two assignment vectors that result in the same division of objects correspond to the same partition. For example, if we had three objects, the class assignments $\{c_1, c_2, c_3\} = \{1, 1, 2\}$ would correspond to the same partition as $\{2, 2, 1\}$, since all that differs between these two cases is the labels of the classes. A partition thus defines an equivalence class of assignment vectors, which we denote $[\mathbf{c}]$, with two assignment vectors belonging to the same equivalence class if they correspond to the same partition. A distribution over partitions is sufficient to allow us to define an infinite mixture model, provided the prior distribution on the parameters is the same for all classes. In this case, these equivalence classes of class assignments are the same as those induced by identifiability: $p(\mathbf{X}|\mathbf{c})$ is the same for all assignment vectors $\mathbf{c}$ that correspond to the same partition, so we can apply statistical inference at the level of partitions rather than the level of assignment vectors.

Assume we have a partition of N objects into K_+ subsets, and we have $K = K_0 + K_+$ class labels that can be applied to those subsets. Then there are $\frac{K!}{K_0!}$ assignment vectors $\mathbf{c}$ that belong to the equivalence class defined by that partition, $[\mathbf{c}]$. We can define a probability distribution over partitions by summing over all class assignments that belong to the equivalence class defined by each partition. The probability of each of those class assignments is equal under the distribution specified by Equation 4, so we obtain

$$\begin{aligned} P([\mathbf{c}]) &= \sum_{\mathbf{c}\in[\mathbf{c}]} P(\mathbf{c}) \\ &= \frac{K!}{K_0!}\left(\frac{\alpha}{K}\right)^{K_+}\left(\prod_{k=1}^{K^+}\prod_{j=1}^{m_k-1}(j+\tfrac{\alpha}{K})\right)\frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}. \end{aligned}$$

Rearranging the first two terms, we can compute the limit of the probability of a partition as $K \to \infty$, which is

$$\begin{aligned} \lim_{K\to\infty} \; & \alpha^{K_+} \cdot \frac{K!}{K_0!\,K^{K_+}} \cdot \left(\prod_{k=1}^{K_+}\prod_{j=1}^{m_k-1}(j+\tfrac{\alpha}{K})\right) \cdot \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)} \\ = \; & \alpha^{K_+} \cdot \quad 1 \quad \cdot \left(\prod_{k=1}^{K_+}(m_k-1)!\right) \cdot \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}. \end{aligned} \tag{5}$$

The details of the steps taken in computing this limit are given in Appendix A. These limiting probabilities define a valid distribution over partitions, and thus over equivalence classes of class assignments, providing a prior over class assignments for an infinite mixture model. Objects are exchangeable under this distribution, just as in the finite case: the probability of a partition is not affected by the ordering of the objects, since it depends only on the counts m_k.

As noted above, the distribution over partitions specified by Equation 5 can be derived in a variety of ways—by taking limits (Green and Richardson, 2001; Neal, 1992, 2000), from the Dirichlet process (Blackwell and MacQueen, 1973), or from other equivalent stochastic processes (Ishwaran and James, 2001; Sethuraman, 1994). We will briefly discuss a simple process that produces the same distribution over partitions: the Chinese restaurant process.

2.3 The Chinese Restaurant Process

The Chinese restaurant process (CRP) was named by Jim Pitman and Lester Dubins, based upon a metaphor in which the objects are customers in a restaurant, and the classes are the tables at which they sit (the process first appears in Aldous 1985, where it is attributed to Pitman, although

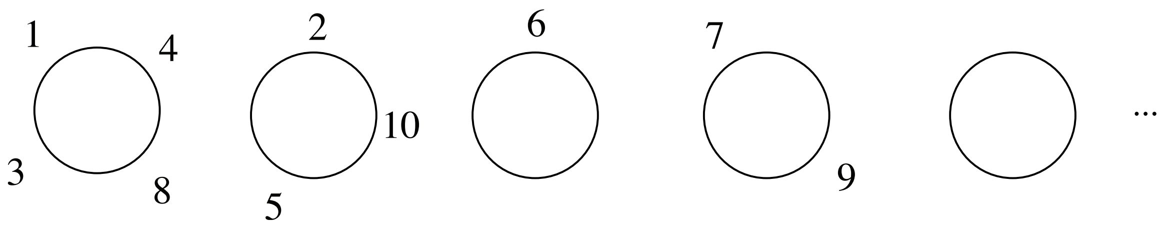


Figure 2: A partition induced by the Chinese restaurant process. Numbers indicate customers (objects), circles indicate tables (classes).

it is identical to the extended Polya urn scheme introduced by Blackwell and MacQueen 1973). Imagine a restaurant with an infinite number of tables, each with an infinite number of seats.[2] The customers enter the restaurant one after another, and each choose a table at random. In the CRP with parameter α, each customer chooses an occupied table with probability proportional to the number of occupants, and chooses the next vacant table with probability proportional to α. For example, Figure 2 shows the state of a restaurant after 10 customers have chosen tables using this procedure. The first customer chooses the first table with probability $\frac{\alpha}{\alpha} = 1$. The second customer chooses the first table with probability $\frac{1}{1+\alpha}$, and the second table with probability $\frac{\alpha}{1+\alpha}$. After the second customer chooses the second table, the third customer chooses the first table with probability $\frac{1}{2+\alpha}$, the second table with probability $\frac{1}{2+\alpha}$, and the third table with probability $\frac{\alpha}{2+\alpha}$. This process continues until all customers have seats, defining a distribution over allocations of people to tables, and, more generally, objects to classes. Extensions of the CRP and connections to other stochastic processes are pursued in depth by Pitman (2002).

The distribution over partitions induced by the CRP is the same as that given in Equation 5. If we assume an ordering on our N objects, then we can assign them to classes sequentially using the method specified by the CRP, letting objects play the role of customers and classes play the role of tables. The ith object would be assigned to the kth class with probability

$$P(c_i = k | c_1, c_2, \ldots, c_{i-1}) = \begin{cases} \frac{m_k}{i-1+\alpha} & k \leq K_+ \\ \frac{\alpha}{i-1+\alpha} & k = K+1 \end{cases}$$

where m_k is the number of objects currently assigned to class k, and K_+ is the number of classes for which $m_k > 0$. If all N objects are assigned to classes via this process, the probability of a partition of objects $\mathbf{c}$ is that given in Equation 5. The CRP thus provides an intuitive means of specifying a prior for infinite mixture models, as well as revealing that there is a simple sequential process by which exchangeable class assignments can be generated.

2.4 Inference by Gibbs Sampling

Inference in an infinite mixture model is only slightly more complicated than inference in a mixture model with a finite, fixed number of classes. The standard algorithm used for inference in infinite mixture models is Gibbs sampling (Bush and MacEachern, 1996; Neal, 2000). Gibbs sampling

2. Pitman and Dubins, both statisticians at the University of California, Berkeley, were inspired by the apparently infinite capacity of Chinese restaurants in San Francisco when they named the process.

is a Markov chain Monte Carlo (MCMC) method, in which variables are successively sampled from their distributions when conditioned on the current values of all other variables (Geman and Geman, 1984). This process defines a Markov chain, which ultimately converges to the distribution of interest (see Gilks et al., 1996). Recent work has also explored variational inference algorithms for these models (Blei and Jordan, 2006), a topic we will return to later in the paper.

Implementing a Gibbs sampler requires deriving the full conditional distribution for all variables to be sampled. In a mixture model, these variables are the class assignments $\mathbf{c}$. The relevant full conditional distribution is $P(c_i|\mathbf{c}_{-i},\mathbf{X})$, the probability distribution over c_i conditioned on the class assignments of all other objects, $\mathbf{c}_{-i}$, and the data, $\mathbf{X}$. By applying Bayes' rule, this distribution can be expressed as

$$P(c_i=k|\mathbf{c}_{-i},\mathbf{X}) \propto p(\mathbf{X}|\mathbf{c})P(c_i=k|\mathbf{c}_{-i}),$$

where only the second term on the right hand side depends upon the distribution over class assignments, $P(\mathbf{c})$. Here we assume that the parameters associated with each class can be integrated out, so we that the probability of the data depends only on the class assignment. This is possible when a conjugate prior is used on these parameters. For details, and alternative algorithms that can be used when this assumption is violated, see Neal (2000).

In a finite mixture model with $P(\mathbf{c})$ defined as in Equation 3, we can compute $P(c_i=k|\mathbf{c}_{-i})$ by integrating over θ, obtaining

$$\begin{aligned} P(c_i=k|\mathbf{c}_{-i}) &= \int P(c_i=k|\theta)p(\theta|\mathbf{c}_{-i})\,d\theta \\ &= \frac{m_{-i,k}+\frac{\alpha}{K}}{N-1+\alpha}, \end{aligned} \tag{6}$$

where $m_{-i,k}$ is the number of objects assigned to class k, not including object i. This is the posterior predictive distribution for a multinomial distribution with a Dirichlet prior.

In an infinite mixture model with a distribution over class assignments defined as in Equation 5, we can use exchangeability to find the full conditional distribution. Since it is exchangeable, $P([\mathbf{c}])$ is unaffected by the ordering of objects. Thus, we can choose an ordering in which the ith object is the last to be assigned to a class. It follows directly from the definition of the Chinese restaurant process that

$$P(c_i=k|\mathbf{c}_{-i}) = \begin{cases} \frac{m_{-i,k}}{N-1+\alpha} & m_{-i,k}>0 \\ \frac{\alpha}{N-1+\alpha} & k=K_{-i,+}+1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $K_{-i,+}$ is the number of classes for which $m_{-i,k}>0$. The same result can be found by taking the limit of the full conditional distribution in the finite model, given by Equation 6 (Neal, 2000).

When combined with some choice of $p(\mathbf{X}|\mathbf{c})$, Equations 6 and 7 are sufficient to define Gibbs samplers for finite and infinite mixture models respectively. Demonstrations of Gibbs sampling in infinite mixture models are provided by Neal (2000) and Rasmussen (2000). Similar MCMC algorithms are presented in Bush and MacEachern (1996), West et al. (1994), Escobar and West (1995) and Ishwaran and James (2001). Algorithms that go beyond the local changes in class assignments allowed by a Gibbs sampler are given by Jain and Neal (2004) and Dahl (2003).

2.5 Summary

Our review of infinite mixture models serves three purposes: it shows that infinite statistical models can be defined by specifying priors over infinite combinatorial objects; it illustrates how these priors can be derived by taking the limit of priors for finite models; and it demonstrates that inference in these models can remain possible, despite the large hypothesis spaces they imply. However, infinite mixture models are still fundamentally limited in their representation of objects, assuming that each object can only belong to a single class. In the next two sections, we use the insights underlying infinite mixture models to derive methods for representing objects in terms of infinitely many latent features, leading us to derive a distribution on infinite binary matrices.

3. Latent Feature Models

In a latent feature model, each object is represented by a vector of latent feature values $\mathbf{f}_i$, and the properties $\mathbf{x}_i$ are generated from a distribution determined by those latent feature values. Latent feature values can be continuous, as in factor analysis (Roweis and Ghahramani, 1999) and probabilistic principal component analysis (PCA; Tipping and Bishop, 1999), or discrete, as in cooperative vector quantization (CVQ; Zemel and Hinton, 1994; Ghahramani, 1995). In the remainder of this section, we will assume that feature values are continuous. Using the matrix $\mathbf{F} = \left[\mathbf{f}_1^T \, \mathbf{f}_2^T \, \cdots \, \mathbf{f}_N^T\right]^T$ to indicate the latent feature values for all N objects, the model is specified by a prior over features, $p(\mathbf{F})$, and a distribution over observed property matrices conditioned on those features, $p(\mathbf{X}|\mathbf{F})$. As with latent class models, these distributions can be dealt with separately: $p(\mathbf{F})$ specifies the number of features, their probability, and the distribution over values associated with each feature, while $p(\mathbf{X}|\mathbf{F})$ determines how these features relate to the properties of objects. Our focus will be on $p(\mathbf{F})$, showing how such a prior can be defined without placing an upper bound on the number of features.

We can break the matrix $\mathbf{F}$ into two components: a binary matrix $\mathbf{Z}$ indicating which features are possessed by each object, with $z_{ik} = 1$ if object i has feature k and 0 otherwise, and a second matrix $\mathbf{V}$ indicating the value of each feature for each object. $\mathbf{F}$ can be expressed as the elementwise (Hadamard) product of $\mathbf{Z}$ and $\mathbf{V}$, $\mathbf{F} = \mathbf{Z} \otimes \mathbf{V}$, as illustrated in Figure 3. In many latent feature models, such as PCA and CVQ, objects have non-zero values on every feature, and every entry of $\mathbf{Z}$ is 1. In *sparse* latent feature models (e.g., sparse PCA; d'Aspremont et al., 2004; Jolliffe and Uddin, 2003; Zou et al., 2006) only a subset of features take on non-zero values for each object, and $\mathbf{Z}$ picks out these subsets.

A prior on $\mathbf{F}$ can be defined by specifying priors for $\mathbf{Z}$ and $\mathbf{V}$ separately, with $p(\mathbf{F}) = P(\mathbf{Z})p(\mathbf{V})$. We will focus on defining a prior on $\mathbf{Z}$, since the effective dimensionality of a latent feature model is determined by $\mathbf{Z}$. Assuming that $\mathbf{Z}$ is sparse, we can define a prior for infinite latent feature models by defining a distribution over infinite binary matrices. Our analysis of latent class models provides two desiderata for such a distribution: objects should be exchangeable, and inference should be tractable. It also suggests a method by which these desiderata can be satisfied: start with a model that assumes a finite number of features, and consider the limit as the number of features approaches infinity.

4. A Distribution on Infinite Sparse Binary Matrices

In this section, we derive a distribution on infinite binary matrices by starting with a simple model that assumes K features, and then taking the limit as $K \to \infty$. The resulting distribution corresponds to a simple generative process, which we term the Indian buffet process.

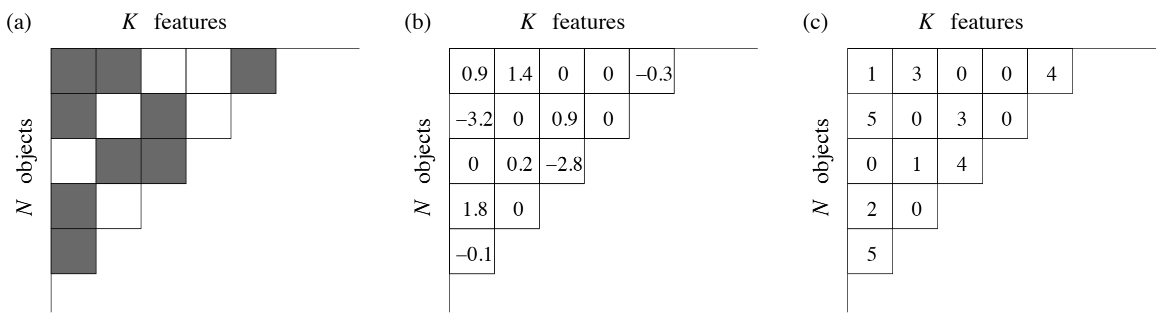


Figure 3: Feature matrices. A binary matrix $\mathbf{Z}$, as shown in (a), can be used as the basis for sparse infinite latent feature models, indicating which features take non-zero values. Elementwise multiplication of $\mathbf{Z}$ by a matrix $\mathbf{V}$ of continuous values gives a representation like that shown in (b). If $\mathbf{V}$ contains discrete values, we obtain a representation like that shown in (c).

4.1 A Finite Feature Model

We have N objects and K features, and the possession of feature k by object i is indicated by a binary variable z_{ik}. Each object can possess multiple features. The z_{ik} thus form a binary $N \times K$ feature matrix, $\mathbf{Z}$. We will assume that each object possesses feature k with probability π_k, and that the features are generated independently. In contrast to the class models discussed above, for which $\sum_k \theta_k = 1$, the probabilities π_k can each take on any value in $[0,1]$. Under this model, the probability of a matrix $\mathbf{Z}$ given $\pi = \{\pi_1, \pi_2, \ldots, \pi_K\}$, is

$$P(\mathbf{Z}|\pi) = \prod_{k=1}^{K} \prod_{i=1}^{N} P(z_{ik}|\pi_k) = \prod_{k=1}^{K} \pi_k^{m_k}(1-\pi_k)^{N-m_k},$$

where $m_k = \sum_{i=1}^{N} z_{ik}$ is the number of objects possessing feature k.

We can define a prior on π by assuming that each π_k follows a beta distribution. The beta distribution has parameters r and s, and is conjugate to the binomial. The probability of any π_k under the Beta(r,s) distribution is given by

$$p(\pi_k) = \frac{\pi_k^{r-1}(1-\pi_k)^{s-1}}{B(r,s)},$$

where $B(r,s)$ is the beta function,

$$\begin{aligned} B(r,s) &= \int_0^1 \pi_k^{r-1}(1-\pi_k)^{s-1}\, d\pi_k \\ &= \frac{\Gamma(r)\Gamma(s)}{\Gamma(r+s)}. \end{aligned} \tag{8}$$

We will take $r = \frac{\alpha}{K}$ and $s = 1$, so Equation 8 becomes

$$B(\tfrac{\alpha}{K},1) = \frac{\Gamma(\frac{\alpha}{K})}{\Gamma(1+\frac{\alpha}{K})} = \frac{K}{\alpha},$$

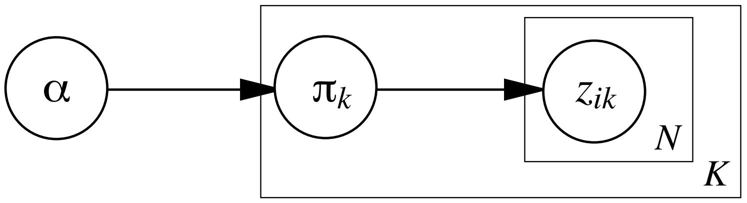


Figure 4: Graphical model for the beta-binomial model used in defining the Indian buffet process. Nodes are variables, arrows indicate dependencies, and plates (Buntine, 1994) indicate replicated structures.

exploiting the recursive definition of the gamma function.[3]

The probability model we have defined is

$$\begin{aligned} \pi_k \,|\, \alpha \ &\sim \text{Beta}(\tfrac{\alpha}{K}, 1), \\ z_{ik} \,|\, \pi_k &\sim \text{Bernoulli}(\pi_k). \end{aligned} \tag{9}$$

Each z_{ik} is independent of all other assignments, conditioned on π_k, and the π_k are generated independently. A graphical model illustrating the dependencies among these variables is shown in Figure 4. Having defined a prior on π, we can simplify this model by integrating over all values for π rather than representing them explicitly. The marginal probability of a binary matrix $\mathbf{Z}$ is

$$\begin{aligned} P(\mathbf{Z}) &= \prod_{k=1}^{K} \int \left(\prod_{i=1}^{N} P(z_{ik}|\pi_k) \right) p(\pi_k)\, d\pi_k \\ &= \prod_{k=1}^{K} \frac{B(m_k + \frac{\alpha}{K}, N - m_k + 1)}{B(\frac{\alpha}{K}, 1)} \\ &= \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(N + 1 + \frac{\alpha}{K})}. \end{aligned} \tag{10}$$

Again, the result follows from conjugacy, this time between the binomial and beta distributions. This distribution is exchangeable, depending only on the counts m_k.

This model has the important property that the expectation of the number of non-zero entries in the matrix $\mathbf{Z}$, $E\left[\mathbf{1}^T\mathbf{Z}\mathbf{1}\right] = E\left[\sum_{ik} z_{ik}\right]$, has an upper bound that is independent of K. Since each column of $\mathbf{Z}$ is independent, the expectation is K times the expectation of the sum of a single column, $E\left[\mathbf{1}^T\mathbf{z}_k\right]$. This expectation is easily computed,

$$E\left[\mathbf{1}^T\mathbf{z}_k\right] = \sum_{i=1}^{N} E(z_{ik}) = \sum_{i=1}^{N} \int_0^1 \pi_k p(\pi_k)\, d\pi_k = N\frac{\frac{\alpha}{K}}{1 + \frac{\alpha}{K}}, \tag{11}$$

where the result follows from the fact that the expectation of a Beta(r,s) random variable is $\frac{r}{r+s}$. Consequently, $E\left[\mathbf{1}^T\mathbf{Z}\mathbf{1}\right] = KE\left[\mathbf{1}^T\mathbf{z}_k\right] = \frac{N\alpha}{1+\frac{\alpha}{K}}$. For finite K, the expectation of the number of entries in $\mathbf{Z}$ is bounded above by $N\alpha$.

3. The motivation for choosing $r = \frac{\alpha}{K}$ will be clear when we take the limit $K \to \infty$ in Section 4.3, while the choice of $s = 1$ will be relaxed in Section 7.1.

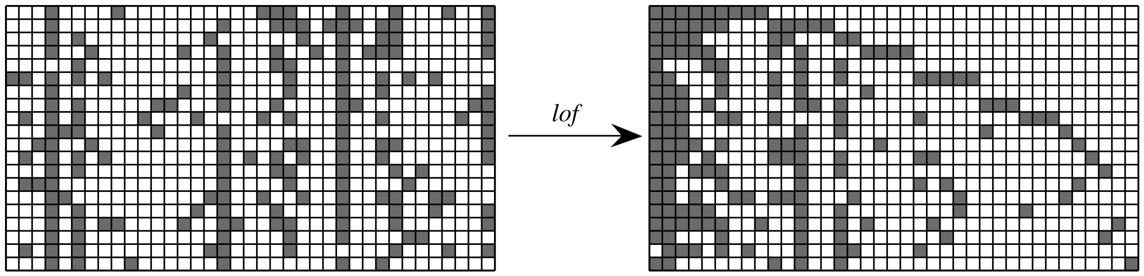


Figure 5: Binary matrices and the left-ordered form. The binary matrix on the left is transformed into the left-ordered binary matrix on the right by the function $lof(\cdot)$. This left-ordered matrix was generated from the exchangeable Indian buffet process with $\alpha = 10$. Empty columns are omitted from both matrices.

4.2 Equivalence Classes

In order to find the limit of the distribution specified by Equation 10 as $K \to \infty$, we need to define equivalence classes of binary matrices—the analogue of partitions for assignment vectors. Identifying these equivalence classes makes it easier to be precise about the objects over which we are defining probability distributions, but the reader who is satisfied with the intuitive idea of taking the limit as $K \to \infty$ can safely skip the technical details presented in this section.

Our equivalence classes will be defined with respect to a function on binary matrices, $lof(\cdot)$. This function maps binary matrices to *left-ordered* binary matrices. $lof(\mathbf{Z})$ is obtained by ordering the columns of the binary matrix $\mathbf{Z}$ from left to right by the magnitude of the binary number expressed by that column, taking the first row as the most significant bit. The left-ordering of a binary matrix is shown in Figure 5. In the first row of the left-ordered matrix, the columns for which $z_{1k} = 1$ are grouped at the left. In the second row, the columns for which $z_{2k} = 1$ are grouped at the left of the sets for which $z_{1k} = 1$. This grouping structure persists throughout the matrix.

Considering the process of placing a binary matrix in left-ordered form motivates the definition of a further technical term. The *history* of feature k at object i is defined to be $(z_{1k}, \ldots, z_{(i-1)k})$. Where no object is specified, we will use *history* to refer to the full history of feature k, $(z_{1k}, \ldots, z_{Nk})$. We will individuate the histories of features using the decimal equivalent of the binary numbers corresponding to the column entries. For example, at object 3, features can have one of four histories: 0, corresponding to a feature with no previous assignments, 1, being a feature for which $z_{2k} = 1$ but $z_{1k} = 0$, 2, being a feature for which $z_{1k} = 1$ but $z_{2k} = 0$, and 3, being a feature possessed by both previous objects were assigned. K_h will denote the number of features possessing the history h, with K_0 being the number of features for which $m_k = 0$ and $K_+ = \sum_{h=1}^{2^N-1} K_h$ being the number of features for which $m_k > 0$, so $K = K_0 + K_+$. The function lof thus places the columns of a matrix in ascending order of their histories.

$lof(\cdot)$ is a many-to-one function: many binary matrices reduce to the same left-ordered form, and there is a unique left-ordered form for every binary matrix. We can thus use $lof(\cdot)$ to define a set of equivalence classes. Any two binary matrices $\mathbf{Y}$ and $\mathbf{Z}$ are lof-equivalent if $lof(\mathbf{Y}) = lof(\mathbf{Z})$, that is, if $\mathbf{Y}$ and $\mathbf{Z}$ map to the same left-ordered form. The lof-equivalence class of a binary matrix $\mathbf{Z}$, denoted $[\mathbf{Z}]$, is the set of binary matrices that are lof-equivalent to $\mathbf{Z}$. lof-equivalence classes are preserved through permutation of either the rows or the columns of a matrix, provided the same permutations are applied to the other members of the equivalence class. Performing inference at the level of lof-equivalence classes is appropriate in models where feature order is not identifiable, with $p(\mathbf{X}|\mathbf{F})$ being unaffected by the order of the columns of $\mathbf{F}$. Any model in which the probability of $\mathbf{X}$ is specified in terms of a linear function of $\mathbf{F}$, such as PCA or CVQ, has this property.

We need to evaluate the cardinality of $[\mathbf{Z}]$, being the number of matrices that map to the same left-ordered form. The columns of a binary matrix are not guaranteed to be unique: since an object can possess multiple features, it is possible for two features to be possessed by exactly the same set of objects. The number of matrices in $[\mathbf{Z}]$ is reduced if $\mathbf{Z}$ contains identical columns, since some re-orderings of the columns of $\mathbf{Z}$ result in exactly the same matrix. Taking this into account, the cardinality of $[\mathbf{Z}]$ is $\binom{K}{K_0 \ldots K_{2^N-1}} = \frac{K!}{\prod_{h=0}^{2^N-1} K_h!}$, where K_h is the count of the number of columns with full history h.

lof-equivalence classes play the same role for binary matrices as partitions do for assignment vectors: they collapse together all binary matrices (assignment vectors) that differ only in column ordering (class labels). This relationship can be made precise by examining the lof-equivalence classes of binary matrices constructed from assignment vectors. Define the *class matrix* generated by an assignment vector $\mathbf{c}$ to be a binary matrix $\mathbf{Z}$ where $z_{ik} = 1$ if and only if $c_i = k$. It is straightforward to show that the class matrices generated by two assignment vectors that correspond to the same partition belong to the same lof-equivalence class, and vice versa.

4.3 Taking the Infinite Limit

Under the distribution defined by Equation 10, the probability of a particular lof-equivalence class of binary matrices, $[\mathbf{Z}]$, is

$$\begin{aligned} P([\mathbf{Z}]) &= \sum_{\mathbf{Z} \in [\mathbf{Z}]} P(\mathbf{Z}) \\ &= \frac{K!}{\prod_{h=0}^{2^N-1} K_h!} \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(N + 1 + \frac{\alpha}{K})}. \end{aligned} \tag{12}$$

In order to take the limit of this expression as $K \to \infty$, we will divide the columns of $\mathbf{Z}$ into two subsets, corresponding to the features for which $m_k = 0$ and the features for which $m_k > 0$. Re-ordering the columns such that $m_k > 0$ if $k \leq K_+$, and $m_k = 0$ otherwise, we can break the product in Equation 12 into two parts, corresponding to these two subsets. The product thus becomes

$$\begin{aligned} & \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(N + 1 + \frac{\alpha}{K})} \\ &= \left(\frac{\frac{\alpha}{K}\Gamma(\frac{\alpha}{K})\Gamma(N+1)}{\Gamma(N + 1 + \frac{\alpha}{K})} \right)^{K-K_+} \prod_{k=1}^{K_+} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(N + 1 + \frac{\alpha}{K})} \\ &= \left(\frac{\frac{\alpha}{K}\Gamma(\frac{\alpha}{K})\Gamma(N+1)}{\Gamma(N + 1 + \frac{\alpha}{K})} \right)^{K} \prod_{k=1}^{K_+} \frac{\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(\frac{\alpha}{K})\Gamma(N+1)} \\ &= \left(\frac{N!}{\prod_{j=1}^{N}(j + \frac{\alpha}{K})} \right)^{K} \left(\frac{\alpha}{K}\right)^{K_+} \prod_{k=1}^{K_+} \frac{(N - m_k)! \prod_{j=1}^{m_k - 1}(j + \frac{\alpha}{K})}{N!}, \end{aligned} \tag{13}$$

where we have used the fact that $\Gamma(x) = (x-1)\Gamma(x-1)$ for $x > 1$. Substituting Equation 13 into Equation 12 and rearranging terms, we can compute our limit

$$\begin{aligned}\lim_{K\to\infty} \frac{\alpha^{K_+}}{\prod_{h=1}^{2^N-1} K_h!} \cdot \frac{K!}{K_0!\,K^{K_+}} \cdot \left(\frac{N!}{\prod_{j=1}^{N}(j+\frac{\alpha}{K})}\right)^K \cdot \prod_{k=1}^{K_+} \frac{(N-m_k)!\prod_{j=1}^{m_k-1}(j+\frac{\alpha}{K})}{N!} \\ = \frac{\alpha^{K_+}}{\prod_{h=1}^{2^N-1} K_h!} \cdot \quad 1 \quad \cdot \quad \exp\{-\alpha H_N\} \quad \cdot \prod_{k=1}^{K_+} \frac{(N-m_k)!(m_k-1)!}{N!},\end{aligned} \tag{14}$$

where H_N is the Nth harmonic number, $H_N = \sum_{j=1}^{N}\frac{1}{j}$. The details of the steps taken in computing this limit are given in Appendix A. Again, this distribution is exchangeable: neither the number of identical columns nor the column sums are affected by the ordering on objects.

4.4 The Indian Buffet Process

The probability distribution defined in Equation 14 can be derived from a simple stochastic process. As with the CRP, this process assumes an ordering on the objects, generating the matrix sequentially using this ordering. We will also use a culinary metaphor in defining our stochastic process, appropriately adjusted for geography.[4] Many Indian restaurants offer lunchtime buffets with an apparently infinite number of dishes. We can define a distribution over infinite binary matrices by specifying a procedure by which customers (objects) choose dishes (features).

In our Indian buffet process (IBP), N customers enter a restaurant one after another. Each customer encounters a buffet consisting of infinitely many dishes arranged in a line. The first customer starts at the left of the buffet and takes a serving from each dish, stopping after a Poisson(α) number of dishes as his plate becomes overburdened. The ith customer moves along the buffet, sampling dishes in proportion to their popularity, serving himself with probability $\frac{m_k}{i}$, where m_k is the number of previous customers who have sampled a dish. Having reached the end of all previous sampled dishes, the ith customer then tries a Poisson($\frac{\alpha}{i}$) number of new dishes.

We can indicate which customers chose which dishes using a binary matrix $\mathbf{Z}$ with N rows and infinitely many columns, where $z_{ik} = 1$ if the ith customer sampled the kth dish. Figure 6 shows a matrix generated using the IBP with $\alpha = 10$. The first customer tried 17 dishes. The second customer tried 7 of those dishes, and then tried 3 new dishes. The third customer tried 3 dishes tried by both previous customers, 5 dishes tried by only the first customer, and 2 new dishes. Vertically concatenating the choices of the customers produces the binary matrix shown in the figure.

Using $K_1^{(i)}$ to indicate the number of new dishes sampled by the ith customer, the probability of any particular matrix being produced by this process is

$$P(\mathbf{Z}) = \frac{\alpha^{K_+}}{\prod_{i=1}^{N} K_1^{(i)}!} \exp\{-\alpha H_N\} \prod_{k=1}^{K_+} \frac{(N-m_k)!(m_k-1)!}{N!}. \tag{15}$$

As can be seen from Figure 6, the matrices produced by this process are generally not in left-ordered form. However, these matrices are also not ordered arbitrarily because the Poisson draws always result in choices of new dishes that are to the right of the previously sampled dishes. Customers are not exchangeable under this distribution, as the number of dishes counted as $K_1^{(i)}$ depends upon

4. This work was started when both authors were at the Gatsby Computational Neuroscience Unit in London, where the Indian buffet is the dominant culinary metaphor.

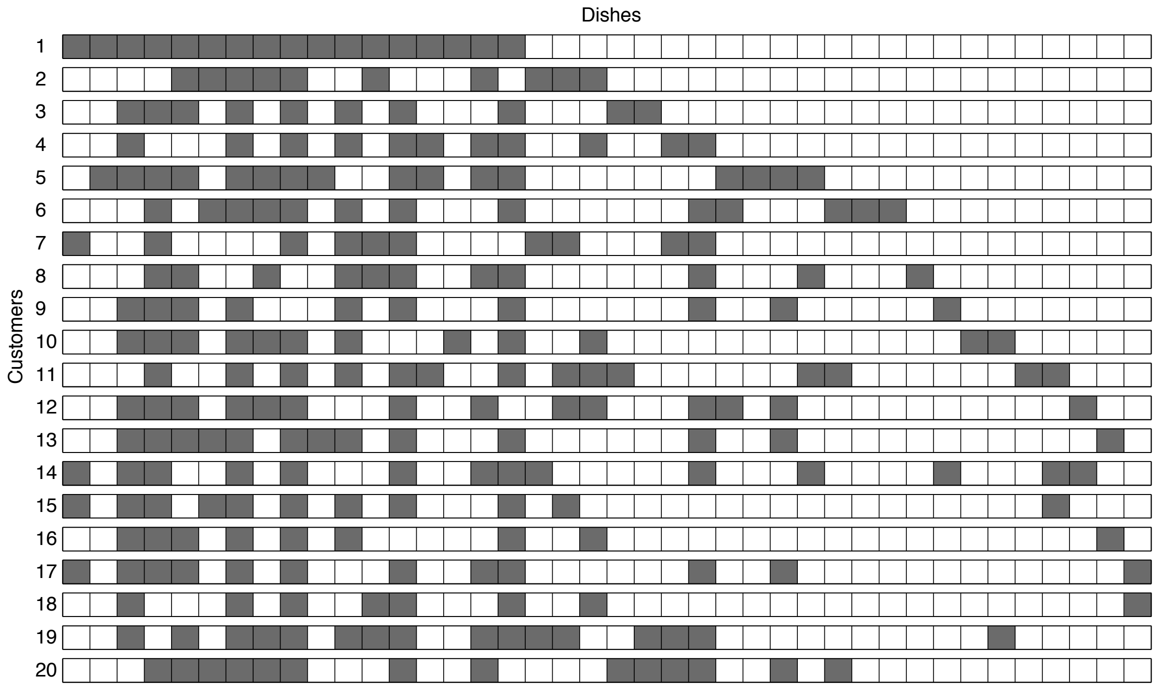


Figure 6: A binary matrix generated by the Indian buffet process with $\alpha = 10$.

the order in which the customers make their choices. However, if we only pay attention to the lof-equivalence classes of the matrices generated by this process, we obtain the exchangeable distribution $P([\mathbf{Z}])$ given by Equation 14: $\frac{\prod_{i=1}^{N} K_1^{(i)}!}{\prod_{h=1}^{2^N-1} K_h!}$ matrices generated via this process map to the same left-ordered form, and $P([\mathbf{Z}])$ is obtained by multiplying $P(\mathbf{Z})$ from Equation 15 by this quantity.

It is possible to define a similar sequential process that directly produces a distribution on lof equivalence classes in which customers are exchangeable, but this requires more effort on the part of the customers. In the *exchangeable* Indian buffet process, the first customer samples a Poisson(α) number of dishes, moving from left to right. The ith customer moves along the buffet, and makes a single decision for each set of dishes with the same history. If there are K_h dishes with history h, under which m_h previous customers have sampled each of those dishes, then the customer samples a Binomial($\frac{m_h}{i}, K_h$) number of those dishes, starting at the left. Having reached the end of all previous sampled dishes, the ith customer then tries a Poisson($\frac{\alpha}{i}$) number of new dishes. Attending to the history of the dishes and always sampling from the left guarantees that the resulting matrix is in left-ordered form, and it is easy to show that the matrices produced by this process have the same probability as the corresponding lof-equivalence classes under Equation 14.

4.5 A Distribution over Collections of Histories

In Section 4.2, we noted that lof-equivalence classes of binary matrices generated from assignment vectors correspond to partitions. Likewise, lof-equivalence classes of general binary matrices correspond to simple combinatorial structures: vectors of non-negative integers. Fixing some ordering of N objects, a collection of feature histories on those objects can be represented by a frequency vector $\mathbf{K} = (K_1, \ldots, K_{2^N-1})$, indicating the number of times each history appears in the collection. A collection of feature histories can be translated into a left-ordered binary matrix by horizontally concatenating an appropriate number of copies of the binary vector representing each history into a matrix. A left-ordered binary matrix can be translated into a collection of feature histories by counting the number of times each history appears in that matrix. Since partitions are a subset of all collections of histories—namely those collections in which each object appears in only one history—this process is strictly more general than the CRP.

This connection between lof-equivalence classes of feature matrices and collections of feature histories suggests another means of deriving the distribution specified by Equation 14, operating directly on the frequencies of these histories. We can define a distribution on vectors of non-negative integers $\mathbf{K}$ by assuming that each K_h is generated independently from a Poisson distribution with parameter $\alpha B(m_h, N - m_h + 1) = \alpha \frac{(m_h-1)!(N-m_h)!}{N!}$ where m_h is the number of non-zero elements in the history h. This gives

$$\begin{aligned} P(\mathbf{K}) &= \prod_{h=1}^{2^N-1} \frac{\left(\alpha \frac{(m_h-1)!(N-m_h)!}{N!}\right)^{K_h}}{K_h!} \exp\left\{-\alpha \frac{(m_h-1)!(N-m_h)!}{N!}\right\} \\ &= \frac{\alpha^{\sum_{h=1}^{2^N-1} K_h}}{\prod_{h=1}^{2^N-1} K_h!} \exp\{-\alpha H_N\} \prod_{h=1}^{2^N-1} \left(\frac{(m_h-1)!(N-m_h)!}{N!}\right)^{K_h}, \end{aligned}$$

which is easily seen to be the same as $P([\mathbf{Z}])$ in Equation 14. The harmonic number in the exponential term is obtained by summing $\frac{(m_h-1)!(N-m)!}{N!}$ over all histories h. There are $\binom{N}{j}$ histories for which $m_h = j$, so we have

$$\sum_{h=1}^{2^N-1} \frac{(m_h-1)!(N-m_h)!}{N!} = \sum_{j=1}^{N} \binom{N}{j} \frac{(j-1)!(N-j)!}{N!} = \sum_{j=1}^{N} \frac{1}{j} = H_N. \tag{16}$$

4.6 Properties of this Distribution

These different views of the distribution specified by Equation 14 make it straightforward to derive some of its properties. First, the effective dimension of the model, K_+, follows a Poisson(αH_N) distribution. This is easily shown using the generative process described in Section 4.5: $K_+ = \sum_{h=1}^{2^N-1} K_h$, and under this process is thus the sum of a set of Poisson distributions. The sum of a set of Poisson distributions is a Poisson distribution with parameter equal to the sum of the parameters of its components. Using Equation 16, this is αH_N. Alternatively, we can use the fact that the number of new columns generated at the ith row is Poisson($\frac{\alpha}{i}$), with the total number of columns being the sum of these quantities.

A second property of this distribution is that the number of features possessed by each object follows a Poisson(α) distribution. This follows from the definition of the exchangeable IBP. The first customer chooses a Poisson(α) number of dishes. By exchangeability, all other customers must also choose a Poisson(α) number of dishes, since we can always specify an ordering on customers which begins with a particular customer.

Finally, it is possible to show that $\mathbf{Z}$ remains sparse as $K \to \infty$. The simplest way to do this is to exploit the previous result: if the number of features possessed by each object follows a Poisson(α) distribution, then the expected number of entries in $\mathbf{Z}$ is $N\alpha$. This is consistent with the quantity obtained by taking the limit of this expectation in the finite model, which is given in Equation 11: $\lim_{K\to\infty} E\left[\mathbf{1}^T\mathbf{Z}\mathbf{1}\right] = \lim_{K\to\infty} \frac{N\alpha}{1+\frac{\alpha}{K}} = N\alpha$.

4.7 Inference by Gibbs Sampling

We have defined a distribution over infinite binary matrices that satisfies one of our desiderata—objects (the rows of the matrix) are exchangeable under this distribution. It remains to be shown that inference in infinite latent feature models is tractable, as was the case for infinite mixture models. We will derive a Gibbs sampler for sampling from the distribution defined by the IBP, which suggests a strategy for inference in latent feature models in which the exchangeable IBP is used as a prior. We will consider alternative inference algorithms later in the paper.

To sample from the distribution defined by the IBP, we need to compute the conditional distribution $P(z_{ik} = 1|\mathbf{Z}_{-(ik)})$, where $\mathbf{Z}_{-(ik)}$ denotes the entries of $\mathbf{Z}$ other than z_{ik}. In the finite model, where $P(\mathbf{Z})$ is given by Equation 10, it is straightforward to compute the conditional distribution for any z_{ik}. Integrating over π_k gives

$$\begin{aligned} P(z_{ik} = 1|\mathbf{z}_{-i,k}) &= \int_0^1 P(z_{ik}|\pi_k)p(\pi_k|\mathbf{z}_{-i,k})\,d\pi_k \\ &= \frac{m_{-i,k}+\frac{\alpha}{K}}{N+\frac{\alpha}{K}}, \end{aligned} \tag{17}$$

where $\mathbf{z}_{-i,k}$ is the set of assignments of other objects, not including i, for feature k, and $m_{-i,k}$ is the number of objects possessing feature k, not including i. We need only condition on $\mathbf{z}_{-i,k}$ rather than $\mathbf{Z}_{-(ik)}$ because the columns of the matrix are generated independently under this prior.

In the infinite case, we can derive the conditional distribution from the exchangeable IBP. Choosing an ordering on objects such that the ith object corresponds to the last customer to visit the buffet, we obtain

$$P(z_{ik} = 1|\mathbf{z}_{-i,k}) = \frac{m_{-i,k}}{N}, \tag{18}$$

for any k such that $m_{-i,k} > 0$. The same result can be obtained by taking the limit of Equation 17 as $K \to \infty$. Similarly the number of new features associated with object i should be drawn from a Poisson($\frac{\alpha}{N}$) distribution. This can also be derived from Equation 17, using the same kind of limiting argument as that presented above to obtain the terms of the Poisson.

This analysis results in a simple Gibbs sampling algorithm for generating samples from the distribution defined by the IBP. We start with an arbitrary binary matrix. We then iterate through the rows of the matrix, i. For each column k, if $m_{-i,k}$ is greater than 0 we set $z_{ik} = 1$ with probability given by Equation 18. Otherwise, we delete that column. At the end of the row, we add Poisson($\frac{\alpha}{N}$) new columns that have ones in that row. After sufficiently many passes through the rows, the resulting matrix will be a draw from the distribution $P(\mathbf{Z})$ given by Equation 15.

This algorithm suggests a heuristic strategy for sampling from the posterior distribution $P(\mathbf{Z}|\mathbf{X})$ in a model that uses the IBP to define a prior on $\mathbf{Z}$. In this case, we need to sample from the full conditional distribution

$$P(z_{ik} = 1|\mathbf{Z}_{-(ik)}, \mathbf{X}) \propto p(\mathbf{X}|\mathbf{Z})P(z_{ik} = 1|\mathbf{Z}_{-(ik)})$$

where $p(\mathbf{X}|\mathbf{Z})$ is the likelihood function for the model, and we assume that parameters of the likelihood have been integrated out. We can proceed as in the Gibbs sampler given above, simply incorporating the likelihood term when sampling z_{ik} for columns for which $m_{-i,k}$ is greater than 0 and drawing the new columns from a distribution where the prior is Poisson($\frac{\alpha}{N}$) and the likelihood is given by $P(\mathbf{X}|\mathbf{Z})$.[5]

5. An Example: A Linear-Gaussian Latent Feature Model with Binary Features

We have derived a prior for infinite sparse binary matrices, and indicated how statistical inference can be done in models defined using this prior. In this section, we will show how this prior can be put to use in models for unsupervised learning, illustrating some of the issues that can arise in this process. We will describe a simple linear-Gaussian latent feature model, in which the features are binary. As above, we will start with a finite model and then consider the infinite limit.

5.1 A Finite Linear-Gaussian Model

In our finite model, the D-dimensional vector of properties of an object i, $\mathbf{x}_i$ is generated from a Gaussian distribution with mean $\mathbf{z}_i\mathbf{A}$ and covariance matrix $\Sigma_X = \sigma_X^2\mathbf{I}$, where $\mathbf{z}_i$ is a K-dimensional binary vector, and $\mathbf{A}$ is a $K \times D$ matrix of weights. In matrix notation, $E[\mathbf{X}] = \mathbf{ZA}$. If $\mathbf{Z}$ is a feature matrix, this is a form of binary factor analysis. The distribution of $\mathbf{X}$ given $\mathbf{Z}$, $\mathbf{A}$, and σ_X is matrix Gaussian:

$$p(\mathbf{X}|\mathbf{Z},\mathbf{A},\sigma_X) = \frac{1}{(2\pi\sigma_X^2)^{ND/2}} \exp\{-\frac{1}{2\sigma_X^2}\mathrm{tr}((\mathbf{X}-\mathbf{ZA})^T(\mathbf{X}-\mathbf{ZA}))\} \tag{19}$$

where $\mathrm{tr}(\cdot)$ is the trace of a matrix. This makes it easy to integrate out the model parameters $\mathbf{A}$. To do so, we need to define a prior on $\mathbf{A}$, which we also take to be matrix Gaussian:

$$p(\mathbf{A}|\sigma_A) = \frac{1}{(2\pi\sigma_A^2)^{KD/2}} \exp\{-\frac{1}{2\sigma_A^2}\mathrm{tr}(\mathbf{A}^T\mathbf{A})\}, \tag{20}$$

where σ_A is a parameter setting the diffuseness of the prior. The dependencies among the variables in this model are shown in Figure 7.

Combining Equations 19 and 20 results in an exponentiated expression involving the trace of

$$\begin{aligned}
&\frac{1}{\sigma_X^2}(\mathbf{X}-\mathbf{ZA})^T(\mathbf{X}-\mathbf{ZA}) + \frac{1}{\sigma_A^2}\mathbf{A}^T\mathbf{A} \\
&\qquad = \frac{1}{\sigma_X^2}\mathbf{X}^T\mathbf{X} - \frac{1}{\sigma_X^2}\mathbf{X}^T\mathbf{ZA} - \frac{1}{\sigma_X^2}\mathbf{A}^T\mathbf{Z}^T\mathbf{X} + \mathbf{A}^T(\frac{1}{\sigma_X^2}\mathbf{Z}^T\mathbf{Z} + \frac{1}{\sigma_A^2}\mathbf{I})\mathbf{A} \\
&\qquad = \frac{1}{\sigma_X^2}(\mathbf{X}^T(\mathbf{I}-\mathbf{ZMZ}^T)\mathbf{X}) + (\mathbf{MZ}^T\mathbf{X}-\mathbf{A})^T(\sigma_X^2\mathbf{M})^{-1}(\mathbf{MZ}^T\mathbf{X}-\mathbf{A}),
\end{aligned}$$

5. As was pointed out by an anonymous reviewer, this is a heuristic strategy rather than a valid algorithm for sampling from the posterior because it violates one of the assumptions of Markov chain Monte Carlo algorithms, with the order in which variables are sampled being dependent on the state of the Markov chain. This is not an issue in the algorithm for sampling from $P(\mathbf{Z})$, since the columns of $\mathbf{Z}$ are independent, and the kernels corresponding to sampling from each of the conditional distributions thus act independently of one another.

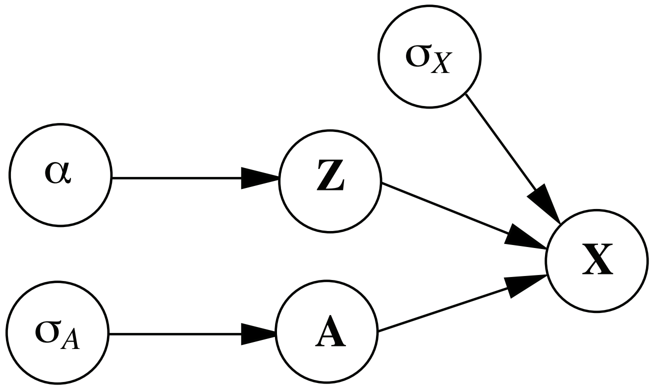


Figure 7: Graphical model for the linear-Gaussian model with binary features.

where $\mathbf{I}$ is the identity matrix, $\mathbf{M} = (\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1}$, and the last line is obtained by completing the square for the quadratic term in $\mathbf{A}$ in the second line. We can then integrate out $\mathbf{A}$ to obtain

$$\begin{aligned}
& p(\mathbf{X}|\mathbf{Z},\sigma_X,\sigma_A) \\
&= \int p(\mathbf{X}|\mathbf{Z},\mathbf{A},\sigma_X)p(\mathbf{A}|\sigma_A)\,d\mathbf{A} \\
&= \frac{1}{(2\pi)^{(N+K)D/2}\sigma_X^{ND}\sigma_A^{KD}}\exp\{-\frac{1}{2\sigma_X^2}\mathrm{tr}(\mathbf{X}^T(\mathbf{I}-\mathbf{Z}\mathbf{M}\mathbf{Z}^T)\mathbf{X})\} \\
&\qquad \int \exp\{-\frac{1}{2}\mathrm{tr}((\mathbf{M}\mathbf{Z}^T\mathbf{X}-\mathbf{A})^T(\sigma_X^2\mathbf{M})^{-1}(\mathbf{M}\mathbf{Z}^T\mathbf{X}-\mathbf{A}))\}\,d\mathbf{A} \\
&= \frac{|\sigma_X^2\mathbf{M}|^{D/2}}{(2\pi)^{ND/2}\sigma_X^{ND}\sigma_A^{KD}}\exp\{-\frac{1}{2\sigma_X^2}\mathrm{tr}(\mathbf{X}^T(\mathbf{I}-\mathbf{Z}\mathbf{M}\mathbf{Z}^T)\mathbf{X})\} \\
&= \frac{1}{(2\pi)^{ND/2}\sigma_X^{(N-K)D}\sigma_A^{KD}|\mathbf{Z}^T\mathbf{Z}+\frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}|^{D/2}} \\
&\qquad \exp\{-\frac{1}{2\sigma_X^2}\mathrm{tr}(\mathbf{X}^T(\mathbf{I}-\mathbf{Z}(\mathbf{Z}^T\mathbf{Z}+\frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1}\mathbf{Z}^T)\mathbf{X})\}. \qquad (21)
\end{aligned}$$

This result is intuitive: the exponentiated term is the difference between the inner product matrix of the raw values of $\mathbf{X}$ and their projections onto the space spanned by $\mathbf{Z}$, regularized to an extent determined by the ratio of the variance of the noise in $\mathbf{X}$ to the variance of the prior on $\mathbf{A}$. This is simply the marginal likelihood for a Bayesian linear regression model (Minka, 2000).

We can use this derivation of $p(\mathbf{X}|\mathbf{Z},\sigma_X,\sigma_A)$ to infer $\mathbf{Z}$ from a set of observations $\mathbf{X}$, provided we have a prior on $\mathbf{Z}$. The finite feature model discussed as a prelude to the IBP is such a prior. The full conditional distribution for z_{ik} is given by:

$$P(z_{ik}|\mathbf{X},\mathbf{Z}_{-(i,k)},\sigma_X,\sigma_A) \propto p(\mathbf{X}|\mathbf{Z},\sigma_X,\sigma_A)P(z_{ik}|\mathbf{z}_{-i,k}). \qquad (22)$$

While evaluating $p(\mathbf{X}|\mathbf{Z},\sigma_X,\sigma_A)$ always involves matrix multiplication, it need not always involve a matrix inverse. $\mathbf{Z}^T\mathbf{Z}$ can be rewritten as $\sum_i \mathbf{z}_i^T\mathbf{z}_i$, allowing us to use rank one updates to efficiently compute the inverse when only one $\mathbf{z}_i$ is modified. Defining $\mathbf{M}_{-i} = (\sum_{j \neq i} \mathbf{z}_j^T \mathbf{z}_j + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1}$, we have

$$\begin{aligned}
\mathbf{M}_{-i} &= (\mathbf{M}^{-1} - \mathbf{z}_i^T \mathbf{z}_i)^{-1} \\
&= \mathbf{M} - \frac{\mathbf{M}\mathbf{z}_i^T \mathbf{z}_i \mathbf{M}}{\mathbf{z}_i \mathbf{M} \mathbf{z}_i^T - 1}, \qquad (23) \\
\mathbf{M} &= (\mathbf{M}_{-i}^{-1} + \mathbf{z}_i^T \mathbf{z}_i)^{-1} \\
&= \mathbf{M}_{-i} - \frac{\mathbf{M}_{-i}\mathbf{z}_i^T \mathbf{z}_i \mathbf{M}_{-i}}{\mathbf{z}_i \mathbf{M}_{-i} \mathbf{z}_i^T + 1}. \qquad (24)
\end{aligned}$$

Iteratively applying these updates allows $p(\mathbf{X}|\mathbf{Z}, \sigma_X, \sigma_A)$, to be computed via Equation 21 for different values of z_{ik} without requiring an excessive number of inverses, although a full rank update should be made occasionally to avoid accumulating numerical errors. The second part of Equation 22, $P(z_{ik}|\mathbf{z}_{-i,k})$, can be evaluated using Equation 17.

5.2 Taking the Infinite Limit

To make sure that we can define an infinite version of this model, we need to check that $p(\mathbf{X}|\mathbf{Z}, \sigma_X, \sigma_A)$ remains well-defined if $\mathbf{Z}$ has an unbounded number of columns. $\mathbf{Z}$ appears in two places in Equation 21: in $|\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}|$ and in $\mathbf{Z}(\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1}\mathbf{Z}^T$. We will examine how these behave as $K \to \infty$.

If $\mathbf{Z}$ is in left-ordered form, we can write it as $[\mathbf{Z}_+ \ \mathbf{Z}_0]$, where $\mathbf{Z}_+$ consists of K_+ columns with sums $m_k > 0$, and $\mathbf{Z}_0$ consists of K_0 columns with sums $m_k = 0$. It follows that the first of the two expressions we are concerned with reduces to

$$\begin{aligned}
\left|\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}\right| &= \left|\left[\begin{array}{cc} \mathbf{Z}_+^T\mathbf{Z}_+ & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array}\right] + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}_K\right| \\
&= \left(\frac{\sigma_X^2}{\sigma_A^2}\right)^{K_0} \left|\mathbf{Z}_+^T\mathbf{Z}_+ + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}_{K_+}\right|. \qquad (25)
\end{aligned}$$

The appearance of K_0 in this expression is not a problem, as we will see shortly. The abundance of zeros in $\mathbf{Z}$ leads to a direct reduction of the second expression to

$$\mathbf{Z}(\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1}\mathbf{Z}^T = \mathbf{Z}_+(\mathbf{Z}_+^T\mathbf{Z}_+ + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}_{K_+})^{-1}\mathbf{Z}_+^T,$$

which only uses the finite portion of $\mathbf{Z}$. Combining these results yields the likelihood for the infinite model

$$\begin{aligned}
p(\mathbf{X}|\mathbf{Z}, \sigma_X, \sigma_A) &= \frac{1}{(2\pi)^{ND/2}\sigma_X^{(N-K_+)D}\sigma_A^{K_+D}|\mathbf{Z}_+^T\mathbf{Z}_+ + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}_{K_+}|^{D/2}} \\
&\quad \exp\{-\frac{1}{2\sigma_X^2}\mathrm{tr}(\mathbf{X}^T(\mathbf{I} - \mathbf{Z}_+(\mathbf{Z}_+^T\mathbf{Z}_+ + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}_{K_+})^{-1}\mathbf{Z}_+^T)\mathbf{X})\}. \qquad (26)
\end{aligned}$$

The K_+ in the exponents of σ_A and σ_X appears as a result of introducing $D/2$ multiples of the factor of $\left(\frac{\sigma_X^2}{\sigma_A^2}\right)^{K_0}$ from Equation 25. The likelihood for the infinite model is thus just the likelihood for the finite model defined on the first K_+ columns of $\mathbf{Z}$.

The heuristic Gibbs sampling algorithm defined in Section 4.7 can now be used in this model. Assignments to classes for which $m_{-i,k} > 0$ are drawn in the same way as for the finite model, via Equation 22, using Equation 26 to obtain $p(\mathbf{X}|\mathbf{Z}, \sigma_X, \sigma_A)$ and Equation 18 for $P(z_{ik}|\mathbf{z}_{-i,k})$. As in the finite case, Equations 23 and 24 can be used to compute inverses efficiently. The distribution over the number of new features can be approximated by truncation, computing probabilities for a range of values of $K_1^{(i)}$ up to some reasonable upper bound. For each value, $p(\mathbf{X}|\mathbf{Z}, \sigma_X, \sigma_A)$ can be computed from Equation 26, and the prior on the number of new classes is Poisson($\frac{\alpha}{N}$). More elaborate samplers which do not require truncation are presented in Meeds et al. (2007) and in Teh et al. (2007).

5.3 Demonstrations

As a first demonstration of the ability of this algorithm to recover the latent structure responsible for having generated observed data, we applied the Gibbs sampler for the infinite linear-Gaussian model to a simulated data set consisting of 100 6×6 images, each generated by randomly assigning a feature to each image to a class with probability 0.5, and taking a linear combination of the weights associated with features to which the images were assigned (a similar data set was used by Ghahramani, 1995). Some of these images are shown in Figure 8, together with the weights $\mathbf{A}$ that were used to generate them. The non-zero elements of $\mathbf{A}$ were all equal to 1.0, and σ_X was set to 0.5, introducing a large amount of noise.

The algorithm was initialized with $K_+ = 1$, choosing the feature assignments for the first column by setting $z_{i1} = 1$ with probability 0.5. σ_A was set to 1.0. The Gibbs sampler rapidly discovered that four classes were sufficient to account for the data, and converged to a distribution focused on matrices $\mathbf{Z}$ that closely matched the true class assignments. The results are shown in Figure 8. Each of the features is represented by the posterior mean of the feature weights, $\mathbf{A}$, given $\mathbf{X}$ and $\mathbf{Z}$, which is

$$E[\mathbf{A}|\mathbf{X},\mathbf{Z}] = (\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1}\mathbf{Z}^T\mathbf{X}.$$

for a single sample $\mathbf{Z}$. The results shown in the figure are from the 200th sample produced by the algorithm.

These results indicate that the algorithm can recover the features used to generate simulated data. In a further test of the algorithm with more realistic data, we applied it to a data set consisting of 100 240×320 pixel images. We represented each image, $\mathbf{x}_i$, using a 100-dimensional vector corresponding to the weights of the mean image and the first 99 principal components. Each image contained up to four everyday objects—a \$20 bill, a Klein bottle, a prehistoric handaxe, and a cellular phone. The objects were placed in fixed locations, but were put into the scenes by hand, producing some small variation in location. The images were then taken with a low resolution webcam. Each object constituted a single latent feature responsible for the observed pixel values. The images were generated by sampling a feature vector, $\mathbf{z}_i$, from a distribution under which each feature was present with probability 0.5, and then taking a photograph containing the appropriate objects using a LogiTech digital webcam. Sample images are shown in Figure 9 (a). The only noise in the images was the noise from the camera.

The Gibbs sampler was initialized with $K_+ = 1$, choosing the feature assignments for the first column by setting $z_{i1} = 1$ with probability 0.5. σ_A, σ_X, and α were initially set to 0.5, 1.7, and 1 respectively, and then sampled by adding Metropolis steps to the MCMC algorithm. Figure 9

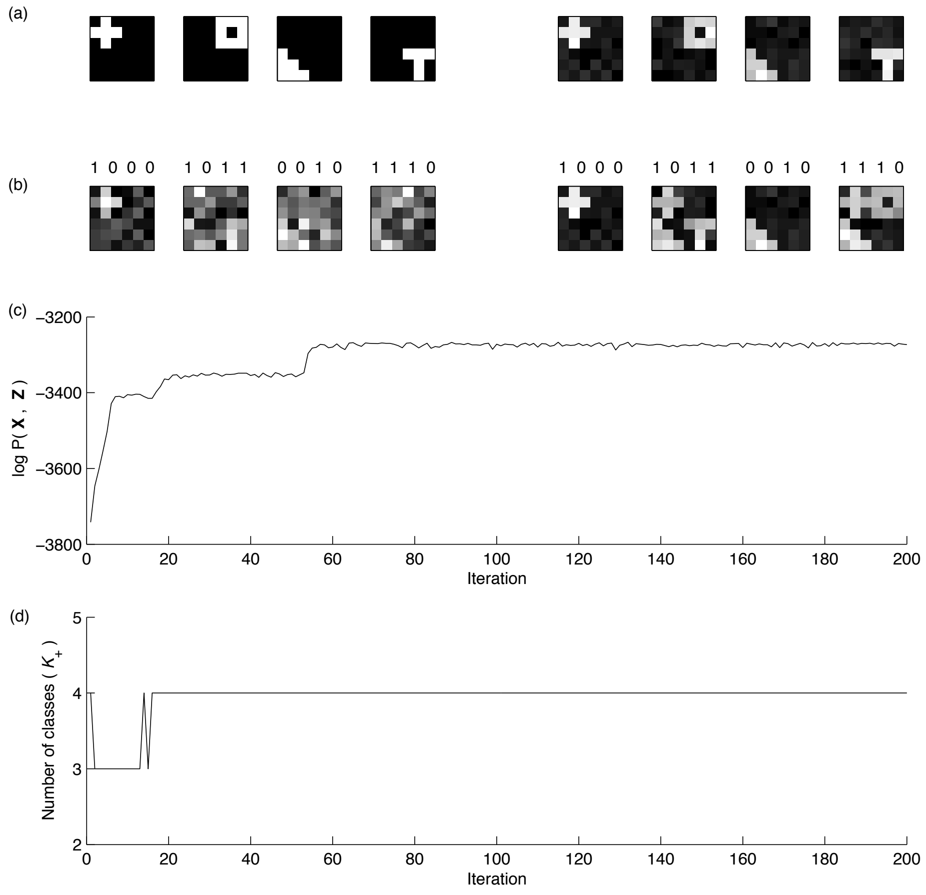


Figure 8: Demonstration of the linear-Gaussian model described in the text, using a binary representation. (a) 100 images were generated as binary linear combinations of four sets of class weights, shown in the images on the left. The images on the right are the posterior mean weights **A** for a single sample of **Z** after 200 iterations, ordered to match the true classes. (b) The images on the left show four of the datapoints to which the model was applied. The numbers above each image indicate the classes responsible for generating that image, matching the order above. The images on the right show the predictions of the model for these images, based on the posterior mean weights, together with the class assignments from the sampled **Z**. (c) Trace plot of $\log P(\mathbf{X},\mathbf{Z})$ over 200 iterations. (d) Trace plot of K_+, the number of classes, over 200 iterations. The data were generated from a model with $K_+ = 4$.

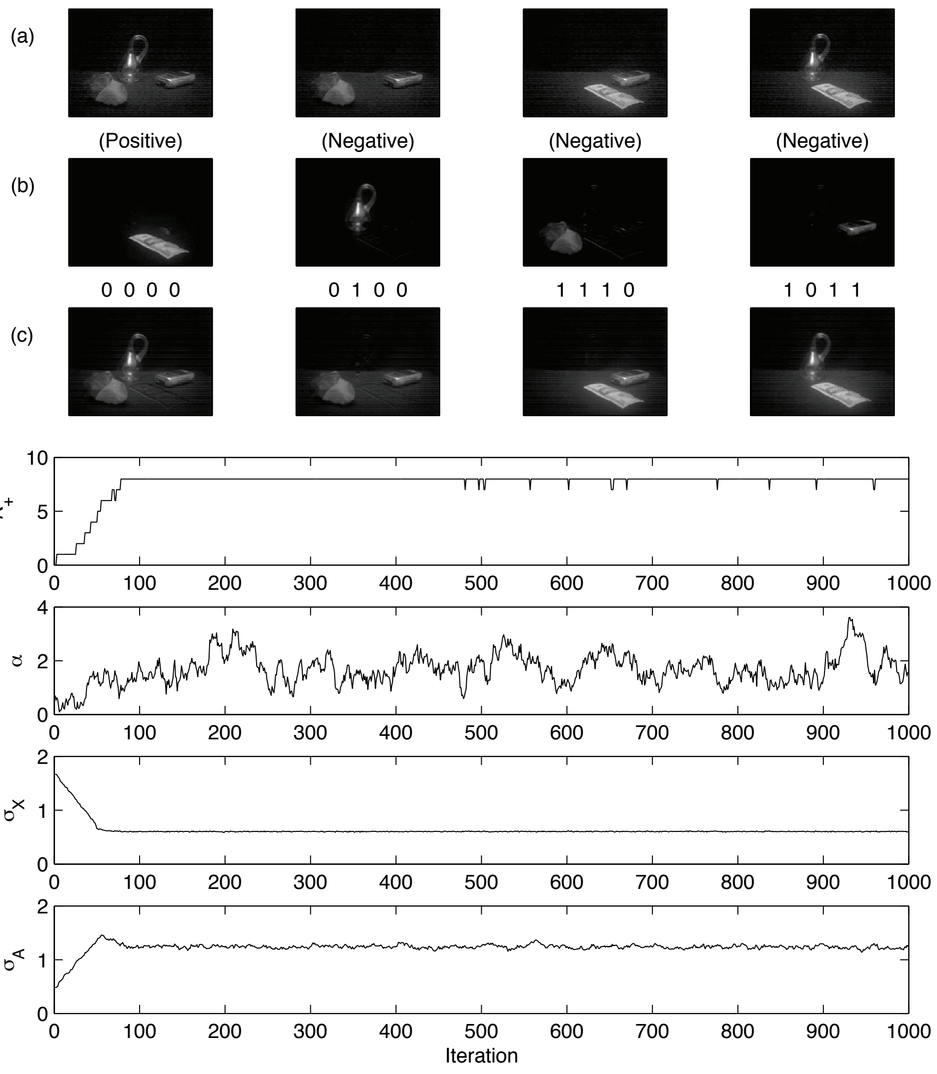


Figure 9: Data and results for the application of the infinite linear-Gaussian model to photographic images. (a) Four sample images from the 100 in the data set. Each image had 320×240 pixels, and contained from zero to four everyday objects. (b) The posterior mean of the weights (**A**) for the four most frequent binary features from the 1000th sample. Each image corresponds to a single feature. These features perfectly indicate the presence or absence of the four objects. The first feature indicates the presence of the \$20 bill, the other three indicate the absence of the Klein bottle, the handaxe, and the cellphone. (c) Reconstructions of the images in (a) using the binary codes inferred for those images. These reconstructions are based upon the posterior mean of **A** for the 1000th sample. For example, the code for the first image indicates that the \$20 bill is absent, while the other three objects are not. The lower panels show trace plots for the dimensionality of the representation (K_+) and the parameters α, σ_X, and σ_A over 1000 iterations of sampling. The values of all parameters stabilize after approximately 100 iterations.

shows trace plots for the first 1000 iterations of MCMC for the number of features used by at least one object, K_+, and the model parameters σ_A, σ_X, and α. All of these quantities stabilized after approximately 100 iterations, with the algorithm finding solutions with approximately seven latent features.

Figure 9 (b) shows the posterior mean of $\mathbf{a}_k$ for the four most frequent features in the 1000th sample produced by the algorithm. These features perfectly indicated presence and absence of the four objects. Three less common features coded for slight differences in the locations of those objects. Figure 9 (c) shows the feature vectors $\mathbf{z}_i$ from this sample for the four images in Figure 9(b), together with the posterior means of the reconstructions of these images for this sample, $\mathbf{z}_i E[\mathbf{A}|\mathbf{X}, \mathbf{Z}]$. Similar reconstructions are obtained by averaging over all values of $\mathbf{Z}$ produced by the Markov chain. The reconstructions provided by the model clearly pick out the relevant content of the images, removing the camera noise in the original images.

These applications of the linear-Gaussian latent feature model are intended primarily to demonstrate that this nonparametric Bayesian approach can efficiently learn satisfying representations without requiring the dimensionality of those representations to be fixed a priori. The data set consisting of images of objects was constructed in a way that removes many of the basic challenges of computer vision, with objects appearing in fixed orientations and locations. Dealing with these issues requires using a more sophisticated image representation or a more complex likelihood function than the linear-Gaussian model. Despite its simplicity, the example of identifying the objects in images illustrates the kind of problems for which the IBP provides an appropriate prior. We describe a range of other applications of the Indian buffet process in detail in the next section.

6. Further Applications and Alternative Inference Algorithms

We now outline six applications of the Indian buffet process, each of which uses the same prior over infinite binary matrices, $P(\mathbf{Z})$, but different choices for the likelihood relating such matrices to observed data. These applications provide an indication of the potential uses of the IBP in machine learning, and have also led to a number of alternative inference algorithms, which we will describe briefly.

6.1 Choice Behavior

Choice behavior refers to our ability to decide between several options. Models of choice behavior are of interest to psychology, marketing, decision theory, and computer science. Our choices are often governed by features of the different options. For example, when choosing which car to buy, one may be influenced by fuel efficiency, cost, size, make, etc. Görür et al. (2006) present a nonparametric Bayesian model based on the IBP which, given the choice data, infers latent features of the options and the corresponding weights of these features. The likelihood function is taken from Tversky's (1972) classic "elimination by aspects" model of choice, with the probability of choosing option A over option B being proportional to the sum of the weights of the distinctive features of A. The IBP is the prior over these latent features, which are assumed to be either present or absent.

The likelihood function used in this model does not have a natural conjugate prior, meaning that the approach taken in our Gibbs sampling algorithm—integrating out the parameters associated with the features—cannot be used. This led Görür et al. to develop a similar Markov chain Monte Carlo algorithm for use with a non-conjugate prior. The basic idea behind the algorithm is analogous to Algorithm 8 of Neal (2000) for Dirichlet process mixture models, using a set of auxiliary variables to represent the weights associated with features that are currently not possessed by any of the available options. These auxiliary variables effectively provide a Monte Carlo approximation to the sum over parameters used in our Gibbs sampler (although there is no approximation error introduced through this step).

6.2 Modeling Protein Interactions

Proteomics aims to understand the functional interactions of proteins, and is a field of growing importance to modern biology and medicine. One of the key concepts in proteomics is a *protein complex*, a group of several interacting proteins. Protein complexes can be experimentally determined by doing high-throughput protein-protein interaction screens. Typically the results of such experiments are subjected to mixture-model based clustering methods. However, a protein can belong to multiple complexes at the same time, making the mixture model assumption invalid. Chu et al. (2006) proposed a nonparametric Bayesian approach based on the IBP for identifying protein complexes and their constituents from interaction screens. The latent binary feature z_{ik} indicates whether protein i belongs to complex k. The likelihood function captures the probability that two proteins will be observed to bind in the interaction screen as a function of how many complexes they both belong to, $\sum_{k=1}^{\infty} z_{ik} z_{jk}$. The approach automatically infers the number of significant complexes from the data and the results are validated using affinity purification/mass spectrometry experimental data from yeast RNA-processing complexes.

6.3 Binary Matrix Factorization for Modeling Dyadic Data

Many interesting data sets are *dyadic*: there are two sets of objects or entities and observations are made on pairs with one element from each set. For example, the two sets might consist of movies and viewers, and the observations are ratings given by viewers to movies. Alternatively, the two sets might be genes and biological tissues and the observations may be expression levels for particular genes in different tissues. Dyadic data can often be represented as matrices and many models of dyadic data can be expressed in terms of matrix factorization. Models of dyadic data make it possible to predict, for example, the ratings a viewer might give to a movie based on ratings from other viewers, a task known as *collaborative filtering*. A traditional approach to modeling dyadic data is *bi-clustering*: simultaneously clustering both the rows (e.g., viewers) and the columns (e.g., movies) of the observation matrix using coupled mixture models. However, as we have discussed, mixture models provide a very limited latent variable representation of data. Meeds et al. (2007) presented a more expressive model of dyadic data based on the two-parameter version of the Indian buffet process. In this model, both movies and viewers are represented by binary latent vectors with an unbounded number of elements, corresponding to the features they might possess (e.g., “likes horror movies”). The two corresponding infinite binary matrices interact via a real-valued weight matrix which links features of movies to features of viewers, resulting in a binary matrix factorization of the observed ratings.

The basic inference algorithm used in this model was similar to the non-conjugate version of the Gibbs sampler outlined above, but the authors also developed a number of novel Metropolis-Hastings proposals that are mixed with the steps of the Gibbs sampler. One proposal directly handles the number of new features associated with each object, facilitating one of the more difficult aspects of non-conjugate inference. Another proposal is a “split-merge” move, analogous to similar proposals used in models based on the CRP (Jain and Neal, 2004; Dahl, 2003). In contrast to the

Gibbs sampler, which slowly affects the number of features used in the model by changing a single feature allocation for a single object at a time, the split-merge proposal explores large-scale moves such as dividing a single feature into two, or collapsing two features together. Combining these large-scale moves with the Gibbs sampler can result in a Markov chain Monte Carlo algorithm that explores the space of latent matrices faster.

6.4 Extracting Features from Similarity Judgments

One of the goals of cognitive psychology is to determine the kinds of representations that underlie people's judgments. In particular, the *additive clustering* method has been used to infer people's beliefs about the features of objects from their judgments of the similarity between them (Shepard and Arabie, 1979). Given a square matrix of judgments of the similarity between N objects, where s_{ij} is the similarity between objects i and j, the additive clustering model seeks to recover a $N \times K$ binary feature matrix $\mathbf{F}$ and a vector of K weights associated with those features such that $s_{ij} \approx \sum_{k=1}^{K} w_k f_{ik} f_{jk}$. A standard problem for this approach is determining the value of K, for which a variety of heuristic methods have been used. Navarro and Griffiths (2007) presented a nonparametric Bayesian solution to this problem, using the IBP to define a prior on $\mathbf{F}$ and assuming that s_{ij} has a Gaussian distribution with mean $\sum_{k=1}^{K_+} w_k f_{ik} f_{jk}$ (following Tenenbaum, 1996). Using this method provides a posterior distribution over the effective dimension of $\mathbf{F}$, K_+, and gives both a weight and a posterior probability for the presence of each feature.

Samples from the posterior distribution over feature matrices reveal some surprisingly rich representations expressed in classic similarity data sets. Performing posterior inference makes it possible to discover that there are multiple sensible sets of features that could account for human similarity judgments, while previous approaches that had focused on finding the single best set of features might only find one such set. For example, the nonparametric Bayesian model reveals that people's similarity judgments for numbers from 0-9 can be accounted for by a set of features that includes both the odd and the even numbers, while previous additive clustering analyses (e.g., Tenenbaum, 1996) had only produced the odd numbers.

The additive clustering model, like the choice model discussed above, is another case in which non-conjugate inference is necessary. In this case, the inference algorithm is rendered simpler by the fact that no attempt is made to model the similarity of an object to itself, s_{ii}. As a consequence, a feature possessed by a single object has no effect on the likelihood, and the number of such features and their associated weights can be drawn directly from the prior. Inference thus proceeds using an algorithm similar to the Gibbs sampler derived above, with the addition of a Metropolis-Hastings step to update the weights associated with each feature.

6.5 Latent Features in Link Prediction

Network data, indicating the relationships among a group of people or objects, have been analyzed by both statisticians and sociologists. A basic goal of these analyses is predicting which unobserved relationships might exist. For example, having observed friendly interactions among several pairs of people, a sociologist might seek to predict which other people are likely to be friends with one another. This problem of link prediction can be solved using a probabilistic model for the structure of graphs. One popular class of models, known as stochastic blockmodels, assume that each entity belongs to a single latent class, and that the probability of a relationship existing between two entities depends only on the classes of those entities (Nowicki and Snijders, 2001; Wang and Wong, 1987). This is analogous to a mixture model, in which the probability that an object has certain observed properties depends only on its latent class. Nonparametric versions of stochastic blockmodels can be defined using the Chinese restaurant process (Kemp et al., 2006), corresponding to an underlying stochastic process that generalizes the Dirichlet process (Roy and Teh, 2009).

Just as allowing objects to have latent features rather than a single latent class makes it possible to go beyond mixture models, this approach allows us to define models for link prediction that are richer than stochastic blockmodels. Miller et al. (2010) defined a class of nonparametric latent feature models that can be used for link prediction. The key idea is to define the probability of the existence of a link between two entities in terms of a "squashing function" (such as the logistic or probit) applied to a real-valued score for that link. The scores then depend on the features of the two entities. For a set of N entities, the pairwise scores are given by the $N \times N$ matrix $\mathbf{ZWZ}^T$, where $\mathbf{Z}$ is a binary feature matrix, as used throughout this paper, and $\mathbf{W}$ is a matrix of real-valued feature weights. Since the feature weights can be positive or negative, features can interact to either increase or decrease the probability of a link. The resulting model is strictly more expressive than a stochastic blockmodel and produces more accurate predictions, particularly in cases where multiple factors interact to influence the existence of a relationship (such as in the decision to co-author a paper, for example).

6.6 Independent Components Analysis and Sparse Factor Analysis

Independent Components Analysis (ICA) is a model which explains observed signals in terms of a linear superposition, or mixing, of independent hidden sources (Comon, 1994; Bell and Sejnowski, 1995; MacKay, 1996; Cardoso, 1998). ICA has been used to solve the problem of "blind source separation" in which the goal is to unmix the hidden sources from the observed mixed signals without assuming much knowledge of the hidden source distribution. This models, for example, a listener in a cocktail party who may want to unmix the signals received on his two ears into the many independent sound sources that produced them. ICA is closely related to factor analysis, except that while in factor analysis the sources are assumed to be Gaussian distributed, in ICA the sources are assumed to have any distribution other than the Gaussian.

One of the key practical problems in ICA is determining the number of hidden sources. Knowles and Ghahramani (2007) provided a solution to this problem by devising a non-parametric Bayesian model for ICA based on the IBP. The basic assumption of this ICA model is that the number of potential sources is unbounded, but that any particular source is typically not present in a given signal. The IBP provides a natural model for determining which sources are present in each signal. In the notation of Section 3, the observed signals are represented by a matrix $\mathbf{X}$, the presence or absence of the hidden sources by the IBP distributed matrix $\mathbf{Z}$, and the value taken by the sources by the matrix $\mathbf{V}$. Knowles and Ghahramani (2007) considered several variants of the model, including ICA models where the elements of $\mathbf{V}$ have Laplacian distributions, sparse FA models where the elements of $\mathbf{V}$ have Gaussian distributions, and one and two parameter versions of the IBP in both cases. The model was applied to discovering gene signatures from gene expression microarray data from an ovarian cancer study.

Rai and Daumé (2009) developed two interesting extensions of this model also motivated by applications to gene expression data. First they considered both factor analysis and factor regression models, where the latter refers to solving a regression problem with a typically large number of input features by making predictions based solely on the factor representation. Second, they used an IBP to model the sparsity in the factor loading matrix (rather than the factor or source matrix in nonparametric ICA) and they moreover assume that the factors are related to each other through a hierarchy. They used Kingman's coalescent as a nonparametric Bayesian model for this hierarchy, following the inference algorithms developed in Teh et al. (2008). This paper shows a nice example of how the IBP can be integrated with other nonparametric Bayesian distributions in a fairly modular manner to solve useful inference problems.

6.7 Bipartite Graphs and Learning Hidden Causes

Wood et al. (2006) used the IBP as part of an algorithm for learning the structure of graphical models. Specifically, they focused on the case where an unknown number of hidden variables (e.g., diseases) are causes for some set of observed variables (e.g., symptoms). Rather than defining a prior over the number of hidden causes, Wood et al. used a non-parametric Bayesian approach based on the IBP to model the structure of graphs with countably infinitely many hidden causes. The binary variable z_{ik} indicates whether hidden variable k has a direct causal influence on observed variable i; in other words whether k is a parent of i in the graph. The data being modeled were the values of the set of observed variables over a number of trials, where each variable was either present or absent on each trial. Each hidden variable could be either present or absent on a particular trial, with the probabilities of these states being determined by a parameter of the model, and hidden variables were assumed to combine via a noisy-OR (Pearl, 1988) to influence the observed variables.

Wood et al. (2006) described an MCMC algorithm for inference in this model. Like many of the cases discussed in this section, this model lacked natural conjugate priors. Inference was done using a variant on the Gibbs sampler introduced above, with additional steps to modify the values of the hidden variables. The sampling step for the introduction of new hidden causes into the graph was facilitated by an analytic result making it possible to sum out the values of the variables associated with those causes in a way that is analogous to summing out the parameters in a conjugate model. However, Wood and Griffiths (2007) developed a sequential Monte Carlo algorithm for use in this model, similar to algorithms that have been developed for use with the CRP (such as Fearnhead, 2004). This algorithm is a form of particle filter, updating the posterior distribution on $\mathbf{Z}$ one row at a time (in this case, as new observed variables are added to the data). The particle filter provides an efficient and straightforward alternative for inference in models that lack conjugate priors, and generalizes naturally to other models using the IBP.

6.8 Structuring Markov Transition Matrices

Discrete Markov processes are widely used in machine learning, as part of hidden Markov models and state-space models. Nonparametric Bayesian methods have been used to define "infinite" versions of these models, allowing the number of states in a hidden Markov model to be unbounded (Beal et al., 2002). An infinite discrete Markov process can be defined by assuming that transitions from each state follow a Chinese restaurant process, with transitions that have been made frequently in the past being more likely in the future. When a new transition is generated, the next state is drawn from a higher-level Chinese restaurant process that is shared across all states. The resulting distribution can also be obtained from a hierarchical Dirichlet process (Teh et al., 2004).

Fox et al. (2010) recently explored another way of defining an infinite discrete Markov process, which allows for more structure in the transition matrix. In this model, it is assumed that each state can only make transitions to a subset of other states. Thus, each state is associated with a binary vector indicating whether or not it makes transitions to other states. With an infinite set of states, a distribution over these vectors can be defined using the IBP. This approach was used to define a nonparametric autoregressive hidden Markov model, in which a sequence of continuous variables were predicted as a linear function of the variables at the previous timestep, but the parameters of the function were determined by a latent Markov process. The resulting model was able to identify meaningful action components in motion capture data. In addition to introducing a novel model, this paper explored the use of "birth and death" moves in the Markov chain Monte Carlo algorithm used for inference, in which entire columns of the matrix produced by the IBP were created or destroyed.

6.9 Other Inference Algorithms

The broad range of settings in which the IBP has been applied have encouraged the development of more efficient methods for probabilistic inference in the resulting nonparametric Bayesian models. As discussed above, several innovations have been used to speed mixing in the Markov chain Monte Carlo algorithms used with specific models. Other work has explored schemes for making inference in the linear-Gaussian model discussed in Section 5 more efficient and scalable to larger data sets. For example, if instead of integrating out the weight matrix $\mathbf{A}$, the posterior distribution over $\mathbf{A}$ is maintained, it is possible to use an alternative sampling scheme that still mixes quickly where the time for each iteration scales linearly in N (Doshi-Velez and Ghahramani, 2009a). This observation also provides the basis for a parallelization scheme in which the features of different objects are computed on different machines, with the potential to make large-scale applications of this linear-Gaussian model possible (Doshi-Velez et al., 2010). Similar principles may apply in the other models using the IBP discussed in this section.

An alternative approach to probabilistic inference is to reject the stochastic approximations provided by MCMC algorithms in favor of deterministic approximations, using variational inference to approximate the posterior. A mean field approximation to the IBP was developed by Doshi-Velez et al. (2009), building on similar approximations for Dirichlet process mixture models (Blei and Jordan, 2006). This variational inference method was applied to the infinite ICA model discussed in Section 6.6, and compared against sampling schemes on both synthetic and real data. The results of these comparisons suggested that the variational approach provides a more efficient strategy for inference in this model when the dimensionality of the observed data is high. Variational inference may thus be useful in working with some of the other models discussed in this section, at least in specific regimes.

7. Extensions and Connections to Other Processes

The Indian buffet process gives a way to characterize our distribution on infinite binary matrices in terms of a simple stochastic process. In this section we review how the IBP can be extended to yield more general classes of distributions, and summarize some of the connections between the IBP and other stochastic processes. Our derivation of the IBP was based on considering the infinite limit of a distribution on finite binary matrices. As with the CRP, this distribution can also be derived via a stick-breaking construction, or by marginalizing out an underlying measure. These different views of the IBP yield different generalizations of the distribution, and different opportunities for developing inference algorithms.

7.1 A Two-Parameter Generalization

As was discussed in Section 4.6, the distribution on the number of features per object and on the total number of features produced by the IBP are directly coupled, through α. This is an undesirable constraint, as the sparsity of a matrix and its dimensionality should be able to vary independently. Ghahramani et al. (2007) introduced a two-parameter generalization of the IBP that separates these two aspects of the distribution.[6] This generalization keeps the average number of features per object at α as before, but allows the overall number of represented features to range from α, an extreme where all features are shared between all objects, to $N\alpha$, an extreme where no features are shared at all. Between these extremes lie many distributions that capture the amount of sharing appropriate for different domains.

As the one-parameter model, this two-parameter model can be derived by taking the limit of a finite model, but using $\pi_k|\alpha,\beta \sim \text{Beta}(\frac{\alpha\beta}{K},\beta)$ instead of Equation 9. Here we will focus on the equivalent sequential generative process. To return to the language of the Indian buffet, the first customer starts at the left of the buffet and samples Poisson(α) dishes. The ith customer serves himself from any dish previously sampled by $m_k > 0$ customers with probability $m_k/(\beta+i-1)$, and in addition from Poisson$(\alpha\beta/(\beta+i-1))$ new dishes. The parameter β is introduced in such a way as to preserve the expected number of features per object, α, but the expected overall number of features is $\alpha\sum_{i=1}^{N}\frac{\beta}{\beta+i-1}$, and the distribution of K_+ is Poisson with this mean. The total number of features used thus increases as β increases. For finite β, the expected number of features increases as $\alpha\beta\ln N$, but if $\beta \gg 1$ the logarithmic regime is preceded by linear growth at small $N < \beta$.

Figure 10 shows three matrices drawn from the two-parameter IBP, all with $\alpha = 10$ but with $\beta = 0.2$, $\beta = 1$, and $\beta = 5$ respectively. Although all three matrices have roughly the same number of non-zero entries, the number of features used varies considerably. At small values of β features become likely to be shared by all objects. At high values of β features are more likely to be specific to particular objects. Further details about the properties of this distribution are provided in Ghahramani et al. (2007).

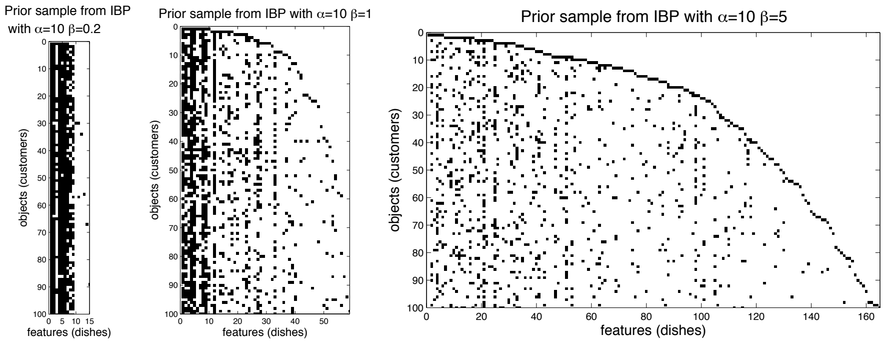


Figure 10: Three samples from the two-parameter Indian buffet process with $\alpha = 10$ and $\beta = 0.2$ (left), $\beta = 1$ (middle), and $\beta = 5$ (right).

7.2 A Stick-Breaking Construction

Our strategy of taking the limit of a finite exchangeable distribution in deriving the IBP was inspired by the derivation of the CRP as the limit of a Dirichlet-multinomial model. However, there are many

6. The original idea and analysis was described in an unpublished note by Sollich (2005).

other routes by which the CRP can be derived. One of these is via the Dirichlet process (Ferguson, 1973). A simple way to think about the Dirichlet process is in terms of a probability measure over probability measures. The parameters of the process are its concentration α and a base measure G_0. In a typical use, we would draw a measure G from the Dirichlet process, and then generate parameters for a model ϕ_i by sampling them independently from G. Since the Dirichlet process generates discrete measures with probability 1, it is possible for multiple parameters ϕ_i and ϕ_j drawn from G to take the same value. We can thus imagine indexing the values taken by the ϕ_i with discrete variables z_i, such that $z_i = z_j$ if and only if $\phi_i = \phi_j$. The z_i thus index unique values of ϕ_i, and correspond to a partition of the indices of the ϕ_i. The distribution over partitions $\mathbf{z}$ produced by the Dirichlet process, integrating over G, is the CRP (Blackwell and MacQueen, 1973).

A straightforward way to understand how the Dirichlet process allocates probabilities to a discrete set of atoms is to think about assigning probabilities in terms of breaking off pieces of a stick. The stick is one unit in length, corresponding to the fact that our probabilities must sum to one. Each piece of stick we break off represents the probability assigned to another discrete atom. After breaking off each piece, we then consider how much of the remainder to break off as the next piece. Sethuraman (1994) showed that if this process is repeated infinitely often, with a proportion of the stick drawn from a Beta$(\alpha, 1)$ distribution being broken off at each step, the lengths of the pieces of broken stick are equivalent to the probabilities assigned to a discrete set of atoms by the Dirichlet process with parameter α. This stick-breaking representation of the Dirichlet process is useful in deriving its properties, and in developing inference algorithms such as the variational inference algorithm proposed by Blei and Jordan (2006).

Teh et al. (2007) showed that a similar stick-breaking construction can be defined for the IBP. First, we imagine sorting the π_k representing the probability of each feature being possessed by an object from largest to smallest. Then, if we consider the proportion of the stick that is broken off and discarded at each break in the stick-breaking construction for the Dirichlet process, the distribution of the sequence of stick lengths corresponds exactly to the distribution of these ordered probabilities. This stick-breaking construction identifies an interesting relationship between the IBP and the Dirichlet process, and is useful for exactly the same reasons. In particular, the stick-breaking construction was used in defining the variational inference algorithm summarized in Section 6.9, and can also be used to derive other inference algorithms for the IBP, such as slice sampling (Teh et al., 2007).

7.3 Connections to the Beta Process

The relationship between the CRP and the Dirichlet process is an instance of a more general relationship between exchangeable distributions and underlying probability measures. The results summarized in the previous paragraph indicate that we can write

$$P(\mathbf{z}) = \int \prod_{i=1}^{N} P(z_i|G)p(G)dG,$$

where the z_i are drawn independently from the measure G, which is generated from the Dirichlet process. The fact that we can represent the exchangeable distribution $P(\mathbf{z})$ as the result of generating the z_i independently from a latent measure is a specific instance of the more general principle stated in de Finetti's exchangeability theorem, which indicates that *any* exchangeable distribution can be represented in this way (see Bernardo and Smith, 1994, for details). This raises a natural question: is there a similar measure underlying the exchangeable distribution produced by the IBP?

Thibaux and Jordan (2007) provided an answer to this question, showing that the exchangeable distribution produced by the IBP corresponds to the use of a latent measure based on the beta process (Hjort, 1990). The beta process provides a source of Bernoulli parameters π_k associated with the elements of a (possibly continuous) index set. Sampling each of the z_{ik} independently according to the distribution defined by the appropriate parameter results in the same distribution on $\mathbf{Z}$ as the IBP. This perspective also makes it straightforward to define analogues of the two-parameter process described in Section 7.1, and to extend the IBP to a hierarchical model that can capture correlations in the features exhibited in multiple data sets. Teh and Görür (2010) also recently used the relationship to the beta process to define a variant of the IBP that produces a power-law distribution in feature frequencies, exploiting a connection to stable processes. Variants of this kind may be useful in settings where power-law distributions are common, such as natural language processing.

7.4 Relaxing the Assumption of Exchangeability

The IBP assumes independence between the columns of $\mathbf{Z}$, and only the kind of weak dependency implied by exchangeability for the rows of $\mathbf{Z}$. Both of these assumptions have been relaxed in subsequent work. Producing correlations between the columns of $\mathbf{Z}$ can be done by supplementing the IBP with a secondary process capturing patterns in the latent features (Doshi-Velez and Ghahramani, 2009b). Modifying the assumption of exchangeability is potentially more problematic. Exchangeability was one of our original desiderata, since it is a reasonable assumption in many settings and simplifies probabilistic inference. However, this assumption is not warranted in cases where we have additional information about the properties of our observations, such as the fact that they were produced in a particular temporal sequence, or reflect a known pattern of correlation. The challenge is thus to identify how the assumption of exchangeability can be relaxed while maintaining the tractability of probabilistic inference. Two recent papers have presented strategies for modifying the IBP to capture different forms of dependency between the rows of $\mathbf{Z}$.

The first kind of dependency can arise as the consequence of observations being generated in a specific sequence. In such a case, it might be appropriate to assume that the latent features associated with observations made closer in time should be more correlated. A strategy for modifying the IBP to capture this kind of dependency was introduced by Van Gael et al. (2009). In this model—the Markov Indian buffet process—it is assumed that the rows of $\mathbf{Z}$ are generated via a Markov process, where the values in each column are generated based on the corresponding values in the previous row. This Markov process has two parameters, giving the probability of a 0 in the previous row changing to a 1, and the probability of a 1 in the previous row remaining unchanged. By assuming that these parameters are generated from a Beta distribution and taking a limit analogous to that used in the derivation of the IBP, it is possible to define a distribution over equivalence classes of binary matrices in which the rows of the matrix reflect a Markov dependency structure. This model can be used to define richer nonparametric models for temporal data, such as an infinite factorial hidden Markov model, and probabilistic inference can be carried out using a slice sampler (see Van Gael et al., 2009, for details).

A second kind of dependency can be the result of known degrees of relatedness among observations. For example, one might seek to draw inferences about a group of people with known genetic relationships, or about a set of organisms or languages with a known evolutionary history. In cases where the degrees of relatedness can be expressed in a tree, the phylogenetic Indian buffet process

(Miller et al., 2008) can be used. In this model, the tree expresses the dependency structure that governs the rows of $\mathbf{Z}$, and each column is generated independently by sampling from a stochastic process defined on the tree. The parameters of the stochastic process are specified in a way that guarantees the total number of columns follows a Poisson distribution, and the original IBP is recovered as the special case where the tree is degenerate, with all branches meeting at the root. Trees can be used to capture a wide range of dependency structures, including partial exchangeability, and probabilistic inference by MCMC remains tractable because belief propagation on the tree can be used to efficiently compute the relevant conditional probabilities.

8. Conclusions and Future Work

The methods that have been used to define infinite latent class models can be extended to models in which objects are represented in terms of a set of latent features, and used to derive distributions on infinite binary matrices that can be used as priors for such models. We used this method to derive a prior that is the infinite limit of a simple distribution on finite binary matrices, and showed that the same distribution can be specified in terms of a simple stochastic process—the Indian buffet process. This distribution satisfies our two desiderata for a prior for infinite latent feature models: objects are exchangeable, and inference remains tractable. When used as a prior in models that represent objects using latent features, this distribution can be used to automatically infer the number of features required to account for observed data. More generally, it can be used as a prior in any setting where a sparse binary matrix with a finite number of rows and infinite number of columns is appropriate, such as estimating the adjacency matrix of a bipartite graph where the size of one class of nodes is unknown.

Recent work has made significant progress on turning this nonparametric approach to inferring latent features into a tool that can be used to solve a wide range of machine learning problems. These advances include more sophisticated MCMC algorithms, schemes for parallelizing probabilistic inference, and deterministic methods for approximating posterior distributions over latent feature matrices. The connections between the IBP and other stochastic processes provide the groundwork for further understanding and extending this class of probabilistic models, making it possible to modify the distribution over feature assignments and to capture different patterns of dependency that might exist among the latent features of objects. As with the CRP, the different views of the IBP that result from considering the stick-breaking construction or the underlying measure that is marginalized out to obtain the combinatorial stochastic process each support different extensions, generalizations, and inference algorithms.

Despite the wide array of successful applications of the IBP and related distributions, we view one of the primary contributions of this work to be the idea that we can define richer nonparametric Bayesian models to suit the unique challenges of machine learning. Our success in transferring the strategy of taking the limit of a finite model from latent classes to latent features suggests that the same strategy might be applied with other representations, broadening the kinds of latent structure that can be recovered through unsupervised learning. This idea receives support both from other examples of new nonparametric models defined via a similar strategy (e.g., Titsias, 2008), and from theoretical analyses of the conditions under which infinite models remain well defined when obtained as limits of finite models (Orbanz, 2010). We anticipate that there will be other combinatorial structures for which this strategy will result in new and useful distributions.

Acknowledgments

This work was presented at the Neural Information Processing Systems conference, and draws on the conference paper (Griffiths and Ghahramani, 2006) and associated technical report (Griffiths and Ghahramani, 2005). The preparation of this article was supported by grants BCS-0631518 and IIS-0845410 from the National Science Foundation, and grant FA-9550-10-1-0232 from the Air Force Office of Scientific Research. We thank three anonymous reviewers for their comments on the manuscript.

Appendix A. Details of Limits

This appendix contains the details of the limits of three expressions that appear in Equations 5 and 14.

The first expression is

$$\begin{aligned}\frac{K!}{K_0!\,K^{K_+}} &= \frac{\prod_{k=1}^{K_+}(K-k+1)}{K^{K_+}}\\ &= \frac{K^{K_+}-\frac{(K_+-1)K_+}{2}K^{K_+-1}+\cdots+(-1)^{K_+-1}(K_+-1)!K}{K^{K_+}}\\ &= 1-\frac{(K_+-1)K_+}{2K}+\cdots+\frac{(-1)^{K_+-1}(K_+-1)!}{K^{K_+-1}}.\end{aligned}$$

For finite K_+, all terms except the first go to zero as $K \to \infty$.

The second expression is

$$\prod_{j=1}^{m_k-1}(j+\tfrac{\alpha}{K}) = (m_k-1)! + \tfrac{\alpha}{K}\sum_{j=1}^{m_k-1}\frac{(m_k-1)!}{j}+\cdots+\left(\tfrac{\alpha}{K}\right)^{m_k-1}.$$

For finite m_k and α, all terms except the first go to zero as $K \to \infty$.

The third expression is

$$\begin{aligned}\left(\frac{N!}{\prod_{j=1}^{N}(j+\frac{\alpha}{K})}\right)^K &= \left(\frac{\prod_{j=1}^{N} j}{\prod_{j=1}^{N}(j+\frac{\alpha}{K})}\right)^K\\ &= \left(\prod_{j=1}^{N}\frac{j}{(j+\frac{\alpha}{K})}\right)^K\\ &= \prod_{j=1}^{N}\left(\frac{1}{1+\frac{\alpha\frac{1}{j}}{K}}\right)^K.\end{aligned} \qquad (27)$$

We can now use the fact that

$$\lim_{K\to\infty}\left(\frac{1}{1+\frac{x}{K}}\right)^K = \exp\{-x\}$$

to compute the limit of Equation 27 as $K \to \infty$, obtaining

$$\begin{aligned}\lim_{K\to\infty}\prod_{j=1}^{N}\left(\frac{1}{1+\frac{\alpha\frac{1}{j}}{K}}\right)^{K} &= \prod_{j=1}^{N}\exp\{-\alpha\tfrac{1}{j}\} \\ &= \exp\{-\alpha\sum_{j=1}^{N}\tfrac{1}{j}\} \\ &= \exp\{-\alpha H_N\},\end{aligned}$$

as desired.

References

D. Aldous. Exchangeability and related topics. In *École d'été de probabilités de Saint-Flour, XIII—1983*, pages 1–198. Springer, Berlin, 1985.

C. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2:1152–1174, 1974.

M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden Markov model. *Machine Learning*, pages 29–245, 2002.

A. J. Bell and T. J. Sejnowski. An information maximisation approach to blind separation and blind deconv olution. *Neural Computation*, 7(6):1129–1159, 1995.

J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. Wiley, New York, 1994.

D. Blackwell and J. MacQueen. Ferguson distributions via Polya urn schemes. *The Annals of Statistics*, 1:353–355, 1973.

D. Blei and M. Jordan. Variational inference for Dirichlet process mixtures. *Journal of Bayesian Analysis*, 1:121–144, 2006.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

C. A. Bush and S. N. MacEachern. A semi-parametric Bayesian model for randomized block designs. *Biometrika*, 83:275–286, 1996.

J.-F. Cardoso. Blind signal separation: statistical principles. *Proceedings of the IEEE*, 86(10): 2009–2025, Oct 1998.

W. Chu, Z. Ghahramani, R. Krause, and D. L. Wild. Identifying protein complexes in high-throughput protein interaction screens using an infinite latent feature model. In *BIOCOMPUTING 2006: Proceedings of the Pacific Symposium*, volume 11, pages 231–242, 2006.

P. Comon. Independent component analysis: A new concept. *Signal Processing*, 36:287–314, 1994.

D. B. Dahl. An improved merge-split sampler for conjugate Dirichlet process mixture models. Technical Report 1086, Department of Statistics, University of Wisconsin, 2003.

A. d'Aspremont, L. El Ghaoui, I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. Technical Report UCB/CSD-04-1330, Computer Science Division, University of California, Berkeley, 2004.

F. Doshi-Velez and Z. Ghahramani. Accelerated Sampling for the Indian Buffet Process. In *International Conference on Machine Learning (ICML 2009)*, 2009a.

F. Doshi-Velez and Z. Ghahramani. Correlated non-parametric latent feature models. In *Proceedings of the Proceedings of the Twenty-Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-09)*, pages 143–150, 2009b.

F. Doshi-Velez, K.T. Miller, J. Van Gael, and Y.W. Teh. Variational Inference for the Indian Buffet Process. In *Artificial Intelligence and Statistics Conference (AISTATS 2009)*, 2009.

F. Doshi-Velez, D. Knowles, S. Mohamed, and Z. Ghahramani. Large scale nonparametric Bayesian inference: Data parallelisation in the Indian buffet process. In *Advances in Neural Information Processing Systems 22*, 2010.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14:11–21, 2004.

T. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1: 209–230, 1973.

T. S. Ferguson. Bayesian density estimation by mixtures of normal distributions. In M. Rizvi, J. Rustagi, and D. Siegmund, editors, *Recent Advances in Statistics*, pages 287–302. Academic Press, New York, 1983.

E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. Sharing features among dynamical systems with beta processes. In *Advances in Neural Information Processing Systems 22*, 2010.

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.

Z. Ghahramani. Factorial learning and the EM algorithm. In *Advances in Neural Information Processing Systems 7*. Morgan Kaufmann, San Francisco, CA, 1995.

Z. Ghahramani, T. L. Griffiths, and P. Sollich. Bayesian nonparametric latent feature models. In *Bayesian Statistics 8*. Oxford University Press, Oxford, 2007.

W.R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, Suffolk, UK, 1996.

D. Görür, F. Jäkel, and C. E. Rasmussen. A choice model with infinitely many latent features. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 361–368, New York, 2006. ACM Press.

P. Green and S. Richardson. Modelling heterogeneity with and without the Dirichlet process. *Scandinavian Journal of Statistics*, 28:355–377, 2001.

T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. Technical Report 2005-001, Gatsby Computational Neuroscience Unit, 2005.

T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. In *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. In *International Conference on Machine Learning (ICML 2005)*, 2005.

N. L. Hjort. Nonparametric Bayes estimators based on Beta processes in models for life history data. *Annals of Statistics*, 18:1259–1294, 1990.

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96:1316–1332, 2001.

S. Jain and R. M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet Process mixture model. *Journal of Computational and Graphical Statistics*, 13:158–182, 2004.

I. T. Jolliffe. *Principal component analysis*. Springer, New York, 1986.

I. T. Jolliffe and M. Uddin. A modified principal component technique based on the lasso. *Journal of Computational and Graphical Statistics*, 12:531–547, 2003.

C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

D. Knowles and Z. Ghahramani. Infinite sparse factor analysis and infinite independent components analysis. In *7th International Conference on Independent Component Analysis and Signal Separation (ICA 2007)*, Lecture Notes in Computer Science Series (LNCS). Springer, 2007.

D. J. C. MacKay. Maximum likelihood and covariant algorithms for independent component analysis. Technical Report Draft 3.7, Cavendish Laboratory, University of Cambridge, Madingley Road, Cambridge CB3 0HE, December 1996.

E. Meeds, Z. Ghahramani, R. Neal, and S. T. Roweis. Modeling dyadic data with binary latent factors. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, Cambridge, MA, 2007. MIT Press.

K. T. Miller, T. L. Griffiths, and M. I. Jordan. The phylogenetic Indian buffet process: A nonexchangeable nonparametric prior for latent features. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI 2008)*, 2008.

K. T. Miller, T. L. Griffiths, and M. I. Jordan. Nonparametric latent feature models for link predictions. In *Advances in Neural Information Processing Systems 22*, 2010.

T. Minka. Bayesian linear regression. Technical report, MIT Media Lab, 2000. http://research.microsoft.com/en-us/um/people/minka/papers/linear.html.

D. J. Navarro and T. L. Griffiths. A nonparametric Bayesian model for inferring features from similarity judgments. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2007. MIT Press.

R. M. Neal. Bayesian mixture modeling. In *Maximum Entropy and Bayesian Methods: Proceedings of the 11th International Workshop on Maximum Entropy and Bayesian Methods of Statistical Analysis*, pages 197–211. Kluwer, Dordrecht, 1992.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

R. M. Neal. Density modeling and clustering using dirichlet diffusion trees. In J. M. Bernardo et al., editor, *Bayesian Statistics 7*, pages 619–629, 2003.

K. Nowicki and T. A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96:1077–1087, 2001.

P. Orbanz. Construction of nonparametric Bayesian models from parametric Bayes equations. In *Advances in Neural Information Processing Systems 22*, 2010.

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

J. Pitman. Combinatorial stochastic processes, 2002. Notes for Saint Flour Summer School.

P. Rai and H. Daumé. The infinite hierarchical factor regression model. In *Advances in Neural Information Processing Systems*, volume 21, 2009.

C. Rasmussen. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems 12*. MIT Press, Cambridge, MA, 2000.

C. E. Rasmussen and Z. Ghahramani. Occam's razor. In *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge, MA, 2001.

S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11:305–345, 1999.

D. M. Roy and Y. W. Teh. The mondrian process. In *Advances in Neural Information Processing Systems 21*, 2009.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

R. Shepard and P. Arabie. Additive clutering: Representation of similarities as combinations of discrete overlapping properties. *Psychological Review*, 86:87–123, 1979.

P. Sollich. Indian buffet process with tunable feature repulsion, 2005.

E. Sudderth, A. Torralba, W. Freeman, and A. Willsky. Describing visual scenes using transformed Dirichlet processes. In *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2004.

Y. W. Teh and D. Görür. Indian buffet processes with power-law behavior. In *Advances in Neural Information Processing Systems 22*, 2010.

Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, San Juan, Puerto Rico, 2007.

Y. W. Teh, H. Daumé, and D. M. Roy. Bayesian agglomerative clustering with coalescents. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

J. B. Tenenbaum. Learning the structure of similarity. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in neural information processing systems 8*, pages 3–9. MIT Press, Cambridge, MA, 1996.

R. Thibaux and M. I. Jordan. Hierarchical Beta processes and the Indian buffet process. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, 2007.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999.

M. Titsias. The infinite gamma-poisson feature model. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

A. Tversky. Elimination by aspects: A theory of choice. *Psychological Review*, 79:281–299, 1972.

N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. In *Advances in Neural Information Processing Systems 15*, Cambridge, 2003. MIT Press.

J. Van Gael, Y.W. Teh, and Z. Ghahramani. The infinite factorial hidden Markov model. In *Advances in Neural Information Processing Systems*, volume 21, 2009.

Y. J. Wang and G. Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82:8–19, 1987.

M. West, P. Muller, and M. Escobar. Hierarchical priors and mixture models, with application in regression and density estimation. In P. Freeman and A. Smith, editors, *Aspects of Uncertainty*, pages 363–386. Wiley, New York, 1994.

F. Wood and T. L. Griffiths. Particle filtering for nonparametric Bayesian matrix factorization. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1513–1520. MIT Press, Cambridge, MA, 2007.

F. Wood, T. L. Griffiths, and Z. Ghahramani. A non-parametric Bayesian method for inferring hidden causes. In *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence (UAI '06)*, 2006.

R. S. Zemel and G. E. Hinton. Developing population codes by minimizing description length. In *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, San Francisco, CA, 1994.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15:262–286, 2006.

Journal of Machine Learning Research 12 (2011) 1225-1248 Submitted 1/11; Published 4/11

DirectLiNGAM: A Direct Method for Learning a Linear Non-Gaussian Structural Equation Model

Shohei Shimizu SSHIMIZU@AR.SANKEN.OSAKA-U.AC.JP
Takanori Inazumi INAZUMI@AR.SANKEN.OSAKA-U.AC.JP
Yasuhiro Sogawa SOGAWA@AR.SANKEN.OSAKA-U.AC.JP
The Institute of Scientific and Industrial Research
Osaka University
Mihogaoka 8-1, Ibaraki, Osaka 567-0047, Japan

Aapo Hyvärinen AAPO.HYVARINEN@HELSINKI.FI
Department of Computer Science and Department of Mathematics and Statistics
University of Helsinki
Helsinki Institute for Information Technology
FIN-00014, Finland

Yoshinobu Kawahara KAWAHARA@AR.SANKEN.OSAKA-U.AC.JP
Takashi Washio WASHIO@AR.SANKEN.OSAKA-U.AC.JP
The Institute of Scientific and Industrial Research
Osaka University
Mihogaoka 8-1, Ibaraki, Osaka 567-0047, Japan

Patrik O. Hoyer PATRIK.HOYER@HELSINKI.FI
Helsinki Institute for Information Technology
University of Helsinki
FIN-00014, Finland

Kenneth Bollen BOLLEN@UNC.EDU
Department of Sociology, CB 3210 Hamilton Hall
University of North Carolina
Chapel Hill, NC 27599-3210
U.S.A.

Abstract

Structural equation models and Bayesian networks have been widely used to analyze causal relations between continuous variables. In such frameworks, linear acyclic models are typically used to model the data-generating process of variables. Recently, it was shown that use of non-Gaussianity identifies the full structure of a linear acyclic model, that is, a causal ordering of variables and their connection strengths, without using any prior knowledge on the network structure, which is not the case with conventional methods. However, existing estimation methods are based on iterative search algorithms and may not converge to a correct solution in a finite number of steps. In this paper, we propose a new direct method to estimate a causal ordering and connection strengths based on non-Gaussianity. In contrast to the previous methods, our algorithm requires no algorithmic parameters and is guaranteed to converge to the right solution within a small fixed number of steps if the data strictly follows the model, that is, if all the model assumptions are met and the sample size is infinite.

Keywords: structural equation models, Bayesian networks, independent component analysis, non-Gaussianity, causal discovery

1. Introduction

Many empirical sciences aim to discover and understand causal mechanisms underlying various natural phenomena and human social behavior. An effective way to study causal relationships is to conduct a controlled experiment. However, performing controlled experiments is often ethically impossible or too expensive in many fields including social sciences (Bollen, 1989), bioinformatics (Rhein and Strimmer, 2007) and neuroinformatics (Londei et al., 2006). Thus, it is necessary and important to develop methods for causal inference based on the data that do not come from such controlled experiments.

Structural equation models (SEM) (Bollen, 1989) and Bayesian networks (BN) (Pearl, 2000; Spirtes et al., 1993) are widely applied to analyze causal relationships in many empirical studies. A *linear* acyclic model that is a special case of SEM and BN is typically used to analyze causal effects between continuous variables. Estimation of the model commonly uses only the covariance structure of the data and in most cases cannot identify the full structure, that is, a causal ordering and connection strengths, of the model with no prior knowledge on the structure (Pearl, 2000; Spirtes et al., 1993).

In Shimizu et al. (2006), a non-Gaussian variant of SEM and BN called a linear non-Gaussian acyclic model (LiNGAM) was proposed, and its full structure was shown to be identifiable without pre-specifying a causal order of the variables. This feature is a significant advantage over the conventional methods (Spirtes et al., 1993; Pearl, 2000). A non-Gaussian method to estimate the new model was also developed in Shimizu et al. (2006) and is closely related to independent component analysis (ICA) (Hyvärinen et al., 2001). In the subsequent studies, the non-Gaussian framework has been extended in various directions for learning a wider variety of SEM and BN (Hoyer et al., 2009; Hyvärinen et al., 2010; Lacerda et al., 2008). In what follows, we refer to the non-Gaussian model as LiNGAM and the estimation method as ICA-LiNGAM algorithm.

Most of major ICA algorithms including Amari (1998) and Hyvärinen (1999) are iterative search methods (Hyvärinen et al., 2001). Therefore, the ICA-LiNGAM algorithms based on the ICA algorithms need some additional information including initial guess and convergence criteria. Gradient-based methods (Amari, 1998) further need step sizes. However, such algorithmic parameters are hard to optimize in a systematic way. Thus, the ICA-based algorithms may get stuck in local optima and may not converge to a reasonable solution if the initial guess is badly chosen (Himberg et al., 2004).

In this paper, we propose a new direct method to estimate a causal ordering of variables in the LiNGAM with no prior knowledge on the structure. The new method estimates a causal order of variables by successively subtracting the effect of each independent component from given data in the model, and this process is completed in steps equal to the number of the variables in the model. It is not based on iterative search in the *parameter space* and needs no initial guess or similar algorithmic parameters. It is *guaranteed* to converge to the right solution within a small fixed number of steps if the data *strictly* follows the model, that is, if all the model assumptions are met and the sample size is infinite. These features of the new method enable more accurate estimation of a causal order of the variables in a disambiguated and direct procedure. Once the causal orders of variables is identified, the connection strengths between the variables are easily estimated using some conventional covariance-based methods such as least squares and maximum likelihood approaches (Bollen, 1989). We also show how prior knowledge on the structure can be incorporated in the new method.

The paper is structured as follows. First, in Section 2, we briefly review LiNGAM and the ICA-based LiNGAM algorithm. We then in Section 3 introduce a new direct method. The performance of the new method is examined by experiments on artificial data in Section 4, and experiments on real-world data in Section 5. Conclusions are given in Section 6. Preliminary results were presented in Shimizu et al. (2009), Inazumi et al. (2010) and Sogawa et al. (2010).

2. Background

In this section, we first review LiNGAM and the ICA-LiNGAM algorithm (Shimizu et al., 2006) in Sections 2.1-2.3 and next mention potential problems of the ICA-based algorithm in Section 2.4.

2.1 A Linear Non-Gaussian Acyclic Model: LiNGAM

In Shimizu et al. (2006), a non-Gaussian variant of SEM and BN, which is called LiNGAM, was proposed. Assume that observed data are generated from a process represented graphically by a directed acyclic graph, that is, DAG. Let us represent this DAG by a $m\times m$ adjacency matrix $\mathbf{B}=\{b_{ij}\}$ where every b_{ij} represents the connection strength from a variable x_j to another x_i in the DAG. Moreover, let us denote by $k(i)$ a causal order of variables x_i in the DAG so that no later variable determines or has a directed path on any earlier variable. (A directed path from x_i to x_j is a sequence of directed edges such that x_j is reachable from x_i.) We further assume that the relations between variables are linear. Without loss of generality, each observed variable x_i is assumed to have zero mean. Then we have

$$x_i = \sum_{k(j)<k(i)} b_{ij}x_j + e_i, \tag{1}$$

where e_i is an external influence. All external influences e_i are continuous random variables having *non-Gaussian* distributions with zero means and non-zero variances, and e_i are independent of each other so that there are no latent confounding variables (Spirtes et al., 1993).

We rewrite the model (1) in a matrix form as follows:

$$\mathbf{x} = \mathbf{Bx} + \mathbf{e}, \tag{2}$$

where $\mathbf{x}$ is a p-dimensional random vector, and $\mathbf{B}$ could be permuted by simultaneous equal row and column permutations to be *strictly* lower triangular due to the acyclicity assumption (Bollen, 1989). Strict lower triangularity is here defined as a lower triangular structure with all zeros on the diagonal. Our goal is to estimate the adjacency matrix $\mathbf{B}$ by observing data $\mathbf{x}$ only. Note that we do *not* assume that the distribution of $\mathbf{x}$ is faithful (Spirtes et al., 1993) to the generating graph.

We note that each b_{ij} represents the direct causal effect of x_j on x_i and each a_{ij}, the (i,j)-th element of the matrix $\mathbf{A}=(\mathbf{I}-\mathbf{B})^{-1}$, the total causal effect of x_j on x_i (Hoyer et al., 2008).

We emphasize that x_i is equal to e_i if no other observed variable x_j ($j\neq i$) inside the model has a directed edge to x_i, that is, all the b_{ij} ($j\neq i$) are zeros. In such a case, an external influence e_i is *observed* as x_i. Such an x_i is called an *exogenous observed* variable. Otherwise, e_i is called an *error*. For example, consider the model defined by

$$\begin{aligned} x_2 &= e_2, \\ x_1 &= 1.5x_2 + e_1, \\ x_3 &= 0.8x_1 - 1.5x_2 + e_3, \end{aligned}$$

where x_2 is equal to e_2 since it is not determined by either x_1 or x_3. Thus, x_2 is an exogenous observed variable, and e_1 and e_3 are errors. Note that there *exists at least one exogenous observed variable* $x_i(=e_i)$ due to the acyclicity and the assumption of no latent confounders.

An exogenous observed variable is usually defined as an observed variable that is determined outside of the model (Bollen, 1989). In other words, an exogenous observed variable is a variable that any other observed variable inside the model does not have a directed edge to. The definition does not require that it is equal to an independent external influence, and the external influences of exogenous observed variables may be dependent. However, in the LiNGAM (2), an exogenous observed variable is always equal to an independent external influence due to the assumption of no latent confounders.

2.2 Identifiability of the Model

We next explain how the connection strengths of the LiNGAM (2) can be identified as shown in Shimizu et al. (2006). Let us first solve Equation (2) for $\mathbf{x}$. Then we obtain

$$\mathbf{x} = \mathbf{A}\mathbf{e}, \tag{3}$$

where $\mathbf{A} = (\mathbf{I} - \mathbf{B})^{-1}$ is a mixing matrix whose elements are called mixing coefficients and can be permuted to be lower triangular as well due to the aforementioned feature of $\mathbf{B}$ and the nature of matrix inversion. Since the components of $\mathbf{e}$ are independent and non-Gaussian, Equation (3) defines the independent component analysis (ICA) model (Hyvärinen et al., 2001), which is known to be identifiable (Comon, 1994; Eriksson and Koivunen, 2004).

ICA essentially can estimate $\mathbf{A}$ (and $\mathbf{W} = \mathbf{A}^{-1} = \mathbf{I} - \mathbf{B}$), but has permutation, scaling and sign indeterminacies. ICA actually gives $\mathbf{W}_{ICA}$=$\mathbf{PDW}$, where $\mathbf{P}$ is an unknown permutation matrix, and $\mathbf{D}$ is an unknown diagonal matrix. But in LiNGAM, the correct permutation matrix $\mathbf{P}$ can be found (Shimizu et al., 2006): the correct $\mathbf{P}$ is the only one that gives no zeros in the diagonal of $\mathbf{DW}$ since $\mathbf{B}$ should be a matrix that can be permuted to be strictly lower triangular and $\mathbf{W} = \mathbf{I} - \mathbf{B}$. Further, one can find the correct scaling and signs of the independent components by using the unity on the diagonal of $\mathbf{W}$=$\mathbf{I}$−$\mathbf{B}$. One only has to divide the rows of $\mathbf{DW}$ by its corresponding diagonal elements to obtain $\mathbf{W}$. Finally, one can compute the connection strength matrix $\mathbf{B} = \mathbf{I} - \mathbf{W}$.

2.3 ICA-LiNGAM Algorithm

The ICA-LiNGAM algorithm presented in Shimizu et al. (2006) is described as follows:

ICA-LiNGAM algorithm

1. Given a p-dimensional random vector $\mathbf{x}$ and its $p \times n$ observed data matrix $\mathbf{X}$, apply an ICA algorithm (FastICA of Hyvärinen 1999 using hyperbolic tangent function) to obtain an estimate of $\mathbf{A}$.

2. Find the unique permutation of rows of $\mathbf{W}$=$\mathbf{A}^{-1}$ which yields a matrix $\widetilde{\mathbf{W}}$ without any zeros on the main diagonal. The permutation is sought by minimizing $\sum_i 1/|\widetilde{\mathbf{W}}_{ii}|$.

3. Divide each row of $\widetilde{\mathbf{W}}$ by its corresponding diagonal element, to yield a new matrix $\widetilde{\mathbf{W}}'$ with all ones on the diagonal.

4. Compute an estimate $\widehat{\mathbf{B}}$ of $\mathbf{B}$ using $\widehat{\mathbf{B}} = \mathbf{I} - \widetilde{\widetilde{\mathbf{W}}}'$.

5. Finally, to estimate a causal order $k(i)$, find the permutation matrix $\widetilde{\mathbf{P}}$ of $\widehat{\mathbf{B}}$ yielding a matrix $\widetilde{\mathbf{B}} = \widetilde{\mathbf{P}}\widehat{\mathbf{B}}\widetilde{\mathbf{P}}^T$ which is as close as possible to a strictly lower triangular structure. The lower-triangularity of $\widetilde{\mathbf{B}}$ can be measured using the sum of squared b_{ij} in its upper triangular part $\sum_{i \leq j} \widetilde{b}_{ij}^2$ for small number of variables, say less than 8. For higher-dimensional data, the following approximate algorithm is used, which sets small absolute valued elements in $\widetilde{\mathbf{B}}$ to zero and tests if the resulting matrix is possible to be permuted to be strictly lower triangular:

   (a) Set the $p(p+1)/2$ smallest (in absolute value) elements of $\widehat{\mathbf{B}}$ to zero.

   (b) Repeat

      i. Test if $\widehat{\mathbf{B}}$ can be permuted to be strictly lower triangular. If the answer is yes, stop and return the permuted $\widehat{\mathbf{B}}$, that is, $\widetilde{\mathbf{B}}$.

      ii. Additionally set the next smallest (in absolute value) element of $\widehat{\mathbf{B}}$ to zero.

2.4 Potential Problems of ICA-LiNGAM

The original ICA-LiNGAM algorithm has several potential problems: i) Most ICA algorithms including FastICA (Hyvärinen, 1999) and gradient-based algorithms (Amari, 1998) may not converge to a correct solution in a finite number of steps if the initially guessed state is badly chosen (Himberg et al., 2004) or if the step size is not suitably selected for those gradient-based methods. The appropriate selection of such algorithmic parameters is not easy. In contrast, our algorithm proposed in the next section is guaranteed to converge to the right solution in a fixed number of steps equal to the number of variables if the data *strictly* follows the model. ii) The permutation algorithms in Steps 2 and 5 are not scale-invariant. Hence they could give a different or *even wrong* ordering of variables depending on scales or standard deviations of variables especially when they have a wide range of scales. However, scales are essentially not relevant to the ordering of variables. Though such bias would vanish for large enough sample sizes, for practical sample sizes, an estimated ordering could be affected when variables are normalized to make unit variance for example, and hence the estimation of a causal ordering becomes quite difficult.

3. A Direct Method: DirectLiNGAM

In this section, we present a new direct estimation algorithm named DirectLiNGAM.

3.1 Identification of an Exogenous Variable Based on Non-Gaussianity and Independence

In this subsection, we present two lemmas and a corollary[1] that ensure the validity of our algorithm proposed in the next subsection 3.2. The basic idea of our method is as follows. We first find an exogenous variable based on its independence of the residuals of a number of pairwise regressions (Lemma 1). Next, we remove the effect of the exogenous variable from the other variables using least squares regression. Then, we show that a LiNGAM also holds for the residuals (Lemma 2) and that the same ordering of the residuals is a causal ordering for the original observed variables as

1. We prove the lemmas and corollary without assuming the faithfulness (Spirtes et al., 1993) unlike our previous work (Shimizu et al., 2009).

well (Corollary 1). Therefore, we can find the second variable in the causal ordering of the original observed variables by analyzing the residuals and their LiNGAM, that is, by applying Lemma 1 to the residuals and finding an "exogenous" residual. The iteration of these effect removal and causal ordering estimates the causal order of the original variables.

We first quote Darmois-Skitovitch theorem (Darmois, 1953; Skitovitch, 1953) since it is used to prove Lemma 1:

Theorem 1 (Darmois-Skitovitch theorem) *Define two random variables y_1 and y_2 as linear combinations of independent random variables $s_i(i=1,\cdots,q)$:*

$$y_1=\sum_{i=1}^{q}\alpha_i s_i,\;\; y_2=\sum_{i=1}^{q}\beta_i s_i.$$

Then, if y_1 and y_2 are independent, all variables s_j for which $\alpha_j\beta_j\neq 0$ are Gaussian. $\square$

In other words, this theorem means that if there exists a non-Gaussian s_j for which $\alpha_j\beta_j\neq 0$, y_1 and y_2 are dependent.

Lemma 1 *Assume that the input data* $\mathbf{x}$ *strictly follows the LiNGAM (2), that is, all the model assumptions are met and the sample size is infinite. Denote by $r_i^{(j)}$ the residual when x_i is regressed on x_j: $r_i^{(j)}=x_i-\frac{\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)}x_j$ $(i\neq j)$. Then a variable x_j is exogenous if and only if x_j is independent of its residuals $r_i^{(j)}$ for all $i\neq j$.* $\square$

Proof (i) Assume that x_j is exogenous, that is, $x_j=e_j$. Due to the model assumption and Equation (3), one can write $x_i=a_{ij}x_j+e_i^{(j)}$ $(i\neq j)$, where $e_i^{(j)}=\sum_{h\neq j}a_{ih}e_h$ and x_j are independent, and a_{ij} is a mixing coefficient from x_j to x_i in Equation (3). The mixing coefficient a_{ij} is equal to the regression coefficient when x_i is regressed on x_j since $\mathrm{cov}(x_i,x_j)=a_{ij}\mathrm{var}(x_j)$. Thus, the residual $r_i^{(j)}$ is equal to the corresponding error term, that is, $r_i^{(j)}=e_i^{(j)}$. This implies that x_j and $r_i^{(j)}(=e_i^{(j)})$ are independent.

(ii) Assume that x_j is not exogenous, that is, x_j has at least one parent. Let P_j denote the (non-empty) set of the variable subscripts of parent variables of x_j. Then one can write $x_j=\sum_{h\in P_j}b_{jh}x_h+e_j$, where x_h and e_j are independent and each b_{jh} is non-zero. Let a vector $\mathbf{x}_{P_j}$ and a column vector $\mathbf{b}_{P_j}$ collect all the variables in P_j and the corresponding connection strengths, respectively. Then, the covariances between $\mathbf{x}_{P_j}$ and x_j are

$$\begin{aligned}E(\mathbf{x}_{P_j}x_j) &= E\{\mathbf{x}_{P_j}(\mathbf{b}_{P_j}^T\mathbf{x}_{P_j}+e_j)\}\\ &= E(\mathbf{x}_{P_j}\mathbf{b}_{P_j}^T\mathbf{x}_{P_j})+E(\mathbf{x}_{P_j}e_j)\\ &= E(\mathbf{x}_{P_j}\mathbf{x}_{P_j}^T)\mathbf{b}_{P_j}. \qquad (4)\end{aligned}$$

The covariance matrix $E(\mathbf{x}_{P_j}\mathbf{x}_{P_j}^T)$ is positive definite since the external influences e_h that correspond to those parent variables x_h in P_j are mutually independent and have positive variances. Thus, the covariance vector $E(\mathbf{x}_{P_j}x_j)=E(\mathbf{x}_{P_j}\mathbf{x}_{P_j}^T)\mathbf{b}_{P_j}$ in Equation (4) cannot equal the zero vector, and there must be at least one variable x_i $(i\in P_j)$ with which x_j covaries, that is, $\mathrm{cov}(x_i,x_j)\neq 0$. Then, for such a variable x_i ($i \in P_j$) that $\mathrm{cov}(x_i,x_j) \neq 0$, we have

$$\begin{aligned}
r_i^{(j)} &= x_i - \frac{\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)} x_j \\
&= x_i - \frac{\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)} \left(\sum_{h \in P_j} b_{jh} x_h + e_j \right) \\
&= \left\{ 1 - \frac{b_{ji}\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)} \right\} x_i - \frac{\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)} \sum_{h \in P_j, h \neq i} b_{jh} x_h \\
&\quad - \frac{\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)} e_j.
\end{aligned}$$

Each of those parent variables x_h (including x_i) in P_j is a linear combination of external influences *other than* e_j due to the relation of x_h to e_j that $x_j = \sum_{h \in P_j} b_{jh} x_h + e_j = \sum_{h \in P_j} b_{jh} \left(\sum_{k(t) \leq k(h)} a_{ht} e_t \right) + e_j$, where e_t and e_j are independent. Thus, the $r_i^{(j)}$ and x_j can be rewritten as linear combinations of independent external influences as follows:

$$\begin{aligned}
r_i^{(j)} &= \left\{ 1 - \frac{b_{ji}\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)} \right\} \left(\sum_{l \neq j} a_{il} e_l \right) - \frac{\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)} \sum_{h \in P_j, h \neq i} b_{jh} \left(\sum_{t \neq j} a_{ht} e_t \right) \\
&\quad - \frac{\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)} e_j, \qquad (5) \\
x_j &= \sum_{h \in P_j} b_{jh} \left(\sum_{t \neq j} a_{ht} e_t \right) + e_j. \qquad (6)
\end{aligned}$$

The first two terms of Equation (5) and the first term of Equation (6) are linear combinations of external influences other than e_j, and the third term of Equation (5) and the second term of Equation (6) depend only on e_j and do not depend on the other external influences. Further, all the external influences including e_j are mutually independent, and the coefficient of non-Gaussian e_j on $r_i^{(j)}$ and that on x_j are non-zero. These imply that $r_i^{(j)}$ and x_j are dependent since $r_i^{(j)}$, x_j and e_j correspond to y_1, y_2, s_j in Darmois-Skitovitch theorem, respectively.

From (i) and (ii), the lemma is proven. ■

Lemma 2 *Assume that the input data* $\mathbf{x}$ *strictly follows the LiNGAM (2). Further, assume that a variable* x_j *is exogenous. Denote by* $\mathbf{r}^{(j)}$ *a (p-1)-dimensional vector that collects the residuals* $r_i^{(j)}$ *when all* x_i *of* $\mathbf{x}$ *are regressed on* x_j $(i \neq j)$. *Then a LiNGAM holds for the residual vector* $\mathbf{r}^{(j)}$: $\mathbf{r}^{(j)} = \mathbf{B}^{(j)}\mathbf{r}^{(j)} + \mathbf{e}^{(j)}$, *where* $\mathbf{B}^{(j)}$ *is a matrix that can be permuted to be strictly lower-triangular by a simultaneous row and column permutation, and elements of* $\mathbf{e}^{(j)}$ *are non-Gaussian and mutually independent.* □

Proof Without loss of generality, assume that $\mathbf{B}$ in the LiNGAM (2) is already permuted to be strictly lower triangular and that $x_j{=}x_1$. Note that $\mathbf{A}$ in Equation (3) is also lower triangular (although its diagonal elements are all ones). Since x_1 is exogenous, a_{i1} are equal to the regression coefficients when x_i are regressed on x_1 $(i \neq 1)$. Therefore, after removing the effects of x_1 from x_i

by least squares estimation, one gets the first column of $\mathbf{A}$ to be a zero vector, and x_1 does not affect the residuals $r_i^{(1)}$. Thus, we again obtain a lower triangular mixing matrix $\mathbf{A}^{(1)}$ with all ones in the diagonal for the residual vector $\mathbf{r}^{(1)}$ and hence have a LiNGAM for the vector $\mathbf{r}^{(1)}$. ■

Corollary 1 *Assume that the input data* $\mathbf{x}$ *strictly follows the LiNGAM (2). Further, assume that a variable* x_j *is exogenous. Denote by* $k_{r^{(j)}}(i)$ *a causal order of* $r_i^{(j)}$. *Recall that* $k(i)$ *denotes a causal order of* x_i. *Then, the same ordering of the residuals is a causal ordering for the original observed variables as well:* $k_{r^{(j)}}(l)<k_{r^{(j)}}(m) \Leftrightarrow k(l)<k(m)$. □

Proof As shown in the proof of Lemma 2, when the effect of an exogenous variable x_1 is removed from the other observed variables, the second to p-th columns of $\mathbf{A}$ remain the same, and the submatrix of $\mathbf{A}$ formed by deleting the first row and the first column is still lower triangular. This shows that the ordering of the other variables is not changed and proves the corollary. ■

Lemma 2 indicates that the LiNGAM for the $(p-1)$-dimensional residual vector $\mathbf{r}^{(j)}$ can be handled as a new input model, and Lemma 1 can be further applied to the model to estimate the next exogenous variable (the next exogenous residual in fact). This process can be repeated until all variables are ordered, and the resulting order of the variable subscripts shows the causal order of the original observed variables according to Corollary 1.

To apply Lemma 1 in practice, we need to use a measure of independence which is not restricted to uncorrelatedness since least squares regression gives residuals always uncorrelated with but not necessarily independent of explanatory variables. A common independence measure between two variables y_1 and y_2 is their mutual information $MI(y_1, y_2)$ (Hyvärinen et al., 2001). In Bach and Jordan (2002), a nonparametric estimator of mutual information was developed using kernel methods.[2] Let K_1 and K_2 represent the Gram matrices whose elements are Gaussian kernel values of the sets of n observations of y_1 and y_2, respectively. The Gaussian kernel values $K_1(y_1^{(i)}, y_1^{(j)})$ and $K_2(y_2^{(i)}, y_2^{(j)})$ $(i, j = 1, \cdots, n)$ are computed by

$$\begin{aligned} K_1(y_1^{(i)}, y_1^{(j)}) &= \exp\left(-\frac{1}{2\sigma^2}\|y_1^{(i)} - y_1^{(j)}\|^2\right), \\ K_2(y_2^{(i)}, y_2^{(j)}) &= \exp\left(-\frac{1}{2\sigma^2}\|y_2^{(i)} - y_2^{(j)}\|^2\right), \end{aligned}$$

where $\sigma>0$ is the bandwidth of Gaussian kernel. Further let κ denote a small positive constant. Then, in Bach and Jordan (2002), the kernel-based estimator of mutual information is defined as:

$$\widehat{MI}_{kernel}(y_1, y_2) = -\frac{1}{2}\log\frac{\det \mathcal{K}_\kappa}{\det \mathcal{D}_\kappa},$$

where

$$\begin{aligned} \mathcal{K}_\kappa &= \begin{bmatrix} \left(K_1 + \frac{n\kappa}{2}I\right)^2 & K_1K_2 \\ K_2K_1 & \left(K_2 + \frac{n\kappa}{2}I\right)^2 \end{bmatrix}, \\ \mathcal{D}_\kappa &= \begin{bmatrix} \left(K_1 + \frac{n\kappa}{2}I\right)^2 & 0 \\ 0 & \left(K_2 + \frac{n\kappa}{2}I\right)^2 \end{bmatrix}. \end{aligned}$$

2. Matlab codes can be downloaded at `http://www.di.ens.fr/~fbach/kernel-ica/index.htm`.

As the bandwidth σ of Gaussian kernel tends to zero, the population counterpart of the estimator converges to the mutual information up to second order when it is expanded around distributions with two variables y_1 and y_2 being independent (Bach and Jordan, 2002). The determinants of the Gram matrices K_1 and K_2 can be efficiently computed by using the incomplete Cholesky decomposition to find their low-rank approximations of rank M ($\ll n$). In Bach and Jordan (2002), it was suggested that the positive constant κ and the width of the Gaussian kernel σ are set to $\kappa = 2\times 10^{-3}$, $\sigma = 1/2$ for $n > 1000$ and $\kappa = 2\times 10^{-2}$, $\sigma = 1$ for $n \leq 1000$ due to some theoretical and computational considerations.

In this paper, we use the kernel-based independence measure. We first evaluate pairwise independence between a variable and each of the residuals and next take the sum of the pairwise measures over the residuals. Let us denote by U the set of the subscripts of variables x_i, that is, $U=\{1, \cdots, p\}$. We use the following statistic to evaluate independence between a variable x_j and its residuals $r_i^{(j)} = x_i - \frac{\mathrm{cov}(x_i,x_j)}{\mathrm{var}(x_j)}x_j$ when x_i is regressed on x_j:

$$T_{kernel}(x_j;U) = \sum_{i\in U, i\neq j} \widehat{MI}_{kernel}(x_j, r_i^{(j)}). \tag{7}$$

Many other nonparametric independence measures (Gretton et al., 2005; Kraskov et al., 2004) and more computationally simple measures that use a single nonlinear correlation (Hyvärinen, 1998) have also been proposed. Any such proposed method of independence could potentially be used instead of the kernel-based measure in Equation (7).

3.2 DirectLiNGAM Algorithm

We now propose a new direct algorithm called DirectLiNGAM to estimate a causal ordering and the connection strengths in the LiNGAM (2):

```
DirectLiNGAM algorithm

  1. Given a $p$-dimensional random vector $\mathbf{x}$, a set of its variable subscripts $U$ and a $p \times n$ data
     matrix of the random vector as $\mathbf{X}$, initialize an ordered list of variables $K := \emptyset$ and $m := 1$.

  2. Repeat until $p-1$ subscripts are appended to $K$:

     (a) Perform least squares regressions of $x_i$ on $x_j$ for all $i \in U\backslash K$ ($i \neq j$) and compute the
         residual vectors $\mathbf{r}^{(j)}$ and the residual data matrix $\mathbf{R}^{(j)}$ from the data matrix $\mathbf{X}$ for all
         $j \in U\backslash K$. Find a variable $x_m$ that is most independent of its residuals:

             $x_m = \arg\min_{j\in U\backslash K} T_{kernel}(x_j; U\backslash K)$,

         where $T_{kernel}$ is the independence measure defined in Equation (7).

     (b) Append $m$ to the end of $K$.

     (c) Let $\mathbf{x} := \mathbf{r}^{(m)}$, $\mathbf{X} := \mathbf{R}^{(m)}$.

  3. Append the remaining variable to the end of $K$.
```

4. Construct a strictly lower triangular matrix $\mathbf{B}$ by following the order in K, and estimate the connection strengths b_{ij} by using some conventional covariance-based regression such as least squares and maximum likelihood approaches on the original random vector $\mathbf{x}$ and the original data matrix $\mathbf{X}$. We use least squares regression in this paper.

3.3 Computational Complexity

Here, we consider the computational complexity of DirectLiNGAM compared with the ICA-LiNGAM with respect to sample size n and number of variables p. A dominant part of DirectLiNGAM is to compute Equation (7) for each x_j in Step 2(a). Since it requires $O(np^2M^2 + p^3M^3)$ operations (Bach and Jordan, 2002) in $p-1$ iterations, complexity of the step is $O(np^3M^2 + p^4M^3)$, where M ($\ll n$) is the maximal rank found by the low-rank decomposition used in the kernel-based independence measure. Another dominant part is the regression to estimate the matrix $\mathbf{B}$ in Step 4. The complexity of many representative regressions including the least square algorithm is $O(np^3)$. Hence, we have a total budget of $O(np^3M^2 + p^4M^3)$. Meanwhile, the ICA-LiNGAM requires $O(p^4)$ time to find a causal order in Step 5. Complexity of an iteration in FastICA procedure at Step 1 is known to be $O(np^2)$. Assuming a constant number C of the iterations in FastICA steps, the complexity of the ICA-LiNGAM is considered to be $O(Cnp^2 + p^4)$. Though general evaluation of the required iteration number C is difficult, it can be conjectured to grow linearly with regards to p. Hence the complexity of the ICA-LiNGAM is presumed to be $O(np^3 + p^4)$.

Thus, the computational cost of DirectLiNGAM would be larger than that of ICA-LiNGAM especially when the low-rank approximation of the Gram matrices is not so efficient, that is, M is large. However, we note the fact that DirectLiNGAM has guaranteed convergence in a fixed number of steps and is of known complexity, whereas for typical ICA algorithms including FastICA, the run-time complexity and the very convergence are not guaranteed.

3.4 Use of Prior Knowledge

Although DirectLiNGAM requires no prior knowledge on the structure, more efficient learning can be achieved if some prior knowledge on a part of the structure is available because then the number of causal orders and connection strengths to be estimated gets smaller.

We present three lemmas to use prior knowledge in DirectLiNGAM. Let us first define a matrix $\mathbf{A}^{\text{knw}}=[a_{ji}^{\text{knw}}]$ that collects prior knowledge under the LiNGAM (2) as follows:

$$a_{ji}^{\text{knw}} \quad := \quad \begin{cases} \;\;0 & \text{if } x_i \text{ does } \textit{not} \text{ have a directed path to } x_j \\ \;\;1 & \text{if } x_i \text{ has a directed path to } x_j \\ -1 & \text{if no prior knowledge is available to know if either} \\ & \;\text{of the two cases above (0 or 1) is true.} \end{cases}$$

Due to the definition of exogenous variables and that of prior knowledge matrix $\mathbf{A}^{\text{knw}}$, we readily obtain the following three lemmas.

Lemma 3 *Assume that the input data* $\mathbf{x}$ *strictly follows the LiNGAM (2). An observed variable x_j is exogenous if a_{ji}^{knw} is zero for all $i \neq j$.*

Lemma 4 *Assume that the input data* $\mathbf{x}$ *strictly follows the LiNGAM (2). An observed variable x_j is endogenous, that is, not exogenous, if there exist such $i \neq j$ that a_{ji}^{knw} is unity.*

Lemma 5 *Assume that the input data* $\mathbf{x}$ *strictly follows the LiNGAM (2). An observed variable* x_j *does not receive the effect of* x_i *if* a_{ji}^{knw} *is zero.*

The principle of making DirectLiNGAM algorithm more accurate and faster based on prior knowledge is as follows. We first find an exogenous variable by applying Lemma 3 instead of Lemma 1 if an exogenous variable is identified based on prior knowledge. Then we do not have to evaluate independence between any observed variable and its residuals. If no exogenous variable is identified based on prior knowledge, we next find endogenous (non-exogenous) variables by applying Lemma 4. Since endogenous variables are never exogenous we can narrow down the search space to find an exogenous variable based on Lemma 1. We can further skip to compute the residual of an observed variable and take the variable itself as the residual if its regressor does not receive the effect of the variable due to Lemma 5. Thus, we can decrease the number of causal orders and connection strengths to be estimated, and it improves the accuracy and computational time. The principle can also be used to further analyze the residuals and find the next exogenous residual because of Corollary 1. To implement these ideas, we only have to replace Step 2a in DirectLiNGAM algorithm by the following steps:

2a-1 Find such a variable(s) x_j ($j \in U \backslash K$) that the j-th row of $\mathbf{A}^{\mathrm{knw}}$ has zero in the i-th column for all $i \in U \backslash K$ ($i \neq j$) and denote the set of such variables by U_{exo}. If U_{exo} is not empty, set $U_c := U_{exo}$. If U_{exo} is empty, find such a variable(s) x_j ($j \in U \backslash K$) that the j-th row of $\mathbf{A}^{\mathrm{knw}}$ has unity in the i-th column for at least one of $i \in U \backslash K$ ($i \neq j$), denote the set of such variables by U_{end} and set $U_c := U \backslash K \backslash U_{end}$.

2a-2 Denote by $V^{(j)}$ a set of such a variable subscript $i \in U \backslash K$ ($i \neq j$) that $a_{ij}^{\mathrm{knw}} = 0$ for all $j \in U_c$. First set $\mathbf{r}_i^{(j)} := \mathbf{x}_i$ for all $i \in V^{(j)}$, next perform least squares regressions of x_i on x_j for all $i \in U \backslash K \backslash V^{(j)}$ ($i \neq j$) and estimate the residual vectors $\mathbf{r}^{(j)}$ and the residual data matrix $\mathbf{R}^{(j)}$ from the data matrix $\mathbf{X}$ for all $j \in U_c$. If U_c has a single variable, set the variable to be x_m. Otherwise, find a variable x_m in U_c that is most independent of the residuals:

$$x_m = \arg\min_{j \in U_c} T_{kernel}(x_j; U \backslash K),$$

where T_{kernel} is the independence measure defined in Equation (7).

4. Simulations

We first randomly generated 5 data sets based on sparse networks under each combination of number of variables p and sample size n (p=10, 20, 50, 100; n=500, 1000, 2000):

1. We constructed the $p \times p$ adjacency matrix with all zeros and replaced every element in the lower-triangular part by independent realizations of Bernoulli random variables with success probability s similarly to Kalisch and Bühlmann (2007). The probability s determines the sparseness of the model. The expected number of adjacent variables of each variable is given by $s(p-1)$. We randomly set the sparseness s so that the number of adjacent variables was 2 or 5 (Kalisch and Bühlmann, 2007).

2. We replaced each non-zero (unity) entry in the adjacency matrix by a value randomly chosen from the interval $[-1.5, -0.5] \cup [0.5, 1.5]$ and selected variances of the external influences

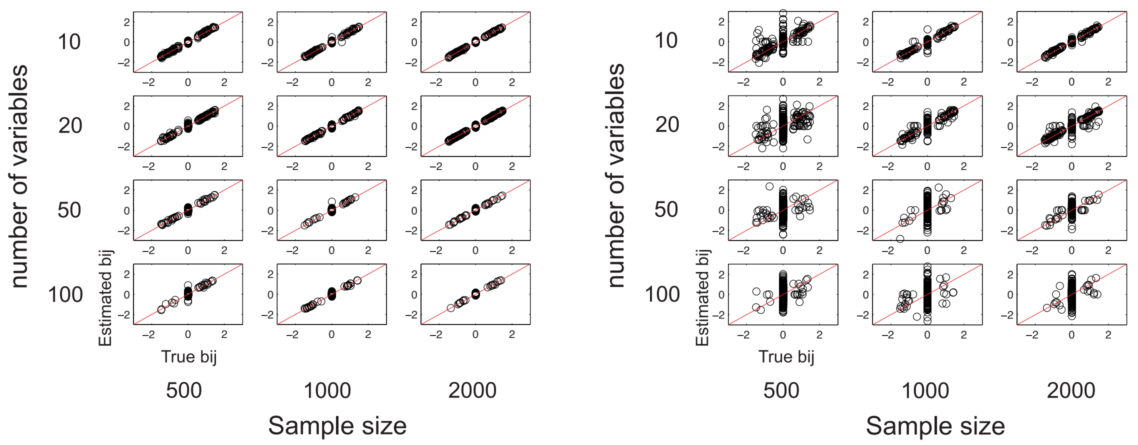


Figure 1: Left: Scatterplots of the estimated b_{ij} by DirectLiNGAM versus the true values for *sparse* networks. Right: Scatterplots of the estimated b_{ij} by ICA-LiNGAM versus the true values for *sparse* networks.

e_i from the interval $[1,3]$ as in Silva et al. (2006). We used the resulting matrix as the data-generating adjacency matrix $\mathbf{B}$.

3. We generated data with sample size n by independently drawing the external influence variables e_i from various 18 non-Gaussian distributions used in Bach and Jordan (2002) including (a) Student with 3 degrees of freedom; (b) double exponential; (c) uniform; (d) Student with 5 degrees of freedom; (e) exponential; (f) mixture of two double exponentials; (g)-(h)-(i) symmetric mixtures of two Gaussians: multimodal, transitional and unimodal; (j)-(k)-(l) nonsymmetric mixtures of two Gaussians, multimodal, transitional and unimodal; (m)-(n)-(o) symmetric mixtures of four Gaussians: multimodal, transitional and unimodal; (p)-(q)-(r) nonsymmetric mixtures of four Gaussians: multimodal, transitional and unimodal. See Figure 5 of Bach and Jordan (2002) for the shapes of the probability density functions.

4. The values of the observed variables x_i were generated according to the LiNGAM (2). Finally, we randomly permuted the order of x_i.

Further we similarly generated 5 data sets based on dense (full) networks, that is, full DAGs with every pair of variables is connected by a directed edge, under each combination of number of variables p and sample size n. Then we tested DirectLiNGAM and ICA-LiNGAM on the data sets generated by sparse networks or dense (full) networks. For ICA-LiNGAM, the maximum number of iterations was taken as 1000 (Shimizu et al., 2006). The experiments were conducted on a standard PC using Matlab 7.9. Matlab implementations of the two methods are available on the web:
DirectLiNGAM: `http://www.ar.sanken.osaka-u.ac.jp/~inazumi/dlingam.html`,
ICA-LiNGAM: `http://www.cs.helsinki.fi/group/neuroinf/lingam/`.

We computed the distance between the true $\mathbf{B}$ and ones estimated by DirectLiNGAM and ICA-LiNGAM using the Frobenius norm defined as

$$\sqrt{\mathrm{trace}\{(\mathbf{B}_{true}-\widehat{\mathbf{B}})^T(\mathbf{B}_{true}-\widehat{\mathbf{B}})\}}.$$

Sparse networks		Sample size		
		500	1000	2000
DirectLiNGAM	dim. = 10	0.48	0.31	0.21
	dim. = 20	1.19	0.70	0.50
	dim. = 50	2.57	1.82	1.40
	dim. = 100	5.75	4.61	2.35
ICA-LiNGAM	dim. = 10	3.01	0.74	0.65
	dim. = 20	9.68	3.00	2.06
	dim. = 50	20.61	20.23	12.91
	dim. = 100	40.77	43.74	36.52
DirectLiNGAM with prior knowledge (50%)	dim. = 10	0.48	0.30	0.24
	dim. = 20	1.00	0.71	0.49
	dim. = 50	2.47	1.75	1.19
	dim. = 100	4.94	3.89	2.27

Dense (full) networks		Sample size		
		500	1000	2000
DirectLiNGAM	dim. = 10	0.45	0.46	0.20
	dim. = 20	1.46	1.53	1.12
	dim. = 50	4.40	4.57	3.86
	dim. = 100	7.38	6.81	6.19
ICA-LiNGAM	dim. = 10	1.71	2.08	0.39
	dim. = 20	6.70	3.38	1.88
	dim. = 50	17.28	16.66	12.05
	dim. = 100	34.95	34.02	32.02
DirectLiNGAM with prior knowledge (50%)	dim. = 10	0.45	0.31	0.19
	dim. = 20	0.84	0.90	0.41
	dim. = 50	2.48	1.86	1.56
	dim. = 100	4.67	3.60	2.61

Table 1: Median distances (Frobenius norms) between true **B** and estimated **B** of DirectLiNGAM and ICA-LiNGAM with five replications.

Tables 1 and 2 show the median distances (Frobenius norms) and median computational times (CPU times), respectively. In Table 1, DirectLiNGAM was better in distances of **B** and gave more accurate estimates of **B** than ICA-LiNGAM for all of the conditions. In Table 2, the computation amount of DirectLiNGAM was rather larger than ICA-LiNGAM when the sample size was increased. A main bottleneck of computation was the kernel-based independence measure. However, its computation amount can be considered to be still tractable. In fact, the actual elapsed times were approximately one-quarter of their CPU times respectively probably because the CPU had four cores. Interestingly, the CPU time of ICA-LiNGAM actually decreased with increased sample size in some cases. This is presumably due to better convergence properties.

To visualize the estimation results, Figures 1 and 2 give combined scatterplots of the estimated elements of **B** of DirectLiNGAM and ICA-LiNGAM versus the true ones for sparse networks and

Sparse networks		Sample size		
		500	1000	2000
DirectLiNGAM	dim. = 10	15.16 sec.	37.21 sec.	66.75 sec.
	dim. = 20	1.56 min.	5.75 min.	17.22 min.
	dim. = 50	16.25 min.	1.34 hrs.	2.70 hrs.
	dim. = 100	2.35 hrs.	21.17 hrs.	19.90 hrs.
ICA-LiNGAM	dim. = 10	0.73 sec.	0.41 sec.	0.28 sec.
	dim. = 20	5.40 sec.	2.45 sec.	1.14 sec.
	dim. = 50	14.49 sec.	21.47 sec.	32.03 sec.
	dim. = 100	46.32 sec.	58.02 sec.	1.16 min.
DirectLiNGAM with	dim. = 10	4.13 sec.	17.75 sec.	30.95 sec.
prior knowledge (50%)	dim. = 20	28.02 sec.	1.64 min.	4.98 min.
	dim. = 50	7.62 min.	28.89 min.	1.09 hrs.
	dim. = 100	48.28 min.	1.84 hrs.	7.51 hrs.

Dense (full) networks		Sample size		
		500	1000	2000
DirectLiNGAM	dim. = 10	8.05 sec.	24.52 sec.	49.44 sec.
	dim. = 20	1.00 min.	4.23 min.	6.91 min.
	dim. = 50	16.18 min.	1.12 hrs.	1.92 hrs.
	dim. = 100	2.16 hrs.	8.59 hrs.	17.24 hrs.
ICA-LiNGAM	dim. = 10	0.97 sec.	0.34 sec.	0.27 sec.
	dim. = 20	5.35 sec.	1.25 sec.	4.07 sec.
	dim. = 50	15.58 sec.	21.01 sec.	31.57 sec.
	dim. = 100	47.60 sec.	56.57 sec.	1.36 min.
DirectLiNGAM with	dim. = 10	2.67 sec.	5.66 sec.	12.31 sec.
prior knowledge (50%)	dim. = 20	5.02 sec.	31.70 sec.	38.35 sec.
	dim. = 50	46.74 sec.	2.89 min.	5.00 min.
	dim. = 100	3.19 min.	10.44 min.	19.80 min.

Table 2: Median computational times (CPU times) of DirectLiNGAM and ICA-LiNGAM with five replications.

dense (full) networks, respectively. The different plots correspond to different numbers of variables and different sample sizes, where each plot combines the data for different adjacency matrices $\mathbf{B}$ and 18 different distributions of the external influences $p(e_i)$. We can see that DirectLiNGAM worked well and better than ICA-LiNGAM, as evidenced by the grouping of the data points onto the main diagonal.

Finally, we generated data sets in the same manner as above and gave some prior knowledge to DirectLiNGAM by creating prior knowledge matrices $\mathbf{A}^{\mathrm{knw}}$ as follows. We first replaced every non-zero element by unity and every diagonal element by zero in $\mathbf{A}=(\mathbf{I}-\mathbf{B})^{-1}$ and subsequently hid each of the off-diagonal elements, that is, replaced it by -1, with probability 0.5. The bottoms of Tables 1 and 2 show the median distances and median computational times. It was empirically confirmed that use of prior knowledge gave more accurate estimates and less computational times

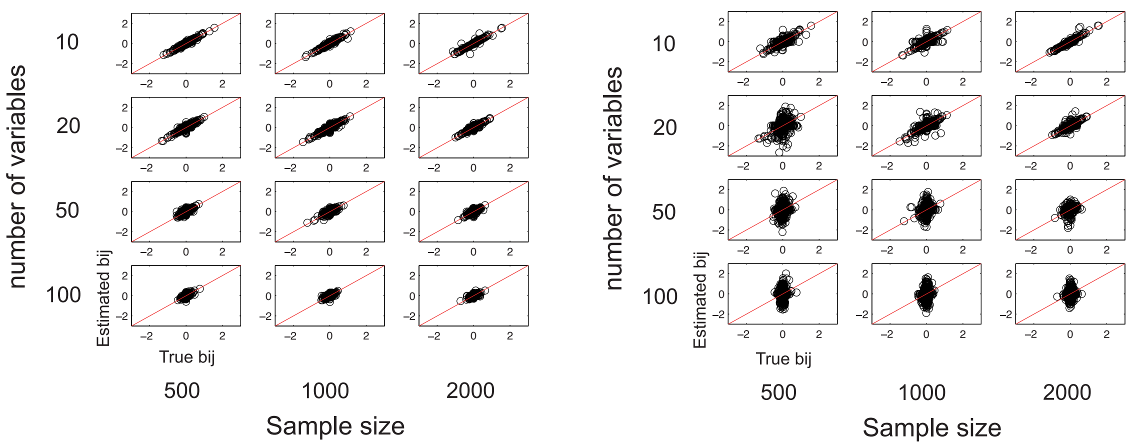


Figure 2: Left: Scatterplots of the estimated b_{ij} by DirectLiNGAM versus the true values for *dense* (full) networks. Right: Scatterplots of the estimated b_{ij} by ICA-LiNGAM versus the true values for *dense* (full) networks.

in most cases especially for dense (full) networks. The reason would probably be that for dense (full) networks more prior knowledge about where directed paths exist were likely to be given and it narrowed down the search space more efficiently.

5. Applications to Real-world Data

We here apply DirectLiNGAM and ICA-LiNGAM on real-world physics and sociology data. Both DirectLiNGAM and ICA-LiNGAM estimate a causal ordering of variables and provide a full DAG. Then we have two options to do further analysis (Hyvärinen et al., 2010): i) Find significant directed edges or direct causal effects b_{ij} and significant total causal effects a_{ij} with $\mathbf{A}=(\mathbf{I}-\mathbf{B})^{-1}$; ii) Estimate redundant directed edges to find the underlying DAG. We demonstrate an example of the former in Section 5.1 and that of the latter in Section 5.2.

5.1 Application to Physical Data

We applied DirectLiNGAM and ICA-LiNGAM on a data set created from a physical system called a double-pendulum, a pendulum with another pendulum attached to its end (Meirovitch, 1986) as in Figure 3. The data set was first used in Kawahara et al. (2011). The raw data consisted of four time series provided by Ibaraki University (Japan) filming the pendulum system with a high-speed video camera at every 0.01 second for 20.3 seconds and then reading out the position using an image analysis software. The four variables were θ_1: the angle between the top limb and the vertical, θ_2: the angle between the bottom limb and the vertical, ω_1: the angular speed of θ_1 or $\dot{\theta}_1$ and ω_2: the angular speed of θ_2 or $\dot{\theta}_2$. The number of time points was 2035. The data set is available on the web: `http://www.ar.sanken.osaka-u.ac.jp/~inazumi/data/furiko.html`.

In Kawahara et al. (2011), some theoretical considerations based on the domain knowledge implied that the angle speeds ω_1 and ω_2 are mainly determined by the angles θ_1 and θ_2 in both cases where the swing of the pendulum is sufficiently small ($\theta_1, \theta_2 \approx 0$) and where the swing is not

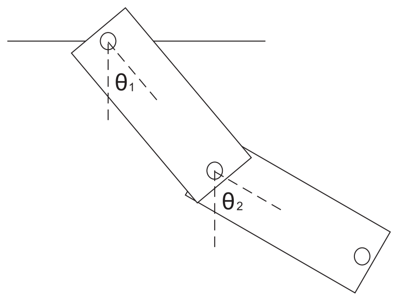


Figure 3: Abstract model of the double-pendulum used in Kawahara et al. (2011).

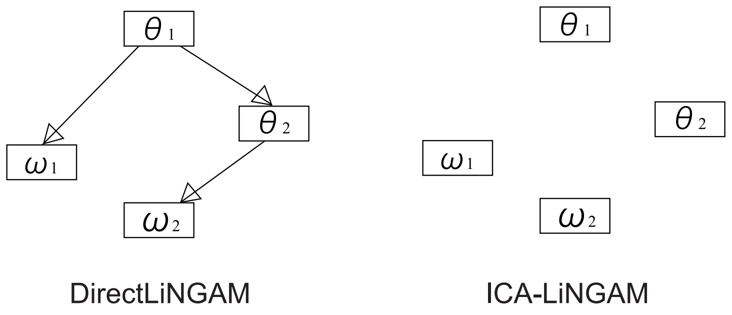


Figure 4: Left: The estimated network by DirectLiNGAM. Only significant directed edges are shown with 5% significance level. Right: The estimated network by ICA-LiNGAM. No significant directed edges were found with 5% significance level.

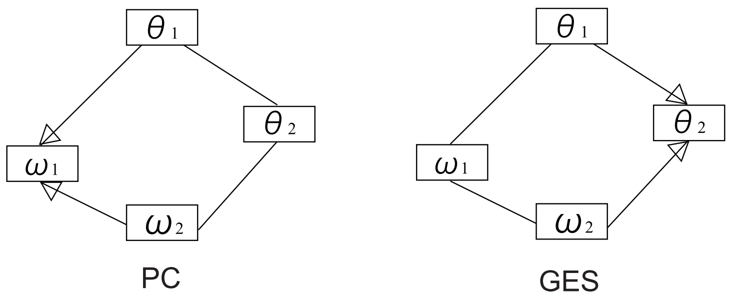


Figure 5: Left: The estimated network by PC algorithm with 5% significance level. Right: The estimated network by GES. An undirected edge between two variables means that there is a directed edge from a variable to the other or the reverse.

very small. Further, in practice, it was reasonable to assume that there were no latent confounders (Kawahara et al., 2011).

As a preprocessing, we first removed the time dependency from the raw data using the ARMA (AutoRegressive Moving Average) model with 2 autoregressive terms and 5 moving average terms following Kawahara et al. (2011). Then we applied DirectLiNGAM and ICA-LiNGAM on the preprocessed data. The estimated adjacency matrices **B** of $\theta_1, \theta_2, \omega_1$ and ω_2 were as follows:

$$\text{DirectLiNGAM} \quad : \quad \begin{array}{c|cccc} & \theta_1 & \theta_2 & \omega_1 & \omega_2 \\ \hline \theta_1 & 0 & 0 & 0 & 0 \\ \theta_2 & -0.23 & 0 & 0 & 0 \\ \omega_1 & 90.39 & -2.88 & 0 & 0 \\ \omega_2 & 5.65 & 94.64 & -0.11 & 0 \end{array},$$

$$\text{ICA} - \text{LiNGAM} \quad : \quad \begin{array}{c|cccc} & \theta_1 & \theta_2 & \omega_1 & \omega_2 \\ \hline \theta_1 & 0 & 0 & 0 & 0 \\ \theta_2 & 1.45 & 0 & 0 & 0 \\ \omega_1 & 108.82 & -52.73 & 0 & 0 \\ \omega_2 & 216.26 & 112.50 & -1.89 & 0 \end{array}.$$

The estimated orderings by DirectLiNGAM and ICA-LiNGAM were identical, but the estimated connection strengths were very different. We further computed their 95% confidence intervals by using bootstrapping (Efron and Tibshirani, 1993) with the number of bootstrap replicates 10000. The estimated networks by DirectLiNGAM and ICA-LiNGAM are graphically shown in Figure 4, where only significant directed edges (direct causal effects) b_{ij} are shown with 5% significance level.[3] DirectLiNGAM found that the angle speeds ω_1 and ω_2 were determined by the angles θ_1 or θ_2, which was consistent with the domain knowledge. Though the directed edge from θ_1 to θ_2 might be a bit difficult to interpret, the effect of θ_1 on θ_2 was estimated to be negligible since the coefficient of determination (Bollen, 1989) of θ_2, that is, $1-\text{var}(\hat{e}_2)/\text{var}(\hat{\theta}_2)$, was very small and was 0.01. (The coefficient of determination of ω_1 and that of ω_2 were 0.46 and 0.49, respectively.) On the other hand, ICA-LiNGAM could not find any significant directed edges since it gave very different estimates for different bootstrap samples.

For further comparison, we also tested two conventional methods (Spirtes and Glymour, 1991; Chickering, 2002) based on conditional independences. Figure 5 shows the estimated networks by PC algorithm (Spirtes and Glymour, 1991) with 5% significance level and GES (Chickering, 2002) with the Gaussianity assumption. We used the Tetrad IV[4] to run the two methods. PC algorithm found the same directed edge from θ_1 on ω_1 as DirectLiNGAM did, but did not found the directed edge from θ_2 on ω_2. GES found the same directed edge from θ_1 on θ_2 as DirectLiNGAM did, but did not find that the angle speeds ω_1 and ω_2 were determined by the angles θ_1 or θ_2.

We also computed the 95% confidence intervals of the total causal effects a_{ij} using bootstrap. DirectLiNGAM found significant total causal effects from θ_1 on θ_2, from θ_1 on ω_1, from θ_1 on ω_2, from θ_2 on ω_1, and from θ_2 on ω_2. These significant total effects would also be reasonable based on similar arguments. ICA-LiNGAM only found a significant total causal effect from θ_2 on ω_2.

Overall, although the four variables θ_1, θ_2, ω_1 and ω_2 are likely to be nonlinearly related according to the domain knowledge (Meirovitch, 1986; Kawahara et al., 2011), DirectLiNGAM gave interesting results in this example.

5.2 Application to Sociology Data

We analyzed a data set taken from a sociological data repository on the Internet called General Social Survey (`http://www.norc.org/GSS+Website/`). The data consisted of six observed vari-

3. The issue of multiple comparisons arises in this context, which we would like to study in future work.
4. Tetrad IV is available at `http://www.phil.cmu.edu/projects/tetrad/`.

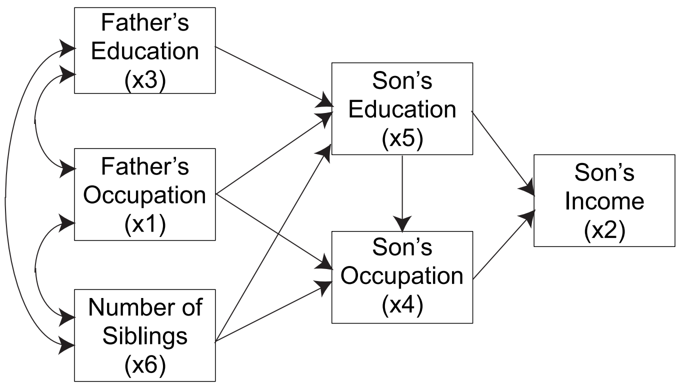


Figure 6: Status attainment model based on domain knowledge (Duncan et al., 1972). A directed edge between two variables in the figure means that there could be a directed edge between the two. A bi-directed edge between two variables means that the relation is not modeled. For instance, there could be latent confounders between the two, there could be a directed edge between the two, or the two could be independent.

ables, x_1: father's occupation level, x_2: son's income, x_3: father's education, x_4: son's occupation level, x_5: son's education, x_6: number of siblings. (x_6 is discrete but is relatively close to be continuous since it is an ordinal scale with many points.) The sample selection was conducted based on the following criteria: i) non-farm background; ii) ages 35 to 44; iii) white; iv) male; v) in the labor force at the time of the survey; vi) not missing data for any of the covariates; vii) years 1972-2006. The sample size was 1380. Figure 6 shows domain knowledge about their causal relations (Duncan et al., 1972). As shown in the figure, there could be some latent confounders between x_1 and x_3, x_1 and x_6, or x_3 and x_6. An objective of this example was to see how our method behaves when such a model assumption of LiNGAM could be violated that there is no latent confounder.

The estimated adjacency matrices **B** by DirectLiNGAM and ICA-LiNGAM were as follows:

$$\text{DirectLiNGAM}: \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \left(\begin{array}{cccccc} 0 & 0 & 3.19 & 0.10 & 0.41 & 0.21 \\ 33.48 & 0 & 452.84 & 422.87 & 1645.45 & 347.96 \\ 0 & 0 & 0 & 0 & 0.55 & -0.18 \\ 0 & 0 & 0.17 & 0 & 4.61 & -0.19 \\ 0 & 0 & 0 & 0 & 0 & -0.12 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array},$$

$$\text{ICA} - \text{LiNGAM}: \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \left(\begin{array}{cccccc} 0 & 0 & 0.93 & 0 & -0.68 & -0.20 \\ 50.70 & 0 & -31.82 & 200.84 & 65.63 & 336.04 \\ 0 & 0 & 0 & 0 & 0.24 & -0.27 \\ 0.17 & 0 & -0.40 & 0 & -0.14 & -0.14 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.08 & 0 \end{array}\right) \end{array}.$$

We subsequently pruned redundant directed edges b_{ij} in the full DAGs by repeatedly applying a sparse method called Adaptive Lasso (Zou, 2006) on each variable and its potential parents. See Appendix A for some more details of Adaptive Lasso. We used a matlab implementation in Sjöstrand (2005) to run the Lasso. Then we obtained the following pruned adjacency matrices $\mathbf{B}$:

$$\text{DirectLiNGAM} \; : \; \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 3.19 & 0 & 0 & 0 \\ 0 & 0 & 0 & 422.87 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.55 & 0 \\ 0 & 0 & 0 & 0 & 4.61 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.12 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array},$$

$$\text{ICA} - \text{LiNGAM} \; : \; \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0.93 & 0 & 0 & 0 \\ 0 & 0 & 0 & 200.84 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.24 & 0 \\ 0 & 0 & 0 & 0 & -0.14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.08 & 0 \end{pmatrix} \end{array}.$$

The estimated networks by DirectLiNGAM and ICA-LiNGAM are graphically shown in Figure 7 and Figure 8, respectively. All the directed edges estimated by DirectLiNGAM were reasonable to the domain knowledge other than the directed edge from x_5: son's education to x_3: father's education. Since the sample size was large and yet the estimated model was not fully correct, the mistake on the directed edge between x_5 and x_3 might imply that some model assumptions might be more or less violated in the data. ICA-LiNGAM gave a similar estimated network but did one more mistake that x_6: number of siblings is determined by x_5: son's education.

Further, Figure 9 and Figure 10 show the estimated networks by PC algorithm with 5% significance level and GES with the Gaussianity assumption. Both of the conventional methods did not find the directions of many edges. The two conventional methods found a reasonable direction of the edge between x_1: father's occupation and x_3: father's education, but they gave a wrong direction of the edge between x_1: father's occupation and x_4: son's occupation.

6. Conclusion

We presented a new estimation algorithm for the LiNGAM that has guaranteed convergence to the right solution in a fixed number of steps if the data strictly follows the model, that is, if all the model assumptions are met and the sample size is infinite. Further, the new algorithm has known computational complexity. This is the first algorithm specialized to estimate the LiNGAM. Simulations implied that the new method often provides better statistical performance than a state of the art method based on ICA. In real-world applications to physics and sociology, interesting results were obtained. Future works would include i) assessment of practical performance of statistical tests to detect violations of the model assumptions including tests of independence (Gretton and Györfi, 2010); ii) implementation issues of our algorithm to improve the practical computational efficiency; iii) extensions of our algorithm to more general cases including the cases with latent confounders

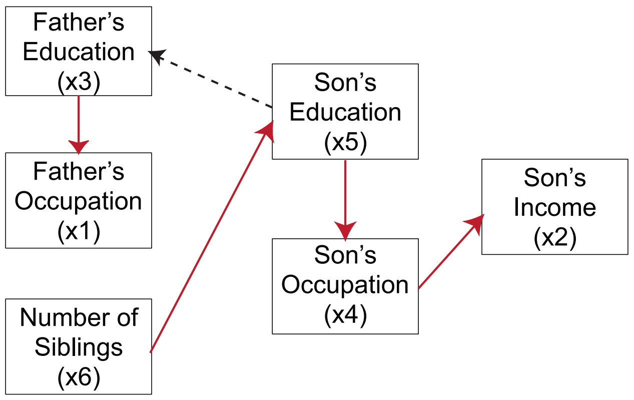


Figure 7: The estimated network by DirectLiNGAM and Adaptive Lasso. A red solid directed edge is reasonable to the domain knowledge.

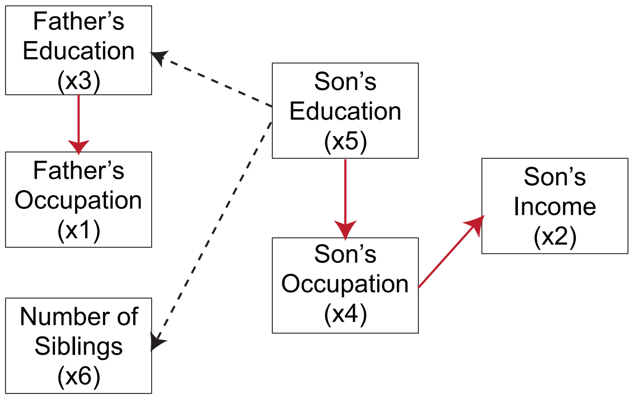


Figure 8: The estimated network by ICA-LiNGAM and Adaptive Lasso. A red solid directed edge is reasonable to the domain knowledge.

(Hoyer et al., 2008; Kawahara et al., 2010) or nonlinear relations (Hoyer et al., 2009; Mooij et al., 2009) and iv) comparison of our method and related algorithms on many other real-world data sets.

Acknowledgments

We are very grateful to Hiroshi Hasegawa (College of Science, Ibaraki University, Japan) for providing the physics data and Satoshi Hara and Ayumu Yamaoka for interesting discussion. We thank the three anonymous reviewers whose comments helped to improve the exposition of the paper. This work was partially carried out at Department of Mathematical and Computing Sciences and Department of Computer Science, Tokyo Institute of Technology, Japan. S.S., Y.K. and T.W. were partially supported by MEXT Grant-in-Aid for Young Scientists #21700302, by JSPS Grant-in-Aid for Young Scientists #20800019 and by Grant-in-Aid for Scientific Research (A) #19200013, respectively. S.S. and Y.K. were partially supported by JSPS Global COE program 'Computationism

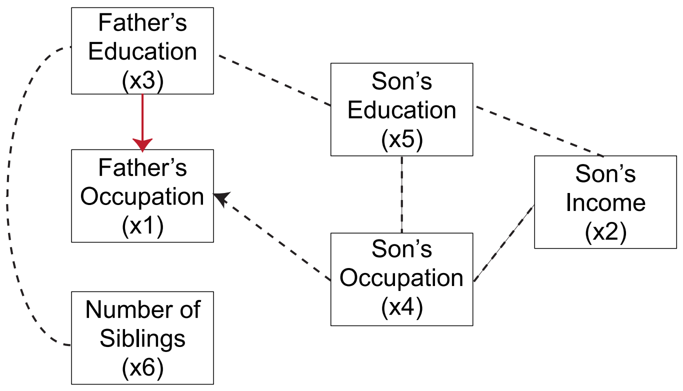


Figure 9: The estimated network by PC algorithm with 5% significance level. An undirected edge between two variables means that there is a directed edge from a variable to the other or the reverse. A red solid directed edge is reasonable to the domain knowledge.

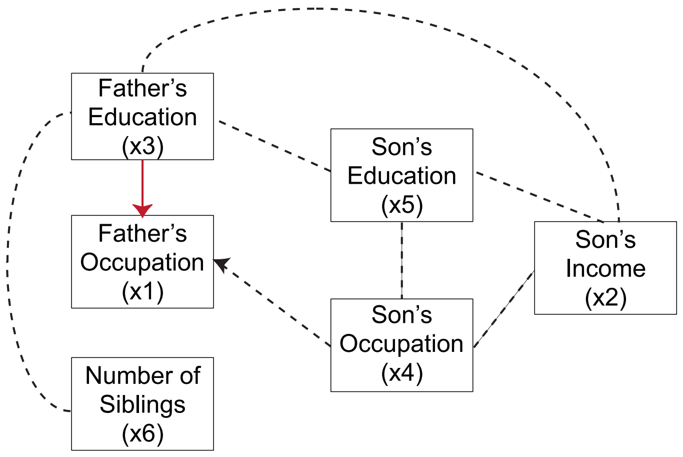


Figure 10: The estimated network by GES. An undirected edge between two variables means that there is a directed edge from a variable to the other or the reverse. A red solid directed edge is reasonable to the domain knowledge.

as a Foundation for the Sciences'. A.H. was partially supported by the Academy of Finland Centre of Excellence for Algorithmic Data Analysis.

Appendix A. Adaptive Lasso

We very briefly review the adaptive Lasso (Zou, 2006), which is a variant of the Lasso (Tibshirani, 1996). See Zou (2006) for more details. The adaptive Lasso is a regularization technique for variable selection and assumes the same data generating process as LiNGAM:

$$x_i = \sum_{k(j)<k(i)} b_{ij} x_j + e_i.$$

A big difference is that the adaptive Lasso assumes that the set of such potential parent variables x_j that $k(j)<k(i)$ is known and LiNGAM estimates the set of such variables. The adaptive Lasso penalizes connection strengths b_{ij} in L_1 penalty by minimizing the objective function defined as:

$$\left\| x_i - \sum_{k(j)<k(i)} b_{ij} x_j \right\|^2 + \lambda \sum_{k(j)<k(i)} \frac{|b_{ij}|}{|\hat{b}_{ij}|^\gamma},$$

where λ and γ are tuning parameters and $\hat{b}_{ij}$ is a consistent estimate of b_{ij}. In Zou (2006), it was suggested to select the tuning parameters by five-fold cross validation and to obtain $\hat{b}_{ij}$ by ordinary least squares regression. The adaptive Lasso has a very attractive property that it asymptotically selects the right set of such variables x_j that b_{ij} is not zero, where $k(j)<k(i)$.

References

S. Amari. Natural gradient learning works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

K. A. Bollen. *Structural Equations with Latent Variables*. John Wiley & Sons, 1989.

D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36:62–83, 1994.

G. Darmois. Analyse générale des liaisons stochastiques. *Review of the International Statistical Institute*, 21:2–8, 1953.

O. D. Duncan, D. L. Featherman, and B. Duncan. *Socioeconomic Background and Achievement*. Seminar Press, New York, 1972.

B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

J. Eriksson and V. Koivunen. Identifiability, separability, and uniqueness of linear ICA models. *IEEE Signal Processing Letters*, 11:601–604, 2004.

A. Gretton and L. Györfi. Consistent nonparametric tests of independence. *Journal of Machine Learning Research*, 11:1391–1423, 2010.

A. Gretton, O. Bousquet, A. J. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *Algorithmic Learning Theory: 16th International Conference (ALT2005)*, pages 63–77. 2005.

J. Himberg, A. Hyvärinen, and F. Esposito. Validating the independent components of neuroimaging time-series via clustering and visualization. *NeuroImage*, 22:1214–1222, 2004.

P. O. Hoyer, S. Shimizu, A. Kerminen, and M. Palviainen. Estimation of causal effects using linear non-gaussian causal models with hidden variables. *International Journal of Approximate Reasoning*, 49(2):362–378, 2008.

P. O. Hoyer, D. Janzing, J. Mooij, J. Peters, and B. Schölkopf. Nonlinear causal discovery with additive noise models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 689–696. 2009.

A. Hyvärinen. New approximations of differential entropy for independent component analysis and projection pursuit. In *Advances in Neural Information Processing Systems*, volume 10, pages 273–279. 1998.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10:626–634, 1999.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, New York, 2001.

A. Hyvärinen, K. Zhang, S. Shimizu, and P. O. Hoyer. Estimation of a structural vector autoregressive model using non-Gaussianity. *Journal of Machine Learning Research*, 11:1709–1731, May 2010.

T. Inazumi, S. Shimizu, and T. Washio. Use of prior knowledge in a non-Gaussian method for learning linear structural equation models. In *Proc. 9th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA2010)*, pages 221–228, 2010.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

Y. Kawahara, K. Bollen, S. Shimizu, and T. Washio. GroupLiNGAM: Linear non-Gaussian acyclic models for sets of variables. *arXiv:1006.5041*, June 2010.

Y. Kawahara, S. Shimizu, and T. Washio. Analyzing relationships among ARMA processes based on non-Gaussianity of external influences. *Neurocomputing*, 2011. Forthcoming.

A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physical Review E*, 69(6):066138, 2004.

G. Lacerda, P. Spirtes, J. Ramsey, and P. O. Hoyer. Discovering cyclic causal models by independent components analysis. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI2008)*, pages 366–374, 2008.

A. Londei, A. D'Ausilio, D. Basso, and M. O. Belardinelli. A new method for detecting causality in fMRI data of cognitive processing. *Cognitive processing*, 7(1):42–52, March 2006.

L. Meirovitch. *Elements of Vibration Analysis (2nd ed.)*. McGraw-Hill, 1986.

J. Mooij, D. Janzing, J. Peters, and B. Schölkopf. Regression by dependence minimization and its application to causal inference in additive noise models. In *Proceedings of the 26th International Conference on Machine Learning (ICML2009)*, pages 745–752, 2009.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000. (2nd ed. 2009).

R. O. 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*, 1:1–37, 2007.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030, 2006.

S. Shimizu, A. Hyvärinen, Y. Kawahara, and T. Washio. A direct method for estimating a causal ordering in a linear non-gaussian acyclic model. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI2009), Montreal, Canada*, pages 506–513. AUAI Press, 2009.

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, Feb 2006.

K. Sjöstrand. Matlab implementation of LASSO, LARS, the elastic net and SPCA, June 2005. URL `http://www2.imm.dtu.dk/pubdb/p.php?3897`. Version 2.0.

W. P. Skitovitch. On a property of the normal distribution. *Doklady Akademii Nauk SSSR*, 89: 217–219, 1953.

Y. Sogawa, S. Shimizu, Y. Kawahara, and T. Washio. An experimental comparison of linear non-Gaussian causal discovery methods and their variants. In *Proceedings of 2010 International Joint Conference on Neural Networks (IJCNN2010)*, pages 768–775, 2010.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:67–72, 1991.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Springer Verlag, 1993. (2nd ed. MIT Press 2000).

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society: Series B*, 58(1):267–288, 1996.

H. Zou. The adaptive Lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

Journal of Machine Learning Research 12 (2011) 1249-1286 Submitted 3/09; Revised 10/10; Published 4/11

Locally Defined Principal Curves and Surfaces

Umut Ozertem UMUT@YAHOO-INC.COM
Yahoo! Labs
701 First Ave.
Sunnyvale, CA 94086, USA

Deniz Erdogmus ERDOGMUS@ECE.NEU.EDU
Department of Electrical and Computer Engineering
409 Dana Research Center, 360 Huntington Avenue
Northeastern University
Boston, MA 02115, USA

Editor: Saharon Rosset

Abstract

Principal curves are defined as self-consistent *smooth* curves passing through the *middle* of the data, and they have been used in many applications of machine learning as a generalization, dimensionality reduction and a feature extraction tool. We redefine principal curves and surfaces in terms of the gradient and the Hessian of the probability density estimate. This provides a geometric understanding of the principal curves and surfaces, as well as a unifying view for clustering, principal curve fitting and manifold learning by regarding those as principal manifolds of different intrinsic dimensionalities. The theory does not impose any particular density estimation method can be used with any density estimator that gives continuous first and second derivatives. Therefore, we first present our principal curve/surface definition without assuming any particular density estimation method. Afterwards, we develop practical algorithms for the commonly used kernel density estimation (KDE) and Gaussian mixture models (GMM). Results of these algorithms are presented in notional data sets as well as real applications with comparisons to other approaches in the principal curve literature. All in all, we present a novel theoretical understanding of principal curves and surfaces, practical algorithms as general purpose machine learning tools, and applications of these algorithms to several practical problems.

Keywords: unsupervised learning, dimensionality reduction, principal curves, principal surfaces, subspace constrained mean-shift

1. Introduction

Principal components analysis (PCA)—also known as Karhunen-Loeve Transform–is perhaps the most commonly used dimensionality reduction method (Jolliffe, 1986; Jackson, 1991), which is defined using the linear projection that maximizes the variance in the projected space (Hotelling, 1933). For a data set, principal axes are the set of orthogonal vectors onto which the variance of the projected data points remains maximal. Another closely related property of PCA is that, for Gaussian distributions, the principal line is also self-consistent. That is, any point on the principal line is the conditional expectation of the data on the orthogonal hyperplane. In fact, this forms the basic idea behind the original principal curve definition by Hastie (1984); Hastie and Stuetzle (1989).

Due to the insufficiency of linear methods for dimensionality reduction, many nonlinear projection approaches have been studied. A common approach is to use a mixture of linear models (Bishop, 1997). Mixture models are attractive, since they arevsimple and analyzable as linear methods; however, assuming a *suitable* model order, they are able to provide much more powerful tools as compared to linear methods. Although model order selection is a tough discrete optimization problem, and mixture methods suffer from the problems introduced by improper selection of model order, there are principled ways to approach this problem such as Dirichlet process mixtures (Ferguson, 1973). Techniques based on local PCA include most well-known examples for mixture models (Fukunaga and Olsen, 1971; Meinicke and Ritter, 1999; Kambhatla and Leen, 1994, 1997).

Another common way of developing nonlinear projections is to use generalized linear models (McCullagh and Nelder, 1989; Fahrmeir and Tutz, 1994). This is based on the idea of constructing the nonlinear projection as a linear combination of nonlinear basis functions. All reproducing kernel Hilbert space techniques such as the well-known kernel PCA (Schölkopf et al., 1998) and kernel LDA (Baudat and Anouar, 2000) belong to this family. The main idea here is to map the data into a high dimensional space and perform the original linear method in this space, where the dot products are computed via a kernel function using the so-called *kernel trick*. More recent methods in this category replace the widely used Gaussian kernel with similarity metrics stemming from a weighted neighborhood graph. These methods are referred to as graph-based kernel methods (Shawe-Taylor and Singer, 2004; Ham et al., 2004).

If the data dimensionality is very high, the most successful methods are manifold learning algorithms, which are based on generating the locality information of data samples using a data proximity graph. Most well known methods that fall into this category include Isomap, local linear embedding, Laplacian eigenmaps, and maximum variance unfolding (Tenenbaum et al., 2000; Roweis and Saul, 2000; Belkin and Niyogi, 2003; Weinberger and Saul, 2006). The idea of defining geodesic distances using the data neighborhood graphs assumes that the graph does not have any *gaps* in the manifold, as well as the graph also does not go *outside* the data manifold. This requires a careful tuning of the parameters of graph construction (K or ε, as in the case of most commonly used K-nearest neighbor or ε-ball graphs), since the efficiency of the dimensionality reduction methods depend on the quality of the neighborhood graph.

At the time, Hastie and Stuetzle's proposition of self consistent principal curves (Hastie, 1984; Hastie and Stuetzle, 1989) pointed out a different track for nonlinear dimensionality reduction. They defined self-consistency over the *local* conditional data expectations, and generalized the self-consistency property of the principal line into nonlinear structures to introduce the concept of principal curves. Hastie and Stuetzle define the principal curve as *an infinitely differentiable finite length curve that passes through the middle of the data*. Self-consistency means that every point on the curve is the expected value of the data points projecting onto this point.

Hastie and Stuetzle's major theoretical contributions are the following: (*i*) they show that if a straight line is self-consistent, it is a principal component (*ii*) based on the MSE criterion, self-consistent principal curves are saddle points of the distance function. They use this second property to develop an algorithm that starts from the principal line and iteratively finds the principal curve by minimizing the average squared distance of the data points and the curve (Hastie, 1984; Hastie and Stuetzle, 1989). Although they cannot prove the convergence of their algorithm, Hastie and Stuetzle claim that principal curves are by definition a fixed point of their algorithm, and if the projection step of their algorithm is replaced with least squares line fitting, the algorithm converges to the principal line. Since there is no proof of convergence for Hastie-Stuetzle algorithm, existence of principal curves could only be proven for special cases such as elliptical distributions or distributions concentrated around a smooth curve, until Duchamp and Stuetzle's studies on principal curves on the plane (Duchamp and Stuetzle, 1996a,b).

Banfield and Raftery extend the Hastie-Stuetzle principal curve algorithm to closed curves and and propose an algorithm that reduces the estimation bias (Banfield and Raftery, 1992). Tibshirani approaches the problem from a mixture model point-of-view, and provides an algorithm that uses expectation maximization (Tibshirani, 1992). Delicado's proposition uses another property of the principal line rather than self-consistency (Delicado, 1998). Delicado's method is based on the total variance and conditional means and finds the principal curve of *oriented points* of the data set. Stanford and Raftery propose another approach that improves on the outlier robustness capabilities of principal curves (Stanford and Raftery, 2000). Probabilistic principal curves approach, which uses a cubic spline over a mixture of Gaussians to estimate the principal curves/surfaces (Chang and Grosh, 2002), is known to be among the most successful methods to overcome the common problem of bias introduced in the regions of high curvature. Verbeek and coworkers used local principal lines to construct principal curves (Verbeek et al., 2002), and a soft version of the algorithm is also available (Verbeek et al., 2001), known as K-segments and soft K-segments methods.

Algorithmically, Manifold Parzen Windows method (Vincent and Bengio, 2003; Bengio et al., 2006) the most similar method in the literature to our approach. They use a kernel density estimation (and in their later paper, a Gaussian mixture model with a regularized covariance) based density estimate that takes the leading eigenvectors of the local covariance matrices into account. Many principal curve approaches in the literature, including the original Hastie-Stuetzle algorithm, are based on the idea of minimizing mean square projection error. An obvious problem with such approaches is overfitting, and there are different methods in the literature to provide regularization. Kegl and colleagues provide a regularized version of Hastie's definition by bounding the total length of the principal curve to avoid overfitting (Kegl et al., 2000), and they also show that principal curves of bounded length always exist, if the data distribution has finite second moments. Sandilya and Kulkarni define the regularization in another way by constraining bounds on the turns of the principal curve (Sandilya and Kulkarni, 2002). Similar to Kegl's principal curve definition of bounded length, they also show that principal curves with bounded turn always exist if the data distribution has finite second moments. Later, Kegl later applies this algorithm to skeletonization of handwritten digits by extending it into the Principal Graph algorithm (Kegl and Kryzak, 2002). At this point, note that the original Hastie-Stuetzle definition requires the principal curve not to intersect itself, which is quite restrictive, and perhaps, Kegl's Principal Graph algorithm is the only approach in the principal curves literature that can handle self-intersecting data.

Overall, the original principal curve definition by Hastie and Stuetzle forms a strong basis for many, possibly all, principal curve algorithms. The idea of using least squares regression or minimum squared projection error properties of linear principal component analysis to build a nonlinear counterpart brings the problem of overfitting. Hence, algorithms based on these definitions have to introduce a regularization term. Here we take a bold step by defining the principal curves with no explicit smoothness constraint at all; we assume that smoothness of principal curves/surfaces is inherent in the smoothness of the underlying probability density (estimate). Providing the definition in terms of data probability density allows us to link open ended problems of principal curve fitting literature—like optimal regularization constraints and outlier robustness—to well established principles in density estimation literature.

In this paper we emphasize the following messages: (i) principal curves and surfaces are geometrically interesting structures of the theoretical probability distribution that underlies the data as opposed to the particular data set realization, (ii) optimal density estimation (in some sense) does not necessarily result in optimal principal surface estimation. The first point illuminates the fact that one should not seek to solve a problem such as manifold learning without precisely characterizing the sought solution; defining the sought manifold as the solution to one's optimality criterion of choice is incorrect, the solution should be defined geometrically first, and then it should be approximated and its optimality properties should be discovered, leading to optimal approximation algorithms. The second point highlights the fact that a maximum likelihood density estimate, for instance, might not lead to a maximum likelihood estimate of the principal surfaces. Statistically optimal and consistent estimation procedures for the latter must be sought by the community.

The following sections try to address the first issue mentioned above but the second issue will be left as future work; we are confident that the community will eventually propose much better algorithms for identifying principal surfaces than the ones we provide, given the framework presented here. Consequently, the subspace constrained mean shift algorithm presented later is not implied to be optimal in any statistical sense—its choice in this paper is merely due to (i) the familiarity of our audience with the mean shift clustering algorithm (which suffers from all the drawbacks we suffer, such as curse of dimensionality for kernel density estimation), (ii) the fact that it includes parametric mixture distributions as a special case of the formulation (i.e., the same formulas apply to both kernel density and mixture model estimates with minor modifications), (iii) the convergence of the algorithm to a point on the principal surface with appropriate dimensionality is guaranteed for any initial point, since mean-shift is a convergent procedure.

2. Principal Curves/Surfaces

We start with an illustration to give some intuition to our approach, and then we provide a formal definition of the principal curves and surfaces, study special cases and connections to PCA, existence conditions, limitations and ambiguities. All this will be conducted in terms of the gradient and the Hessian of the data pdf, and throughout this section, the data pdf is assumed to be known or can be estimated either parametrically or non-parametrically from the data samples. In the next section we will go back to the data samples themselves while we develop a practical algorithm.

2.1 An Illustration

Before we go into the details of the formal definition, we will present a simple illustration. Our principal curve definition essentially corresponds to the *ridge* of the probability density function. Principal curve definitions in the literature are based on local expectations and self-consistency. Hastie's self-consistency principle states that every point on the principal curve is the expected value of the points in the orthogonal subspace of the principal curve at that point—and this orthogonal space rotates along the curve. *In our view, every point on the principal surface is the local maximum, not the expected value, of the probability density in the local orthogonal subspace.*

Consider the modes (local maxima) of the pdf. On the modes, the gradient of the pdf is equal to zero and the eigenvectors of the Hessian matrix are all negative, so that the pdf is decreasing in all directions. The definition of the ridge of the pdf can be given very similarly in terms of the gradient and the Hessian of the pdf. On the ridge of the pdf, one of the eigenvectors of the Hessian is parallel with the gradient. Furthermore, the eigenvalues of the all remaining eigenvectors (which in fact

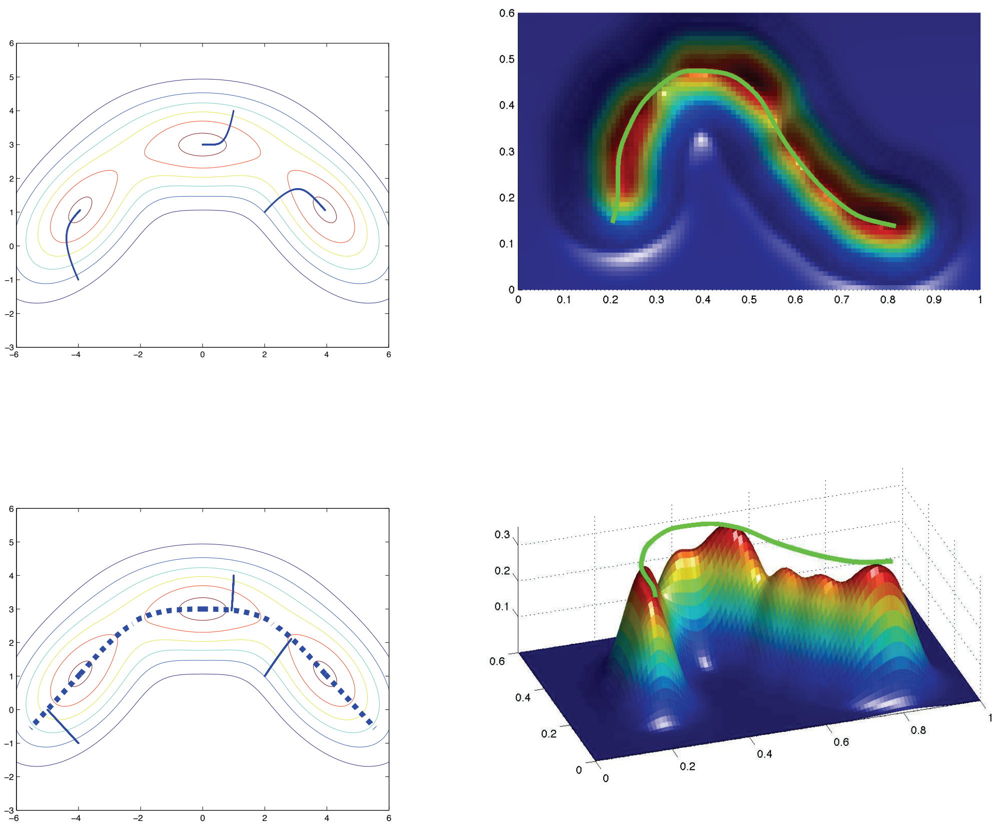


Figure 1: An illustration of the principal curve on a two Gaussian mixtures.

.

span the orthogonal space of the principal curve) are all negative, so that the pdf is decreasing in all these directions; hence the point is on a ridge, not in a valley.

Figure 1 presents two illustrations on two Gaussian mixtures. On the left, a comparison of the proposed principal curve projection, and the trajectories of the gradient of the pdf is presented. Consider a Gaussian mixture with 3 components with the pdf contour plot shown. Following the local gradient (top left) essentially coincides with well-known mean shift algorithm (Cheng, 1995; Comaniciu and Meer, 2002), and maps the points to the modes of the pdf, whereas following the eigenvectors of the local covariance (bottom left) gives an orthogonal projection onto the principal curve. The principal curve—the ridge—of this 3-component Gaussian mixture is also shown with the dashed line. On the right, we present the principal curve of a 7-component Gaussian mixture from two different points of view.

2.2 Formal Definition of Principal Curves and Surfaces

We assert that principal surfaces are geometrically well defined structures that underly the theoretical, albeit usually unknown, probability distribution function of the data; consequently, one should define principal surfaces with the assumption that the density is known—finite sample estimators of these surfaces is a question to be answered based on this characterization. Inspired by differential geometry where principal lines of curvature are well-defined and understood, we define the principal curves and surfaces in terms of the first and second order derivatives of the *assumed probability density function*. Next, we define critical, principal, and minor surfaces of all dimensions and point out facts relating to these structures—proofs are generally trivial and are omitted for most statements.

Given a random vector $\mathbf{x} \in \mathbb{R}^n$, let $p(\mathbf{x})$ be its pdf, $\mathbf{g}(\mathbf{x})$ be the transpose of the local gradient, and $\mathbf{H}(\mathbf{x})$ be the local Hessian of the probability density function. To avoid mathematical complications, we assume that the data distribution $p(\mathbf{x}) > 0$ for all $\mathbf{x}$, and is at least twice differentiable. Also let $\{(\lambda_1(\mathbf{x}), q_1(\mathbf{x})), \ldots, (\lambda_n(\mathbf{x}), q_n(\mathbf{x}))\}$ be the eigenvalue-eigenvector pairs of $\mathbf{H}(\mathbf{x})$, where the eigenvalues are sorted such that $\lambda_1(\mathbf{x}) > \lambda_2(\mathbf{x}) > \ldots > \lambda_n(\mathbf{x})$ and $\lambda_i \neq 0$.[1]

Definition 1. A point $\mathbf{x}$ is an element of the d-dimensional critical set, denoted by $\mathcal{C}^d$ iff the inner product of $\mathbf{g}(\mathbf{x})$ with at least (n-d) eigenvectors of $\mathbf{H}(\mathbf{x})$ is zero.

The definition above is an intentional extension of the familiar notion of critical points in calculus; thus local maxima, minima, and saddle points of the pdf become the simplest special case.

Fact 1. $\mathcal{C}^0$ consists of and only of the critical points (where gradient is zero) of $p(\mathbf{x})$. Furthermore, $\mathcal{C}^d \subset \mathcal{C}^{d+1}$.

In practice, this fact points to the possibility of designing dimension reduction algorithms where each data is projected to a critical manifold of one lower dimension sequentially (deflation). Alternatively, one could trace out critical curves starting off from critical points (inflation). This property of linear PCA has been extensively used in the design of on-line algorithms in the 90's (Kung et al., May 1994; Wong et al., 2000; Hegde et al., 2006).

Definition 2. A point $\mathbf{x} \in \mathcal{C}^d - \mathcal{C}^{d-1}$ is called a regular point of $\mathcal{C}^d$. Otherwise, it is an irregular point.

Fact 2. If $\mathbf{x}$ is a regular point of $\mathcal{C}^d$, then there exists an index set $I_\perp \subset \{1, \ldots, n\}$ with cardinality $|I_\perp| = (n-d)$ such that $\langle \mathbf{g}(\mathbf{x}), \mathbf{q}_i(\mathbf{x}) \rangle = 0$ iff $i \in I_\perp$. If $\mathbf{x}$ is an irregular point of $\mathcal{C}^d$, then $|I_\perp| > (n-d)$.

Regular points of a critical set are the set of points that are not also in the lower dimensional critical sets. At regular points, the gradient is orthogonal to exactly $(n-d)$ eigenvectors of the Hessian, thus these points locally lie on a surface with an intrinsic dimensionality of d. Naturally, these surfaces have their tangent and orthogonal spaces locally.

Definition 3. Let $\mathbf{x}$ be a regular point of $\mathcal{C}^d$ with $I_\perp$. Let $I_\parallel = \{1, \ldots, n\} - I_\perp$. The tangent subspace is $\mathcal{C}^d_\parallel(\mathbf{x}) = span\{\mathbf{q}_i(\mathbf{x}) | i \in I_\parallel\}$ and the normal/orthogonal subspace is $\mathcal{C}^d_\perp(\mathbf{x}) = span\{\mathbf{q}_i(\mathbf{x}) | i \in I_\perp\}$.

Definition 4. A regular point $\mathbf{x}$ of $\mathcal{C}^d$ with $I_\perp$ is (assuming no zero-eigenvalues exist for simplicity):

1. a regular point in the principal set $\mathcal{P}^d$ iff $\lambda_i(\mathbf{x}) < 0 \ \forall i \in I_\perp$; that is, $\mathbf{x}$ is a local maximum in $\mathcal{C}^d_\perp(\mathbf{x})$.

2. a regular point in the minor set $\mathcal{M}^d$ iff $\lambda_i(\mathbf{x}) > 0 \ \forall i \in I_\perp$; that is, $\mathbf{x}$ is a local minimum in $\mathcal{C}^d_\perp(\mathbf{x})$.

3. a regular point in the saddle set $\mathcal{S}^d$ otherwise; that is, $\mathbf{x}$ is a saddle in $\mathcal{C}^d_\perp(\mathbf{x})$.

1. Strict inequalities are assumed here for the theoretical analysis, because in the case of repeated eigenvalues local uncertainties similar to those in PCA will occur. We also assume non-zero eigenvalues for the Hessian of the pdf. These assumptions are not critical to the general theme of the paper and generalized conclusions can be relatively easily obtained. These ambiguities will later be discussed in Section 2.6.

Regular and irregular points in these special cases are defined similarly. Also, tangent and orthogonal subspaces are defined identically.

Clearly, $(\mathcal{P}^d, \mathcal{M}^d, \mathcal{S}^d)$ is a partition of $\mathcal{C}^d$. In practice, while principal surfaces might be useful in dimension reduction as in manifold learning, minor surfaces, valleys in the probability density function, can be useful in semi-supervised learning. A common theme in semi-supervised learning employs the so-called cluster hypothesis, where the valleys in the data probability density function have to be identified (Chapelle et al., 2006), like in the well-known Low Density Separation algorithm (Chapelle and Zien, 2005). Note that allowing zero-eigenvalues would result in local plateaus in pdf, and allowing repeated eigenvalues would result in ill-defined regular points. While conceptually the consequences are clear, we avoid discussing all possible such circumstance for now for the sake of simplicity. We give a detailed discussion on these limitations in Section 2.6.

By construction, we have $\mathbf{x} \in \mathcal{P}^0$ iff $\mathbf{x}$ is a local maximum of $p(\mathbf{x})$; $\mathbf{x} \in \mathcal{M}^0$ iff $\mathbf{x}$ is a local minimum of $p(\mathbf{x})$; $\mathbf{x} \in \mathcal{S}^0$ iff $\mathbf{x}$ is a saddle point of $p(\mathbf{x})$. Furthermore, $\mathcal{P}^d \subset \mathcal{P}^{d+1}$ and $\mathcal{M}^d \subset \mathcal{M}^{d+1}$.[2] In mean shift clustering, projections of data points to $\mathcal{P}^0$ are used to find the solution (Cheng, 1995; Comaniciu and Meer, 2002). In fact, the *attraction basin*[3] of each mode of the pdf can be taken as a local chart that has a curvilinear orthogonal coordinate system determined by the eigenvectors of the Hessian of the pdf (or a nonlinear function of it—consequences of the choice of the nonlinear function will be discussed soon).

Note that the definitions and properties above allow for piecewise smooth principal surfaces and opportunities are much broader than techniques that seek a *globally smooth optimal manifold*, which does not generally exist according to our interpretation of the geometry. Figure 2 illustrates a simple density where a globally smooth curve (for instance a principle line) can not provide a satisfactory underlying manifold; in fact such a case would likely be handled using local PCA—a solution which essentially approximates the principal curve definition we advocate above.

At this point we note that due to the assumption of a second-order continuously differentiable pdf model, the Hessian matrix and its eigenvectors and eigenvalues are continuous everywhere. Consequently, at any point on the d-dimensional principal set (or critical or minor sets) in a small open ball around this point, the points in the principal set form a continuous surface. Considering the union of open balls around points in the $d-1$-dimensional principal surface, we can note that the continuouty of the d-dimensional surface implies continuity of the $d-1$-dimensional subsurface as well as the 1-dimensional projection trajectories in the vicinity. Furthermore, if we assume that the pdf models are three-times continuously differentiable, the projection trajectories (following local Hessian eigenvectors) are not only locally continuous, but also locally continuously differentiable). In general, the order of continuous differentiability of the underlying pdf model is reflected to the emerging principal surfaces and projection trajectories accordingly.

2.3 Principal Surfaces of a Nonlinear Function of the PDF

In this section we show that for a pdf the *set of points* that constitute $\mathcal{P}^d$ is identical to the *set of points* that constitute $\mathcal{P}_f^d$ of the function $f(p(\mathbf{x}))$ where $f(\xi)$ is monotonically increasing. The same

2. Observe this inclusion property by revisiting Figure 1, as the major principal curve (show in the figures on the right) passes through all local maxima of the Gaussian mixture density.
3. The attraction basin is defined as the set of points in the feature space such that initial conditions chosen in this set evolve to a particular attractor -modes of the pdf for this particular case. In the context of mean-shift the underlying criterion is the KDE of the data. In this case, attraction basins are regions bounded by minor curves, and the attractors are the modes of the pdf.

conclusion can be drawn and shown similarly for the minor and critical surfaces; details of this will not be provided here.

Consider $\mathbf{x}$, a regular point of $\mathcal{P}^d$ with pdf $p(\mathbf{x})$ and its gradient-transpose $\mathbf{g}(\mathbf{x})$ and Hessian $\mathbf{H}(\mathbf{x})$. Then, the eigenvectors and eigenvalues of the Hessian at this point can be partitioned into the parallel and orthogonal subspace contributions: $\mathbf{H}(\mathbf{x}) = \mathbf{Q}_\parallel \Lambda_\parallel \mathbf{Q}_\parallel^T + \mathbf{Q}_\perp \Lambda_\perp \mathbf{Q}_\perp^T$, where the parallel subspace is spanned by d eigenvectors in the columns of $\mathbf{Q}_\parallel$ and the orthogonal subspace is spanned by $(n-d)$ eigenvectors in $\mathbf{Q}_\perp$. At a regular point the gradient is in the tangent space, therefore, $\mathbf{g}(\mathbf{x}) = \mathbf{Q}_\parallel \beta$ for some suitable vector β of linear combination coefficients. The gradient-transpose and Hessian of the function $f(p(\mathbf{x}))$ are:

$$\begin{aligned}
\mathbf{g}_f(\mathbf{x}) &= f\prime(p(\mathbf{x}))\mathbf{g}(\mathbf{x}) \\
&= f\prime(p(\mathbf{x}))\mathbf{Q}_\parallel \beta \,, \\
\mathbf{H}_f(\mathbf{x}) &= f\prime(p(\mathbf{x}))\mathbf{H}f(\mathbf{x}) + f\prime\prime(p(\mathbf{x}))\mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x}) \\
&= \left(f\prime(p(\mathbf{x}))\mathbf{Q}_\parallel \Lambda_\parallel \mathbf{Q}_\parallel^T + f\prime\prime(p(\mathbf{x}))\mathbf{Q}_\parallel \beta\beta^T \mathbf{Q}_\parallel^T\right) + f\prime(p(\mathbf{x}))\mathbf{Q}_\perp \Lambda_\perp \mathbf{Q}_\perp^T \,.
\end{aligned}$$

We observe that at $\mathbf{x}$ the gradient $\mathbf{g}_f(\mathbf{x})$ is also in the original d-dimensional tangent space. Further, the orthogonal subspace and the sign of its eigenvalues remain unchanged (since $f\prime(\xi) > 0$). This shows that if $\mathbf{x}$ is a regular point of $\mathcal{P}^d$, then it is also a regular point of $\mathcal{P}_f^d$. The converse statement can also be shown by switching the roles of the two functions and considering the inverse of f as the nonlinear mapping.

Note that we simply proved that the principal surface (as a set of points) of a given dimension remains unchanged under monotonic transformations of the pdf. If one projects points in higher dimensional surfaces to lower dimensional principal surfaces following trajectories traced by the Hessian of $f(p((x)))$, these projection trajectories will depend on f. This brings us to the connection with PCA.

2.4 Special Case of Gaussian Distributions, Connections to PCA

For a jointly Gaussian pdf, choosing $f(\xi) = \log(\xi)$ yields a quadratic function of $\mathbf{x}$, thus the local Hessian $\mathbf{H}_{log}(\mathbf{x}) = -(1/2)\Sigma^{-1}$ becomes independent of position. Consequently, the local Hessian eigendirections form linear trajectories and principal surfaces become hyperplanes spanned by the eigenvectors of the Gaussian's covariance matrix. If this connection to PCA is desired, that is, if the density becomes Gaussian, principal surface projections of points coincide with those one would obtain via linear PCA, then the choice $\log\, p(\mathbf{x})$ becomes attractive. Otherwise, one can seek choices of f that brings other benefits or desirable properties. For this reason, using log as the nonlinearity, we introduce the concept of local covariance matrix.

Definition 5. The local covariance-inverse of a pdf at any point $\mathbf{x}$ is given by -2 times the Hessian of the logarithm of the pdf. Specifically, in terms of the gradient-transpose and the Hessian of the pdf, this corresponds to $\mathbf{\Sigma}^{-1}(\mathbf{x}) = -p^{-1}(\mathbf{x})\mathbf{H}(\mathbf{x}) + p^{-2}\mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})$. If we assume that its eigenvalue-vector pairs are $\{\gamma_i(\mathbf{x}), \mathbf{v}_i(\mathbf{x})\}$ for $i \in \{1, \ldots, n\}$ and if the eigenvalues (some of which might be negative) are sorted as follows: $\gamma_1 < \ldots < \gamma_n$, the local ordering of critical directions from most principal to least follows the same indexing scheme (i.e., γ_n is the first to go when projecting to lower dimensions).

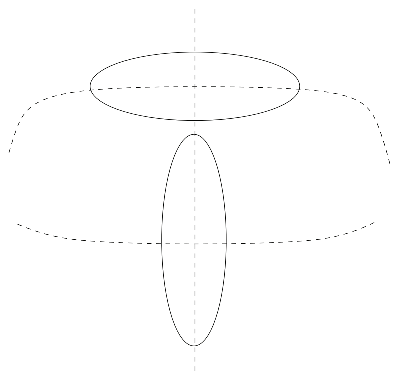


Figure 2: A T-shaped Gaussian mixture

.

2.5 Existence of Principal Curves and Surfaces

Considering Hastie's principal curve definition, the existence proof of principal curves is limited to some special cases, such as elliptical or spherical distributions concentrated around a smooth curve. It should also be noted that this definition of the principal curve requires the principal curve not to intersect itself. The principal curve definition of Kegl et al. (2000) and Sandilya and Kulkarni (2002) are theoretically more appealing in this context, since by their definition, the principal curve always exists if the distribution has finite second moments.

According to our definition, the principal curve exists as long as the data probability density is twice differentiable, such that the Hessian is nonzero. There is no restriction of finite moments, which is an improvement on existing methods. However, also note that by our definition the principal curve does not exist for uniform distributions.[4] In practice, however, since we will build our algorithms based on KDE with Gaussian kernels or GMM, even if the true underlying distribution is uniform, KDE or GMM guarantee that the gradient and Hessian are continuous.

2.6 Local Ranking of the Principal Curves and Ambiguities

In PCA, the ordering of the principal component directions are naturally given by sorting the corresponding eigenvalues of the covariance matrix in a descending order. Note that, since it coincides with PCA for Gaussian distributions, our principal curve definition also has the ambiguity that occurs in PCA; the principal surface of a spherically symmetric distribution is not well-defined.

Conditional expectation or mean squared projection error based definitions have driven the principal curves research, but in general, the definition is limited to the nonlinear counterpart of the first principal component. In fact, there is no definition of *second, third, etc. principal curve* in the literature that we are aware of. Considering the connection to PCA, one can see that our principal curve definition is not limited to the nonlinear counterpart of the first principal component, under the assumption that the Hessian matrix has distinct eigenvalues, one can obtain the *local* ordering for any d-dimensional principal manifold.

In general, data densities may take complex forms and counterintuitive scenarios may arise. Hence, generally, local information may not always indicate the global rank, and a global ordering

4. Note that one can always convolve a distribution with a spherical Gaussian or other circularly symmetric unimodal kernel to introduce continuous first and second derivatives without distorting the geometry of the principal surfaces.

in a principal set of given dimensionality may not be possible. To illustrate this fact, consider again the T-shaped Gaussian mixture consisting of two components. Note that both branches of this principal graph correspond to the leading eigenvector of the local covariance at different portions of the feature space and a global ranking is not possible.

3. Subspace Constrained Mean Shift (SCMS)

Consider the fact that $\mathcal{P}^0$, principal surface of dimensionality zero, is by construction the local maxima points of the $p(\mathbf{x})$. This presents a strong connection to clustering, since mapping to the local maxima points of the data pdf is a widely accepted clustering solution, achieved by the well-known mean shift algorithm (Cheng, 1995; Comaniciu and Meer, 2002). In this section we present a subspace constrained likelihood maximization idea that stems from Definition 4; a point on $\mathcal{P}^d$ is a local maximum in the orthogonal space. We provide an algorithm which is very similar to mean-shift in spirit. This lays an algorithmic connection between clustering and principal curve/surface fitting that accompanies the theoretical connection.

Mean-shift assumes an underlying KDE probability density of the data and implements a fixed-point iteration that maps the data points to the closest mode (local maximum) of the pdf, and the mean-shift update at any point on the feature space is parallel with the gradient of the KDE (Cheng, 1995; Comaniciu and Meer, 2002). A point is on the one dimensional principal surface iff the local gradient is an eigenvector of the local Hessian—since the gradient has to be orthogonal to the other $(n-1)$ eigenvectors—and the corresponding $(n-1)$ eigenvalues are negative. Again via the same underlying KDE assumption, a simple modification of the mean-shift algorithm by constraining the fixed-point iterations in the orthogonal space of corresponding $(n-1)$ eigenvector directions at the current point in the trajectory leads to an update that converges to the principal curves and not to the local maxima. For this case, the orthogonal space of corresponding $(n-1)$ eigenvector directions of the local covariance is the parallel space of the leading eigenvector of the local covariance. The algorithm could be modified to converge to the d-dimensional principal manifold P^d trivially, by selecting the constrained subspace as the subspace spanned by corresponding $(n-d)$ eigenvectors of the local covariance to constrain the mean-shift iterations into the subspace spanned by d leading eigenvectors of the local covariance. To provide both parametric and nonparametric variations, we will present an algorithm that can be used for well-known KDE and GMM density estimators.

Consider the data samples $\{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \Re^n$. The KDE of this data set (using Gaussian kernels) is given as

$$p(\mathbf{x}) = (1/N)\sum_{i=1}^{N} G_{\Sigma_i}(\mathbf{x}-\mathbf{x}_i)\ , \tag{1}$$

where Σ_i is the kernel covariance for $\mathbf{x}_i$; $G_{\Sigma_i}(\mathbf{y}) = C_{\Sigma_i} e^{-\mathbf{y}^T\Sigma_i^{-1}\mathbf{y}/2}$. Note that for in (1) we use the general case of anisotropic variable (data-dependent) kernel functions. For isotropic kernels one can use a scalar value instead of a full covariance, or for fixed kernel functions one can constrain the data dependency and drop the sample index i. Again for the general case, the gradient and the Hessian of the KDE are

$$\begin{aligned}
\mathbf{g}(\mathbf{x}) &= -N^{-1}\sum_{i=1}^{N} c_i\mathbf{u}_i\ ,\\
\mathbf{H}(\mathbf{x}) &= N^{-1}\sum_{i=1}^{N} c_i(\mathbf{u}_i\mathbf{u}_i^T - \Sigma_i^{-1})\ ,
\end{aligned}$$

1. Initialize the trajectories to a mesh or data points and set $t = 0$. Input the Gaussian kernel bandwidth σ (or kernel covariance matrix for anisotropic Gaussian kernels) to the algorithm.
2. For every trajectory evaluate $\mathbf{m}(\mathbf{x}(t))$ using (2) and (3).
3. Evaluate the gradient, the Hessian, and perform the eigendecomposition of $\boldsymbol{\Sigma}^{-1}(\mathbf{x}(t)) = \mathbf{V}\boldsymbol{\Gamma}\mathbf{V}$ (in specific cases, the full eigendecomposition could be avoided).
4. Let $\mathbf{V}_{\perp} = [\mathbf{v}_1 \dots \mathbf{v}_{n-d}]$ be the $(n-d)$ largest eigenvectors of Σ^{-1}
5. $\tilde{\mathbf{x}}(k) = \mathbf{V}_{\perp}\mathbf{V}_{\perp}^T\mathbf{m}(\mathbf{x})$
6. If $\mid \mathbf{g}^T(\mathbf{x})\mathbf{V}_{\perp}^T\mathbf{g}(\mathbf{x}) \mid / (\|\mathbf{g}(\mathbf{x})\| \cdot \|\mathbf{V}_{\perp}^T\mathbf{g}(\mathbf{x})\|) < \varepsilon$ then *stop*, else $\mathbf{x}(t+1) \leftarrow \tilde{\mathbf{x}}$, increment t and go to step 2.

Table 1: KDE-based SCMS Algorithm

$$\begin{aligned} \boldsymbol{\Sigma}^{-1}(\mathbf{x}) &= -p^{-1}(\mathbf{x})\mathbf{H}(\mathbf{x}) + p^{-2}\mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x}) \\ \textit{where} \quad \mathbf{u}_i &= \Sigma_i^{-1}(\mathbf{x}-\mathbf{x}_i) \quad \textit{and} \quad c_i = G_{\Sigma_i}(\mathbf{x}-\mathbf{x}_i)\ . \end{aligned} \tag{2}$$

Let $\{(\gamma_1(\mathbf{x}), \mathbf{v}_1(\mathbf{x})), \dots, (\gamma_n(\mathbf{x}), \mathbf{v}_n(\mathbf{x}))\}$ be the eigenvalue-eigenvector pairs of $\boldsymbol{\Sigma}^{-1}(\mathbf{x})$ as defined in (2) ordered from smallest to largest and the mean-shift update emerging from (2) is

$$\mathbf{x} \leftarrow \mathbf{m}(\mathbf{x}) = (\textstyle\sum_{i=1}^N c_i \Sigma_i^{-1})^{-1} \sum_{i=1}^N c_i \Sigma_i^{-1}\mathbf{x}_i. \tag{3}$$

At $\mathbf{x}$, the subspace mean-shift update is performed by projecting $\mathbf{x}$ into the constrained space $\tilde{\mathbf{x}}_k = (\mathbf{V}_{\perp}\mathbf{V}_{\perp}^T\mathbf{m}(\mathbf{x}))$. The stopping criterion can be constructed from definition directly to check if the gradient is orthogonal to the subspace spanned by the selected $n-d$ eigenvectors when projecting the data from n to d dimensions: $\mid \mathbf{g}^T(\mathbf{x})\mathbf{V}_{\perp}^T\mathbf{g}(\mathbf{x}) \mid / (\|\mathbf{g}(\mathbf{x})\| \cdot \|\mathbf{V}_{\perp}^T\mathbf{g}(\mathbf{x})\|) < \varepsilon$. For the special case of $d = 1$, an equivalent stopping criterion is that the gradient becomes an eigenvector of the Hessian, so one can employ: $\mid \mathbf{g}^T(\mathbf{x})\mathbf{H}\mathbf{g}(\mathbf{x}) \mid / (\|\mathbf{g}(\mathbf{x})\| \cdot \|\mathbf{H}\mathbf{g}(\mathbf{x})\|) > 1-\varepsilon$. Alternatively, the more traditional (but rather more risky) stopping criterion of $\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\| < \varepsilon$ can be used.

The iterations can be used to find the principal curve projection of any arbitrary point of interest in the feature space.[5] To find the principal curve projections of the data samples, a suitable way is to initialize the projection trajectories to the data samples themselves, as in mean-shift clustering. The general version of SCMS algorithm that converges to the d-dimensional principal manifold is presented in Table 1, and SCMS principal curve algorithm can simply be obtained by setting $d = 1$.

Following the derivation of the KDE with Gaussian kernel functions, using SCMS for GMM density estimates is trivial, by replacing the data samples with Gaussian mixture centers and the kernel bandwidth/covariance with the Gaussian mixture bandwidth/covariances. From now on, we will refer these as KDE-SCMS and GMM-SCMS, and we will present results based on both KDE and GMM density estimates in the next section.

5. Note that these fixed-point-update-based projections are relatively coarse approximations and more accurate projections can be obtained via numerical integration of the corresponding differential equations, for instance using Runge-Kutta order-4 method.

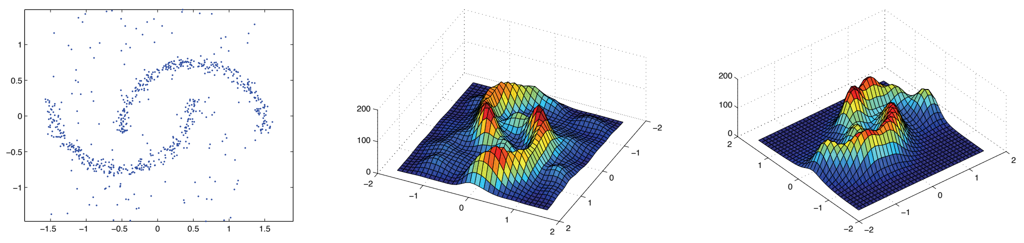


Figure 3: Curves buried in noise (left) and finite bandwidth (middle) and variable bandwidth (right) KDE

3.1 Properties of KDE-SCMS

Before proceeding to experiments we would like to briefly discuss some properties of SCMS. We believe these properties are important since they connect many open-ended questions in principal curves literature to well-studied results in density estimation. Outlier robustness and regularization properties are just some examples of the properties that are adopted from the particular density estimation method—KDE in our case. Similar algorithms that stem from the definitions in Section 2 can be designed for other density estimation methods as well. The properties presented here are a few of many possibilities to illustrate the connections.

3.1.1 COMPUTATIONAL LOAD

The computational complexity of KDE-SCMS is $O(N^2 \times n^3)$, where N is the number of samples, and n is the data dimensionality. The n^3 dependency comes from the eigendecomposition of the Hessian matrix. For GMM-SCMS, the complexity becomes $O(N \times m \times n^3)$, where m is the number of Gaussians in the mixture density estimate.[6] Note that the computational load required by SCMS is only slightly higher than the mean-shift algorithm that has been practically implemented in many application domains. The literature is rich in approaches to accelerate mean shift, all of which are directly applicable for our algorithm as well. These methods vary from simple heuristics to more principled methods like Fast Gaussian Transform (Yang et al., 2003b), quasi-Newton methods (Yang et al., 2003a) or Gaussian Blurring Mean Shift (Carreira-Perpinan, 2006). The cubic computational dependency may become the bottleneck for very high dimensional data. One solution to this problem might be to look for the d leading eigenvalues of the Hessian matrix sequentially, instead of the full eigendecomposition (as in Hegde et al., 2006), which will drop the complexity down to $O(N^2 \times d^3)$ where d is the target dimensionality ($d = 1$ for principal curves). However note that if this is the case, the computational bottleneck is not the only problem. If we have $d^3 \gg N^2$, density estimation will also suffer from *the curse of dimensionality* and our approach—that is based on the density estimation—will fail. In the experimental result section we will show results with such high dimensional data.

6. Note that this excludes the computational load required for the expectation-maximization training to fit the GMM.

3.1.2 STATISTICAL CONSISTENCY

Our algorithmic approach has used the powerful kernel density estimation (KDE) technique to estimate principal surface structures that underly data densities. The convergence properties of KDE have been well understood and bandwidth selection, especially in the case of fixed-bandwidth models, have been rigorously investigated leading to a variety of criteria for KDE construction with optimal density estimate convergence properties. In principal surfaces, however, we rely on the accurate estimation of the first and second derivatives of the multivariate data density along with the density itself. Consequently, an important question that one needs to ask (the authors thank the reviewer who posed this question) is whether the first and second order derivatives of the KDE will converge to the true corresponding derivatives, thus leading to the convergence of the principal surface structures of the KDE to those of the actual data density. Literature on the convergence properties of KDE in estimating derivatives of densities is relatively less developed—however, some work exists on the convergence of KDE derivatives in probability using isotropic kernels with dependent data and general bandwidth sequences (Hansen, 2008). In particular, a timely result on general kernel bandwidth matrices for fixed-bandwidth KDE derivatives, albeit slightly more restrictive since it uses convergence in the mean squared error sense, partly answers this question for us under relatively reasonable assumptions considering typical machine learning applications involving manifold learning (Chacon et al., 2011).

Specifically and without going into too much detail, Chacon et al. (2011) demonstrate that under the assumptions that the (unstructured but fixed) kernel bandwidth matrix converges to zero fast enough, and the underlying density and the kernel have continuous square integrable derivatives up to the necessary order or more (density must have square integrable derivatives 2 orders more than the kernel), and that the kernel has a finite covariance, the integrated mean squared error between the vector of order-r derivatives of the KDE converge to those of the true density of the data (from Theorems 1-3). The order of convergence for the integrated mean squared error has been given, from Theorems 2 & 3, as: $o(n^{-4/(d+2r+4)}) + o(n^{-1}|\mathbf{H}|^{-1/2}tr^r(\mathbf{H}^{-1}) + tr2\mathbf{H})$

This demonstrates that as the number of samples N goes to infinity, given a *sufficiently smooth* density and kernel, the derivatives will also converge. Consequently, principal surfaces characterized by first and second derivatives as in our definition will also converge.

3.1.3 OUTLIER ROBUSTNESS

Outlier robustness is another key issue in principal curve literature. Principal curve definitions that involve conditional sample expectations and mean squared projection error do not incorporate any data likelihood prior; hence, they treat each data sample equally. Such approaches are known to be sensitive to noise, and presence of outlier data samples, of course, will bias the principal curve towards outliers. Stanford and Raftery present an algorithm that improves upon the outlier robustness of the earlier approaches (Stanford and Raftery, 2000).

Outlier robustness is a well-known property of variable bandwidth KDE. In this approach, a data dependent kernel function is evaluated for each sample such that the width of the kernel is directly proportional with the likelihood that sample is an outlier. This can be implemented in various ways, and the most commonly used methods are the K-nearest neighbor based approaches, namely: (*i*) the mean/median distance to the K-nearest neighbor data points, (*ii*) sum of the weights of K-nearest neighbor data points in a weighted KDE. Hence, the kernel bandwidth increases for the samples that are in a sparse neighborhood of data samples. Figure 3 (left) presents a data set consisting

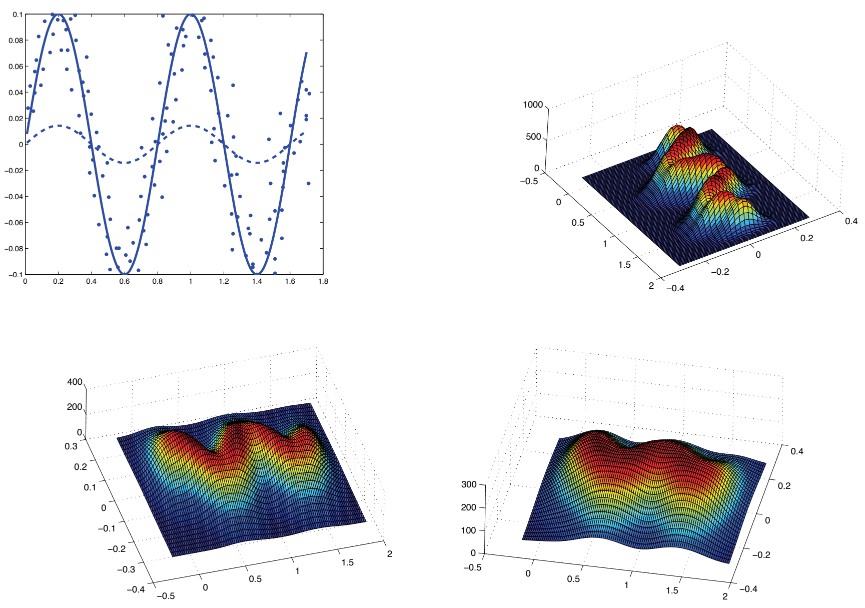


Figure 4: Mean projection error vs. overfitting tradeoff as a kernel bandwidth selection problem. Three density estimates are presented—a narrow bandwidth (left) Maximum Likelihood kernel bandwidth (middle) and a wide kernel bandwidth (right)

of two crescent-like clusters buried in noise. In fact, this data set is similar to the illustration that Stanford and Raftery use as they propose their noise robust principal curve approach (Stanford and Raftery, 2000). We present the fixed and variable bandwidth—using K-nearest neighbor method (i) mentioned above and selecting $K = N^{1/4}$—KDE of the data set in Figure 3 in middle and right, respectively. Note that in the resulting density estimate the variable size KDE eliminates the effects of the outliers without oversmoothing or distorting the pdf significantly in the support of the data. Selecting the kernel functions in a data dependent manner, can make KDE-based SCMS robust to outliers in the data. However, additional computational load of variable kernel bandwidth evaluations may increase the overall computational complexity.

3.1.4 Regularization and Overfitting

If a problem is formulated over sample expectations or minimization of the average projection error, the issue of overfitting arises. In the context of principal curves and surfaces, most explicitly, Kegl brings up this question in his PhD dissertation (Kegl, 1999). Considering the data set and principal curves in Figure 4 (left), Kegl asks, which of the curves is the right one. "Is the solid curve following the data too closely, or is the dashed curve generalizing too much?" In general, of course, this is an open ended question and the answer depends on the particular application.

Still, density estimation methods can provide many approaches to define the regularization, varying from heuristics to theoretically well-founded approaches like maximum likelihood. In other words, instead of trying for different length (Kegl et al., 2000) or curvature (Sandilya and Kulkarni,

2002) parameters, density estimation can provide purely data-driven approaches, where the regularization parameters are learned from the data directly using cross-validation.

Figure 4 shows density estimates obtained using KDE for different kernel bandwidth selections for the data set presented. In SCMS, the trade-off between projection error and overfitting can be adjusted by setting the kernel width. One can select the kernel bandwidth manually by observing the data or exploiting domain specific knowledge. This is, of course, not much different than observing the data and selecting a suitable length or curvature constraint. However, the real advantage here is the rich literature on how to select the kernel function from the data samples directly. There are many theoretically well-founded ways of optimizing the kernel width according to maximum likelihood or similar criteria (Silverman, 1986; Parzen, 1962; Comaniciu, 2003; Sheather and Jones, 1991; Jones et al., 1996; Raykar and Duraiswami, 2006).

Furthermore, anisotropic and/or variable size kernel functions naturally implement many types of constraints that cannot be defined by any length or bound of turn. By selecting anisotropic kernel functions, one can define the regularization constraint at different scales along different directions. This can also be achieved by lenghth/curvature constraints by scaling the data differently among different dimensions. However, data-dependent variable bandwidth kernels can define varying constraints throughout the space. This is not possible to achieve by a constant curvature or length penalty of any sort.

In summary, our KDE based principal curve projection algorithm not only connects the trade off between the projection error and generalization into well studied results of density estimation field, it also allows one to derive data-dependent constraints that vary throughout the space, which cannot be given by any length or curvature constraint whatsoever. Although this still cannot ultimately answer the open-ended question on the trade-off between the regularization and projection error, it provides a principled way to approach the problem and proves to be effective in many real applications as we will show next.

4. Experimental Results

This section consists of three parts. In the first part, we provide comparisons with some earlier principal curve algorithms in the literature. We perform simulations on notional data sets and give performance and computation times. In the second part, we focus on real applications, where we briefly mention some applications with pointers to our recent publications and also provide results in some areas that principal curves has (feature extraction for OCR) or has not been (time-frequency distribution sharpening, MIMO channel equalization) used before. In these applications we use SCMS *directly*. Surely, pre- and post-processing steps can be added to improve performance of these applications, however our aim is to show the versatility of the approach not to optimize every implementation detail. In the third and final part, we focus on the limitations of the method.

Same as the principal line, principal curves—in our definition—extend to infinity. In general though, what one is really interested in is not the whole structure, but the projections of samples onto the underlying structure. Therefore, throughout this section, rather than populating samples on the curve that extend to infinity, we prefer representing the principal curve with *the data samples projected onto the principal curve*, so that the curves in the plots remain in the support of the data. For the same reason, although the underlying structure is continuous (and can be populated into any desired density), the presented principal curves sometimes do not *look* continuous where the data is sparse.

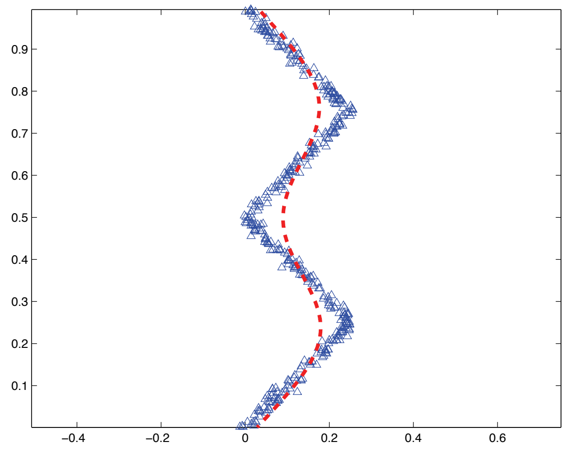


Figure 5: Zig-zag data set, and Hastie-Stuetzle principal curve

4.1 Comparisons with Other Principal Curve Methods

In this section we present comparisons with original Hastie-Stuezle principal curve method (Hastie, 1984; Hastie and Stuetzle, 1989) and the Polygonal Line Algorithm by Kegl et al. (Kegl et al., 2000), and we provide both computation time and performance comparisons.

4.1.1 Zig-Zag Data Set

Zig-Zag data set has been used in an earlier principal curve paper by Kegl et al. (2000) (This data set is provided by Kegl). Figure 5 shows the data samples and result of Hastie's algorithm. Figure 6 presents the results of Kegl's polygonal line algorithm for different penalty coefficients. The length penalty coefficient is equal to $0.1, 0.3, 0.5$, and 0.7, respectively. Polygonal Line algorithm with the right length penalty seems to be working the best for this dataset with high curvature on the corners.

In Figure 7 we compare results of the SCMS algorithm based on three different density estimates: (i) KDE with constant bandwidth, (ii) KDE with variable (data-dependent) covariance (iii) Gaussian mixture with 4 components. For (i) and (ii), the bandwidth and covariance of the Gaussian kernel are selected according to the leave-one-out maximum likelihood criterion (Duda et al., 2000). For the Gaussian mixture model, the *correct* model order is assumed to be known and a standard expectation-maximization algorithm is used to estimate the parameters (Duda et al., 2000).

Here all density estimates lead to very similar results. Since it allows one to learn the kernel covariances elongated with the data, (ii) gives a sharper KDE estimate as compared to (i). However, since there is no significant difference between the principal curve projections of these two, (ii) might be regarded as somewhat overfitting, since too many parameters (d^2 additional parameters per sample, as the constant kernel bandwidth is replaced by a full data-dependent covariance) are learned, leading to no significant changes. The result shown in (iii) is a good example which shows that good results can be obtained if the parametric family fits the distribution very well. Of course, as you can imagine, the GMM based results might have been much worse for an unsuitable selection of the number of components, or if EM converges to a suboptimal result due to poor initialization, whereas KDE is much more robust in this sense. Also note an analogy to Kegl's approach, using

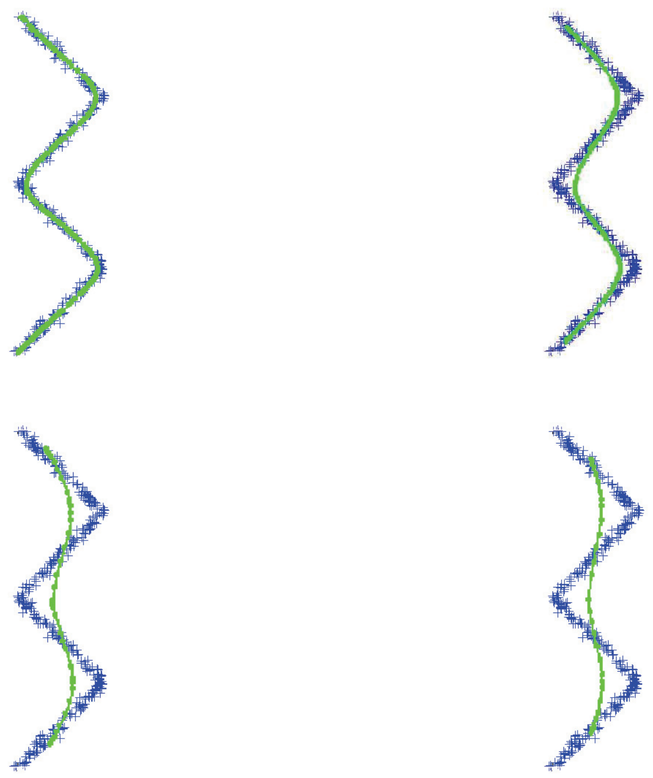


Figure 6: Zig-zag data set, and result of the Polygonal Line Algorithm

GMM-SCMS leads to a piecewise linear structure if the Gaussian components are sufficiently far (in Mahalanobis distance sence) from each other. In the vicinity of the Gaussian component centers, except when components significantly overlap or get close, the principal curves can be approximated well linearly by piecewise local components.

Note that theoretically the principal curves in (i) and (ii) extend to infinity on both ends; and for the GMM based example in (iii), each component crosses and extends to infinity. Here—and also for the rest of the paper—we present the data projected onto principal curve only, that depicts the portion of the principal curve in the support of the input data. The nature of the curves outside this region is obvious from the definition and the density estimate plots.

4.1.2 SPIRAL DATA SET

Since many principal curve algorithms are based on the idea of starting with the principal line and adding complexity to the structure (for example adding a vertex to piecewise linear curve) to minimize mean projected error, a data set that folds onto itself may lead to counterintuitive results, and spiral data set is a benchmark data set that has been used in manifold learning and principal curve algorithm literature (Kegl et al., 2000; Vincent and Bengio, 2003) (again, this data set is provided by Kegl).

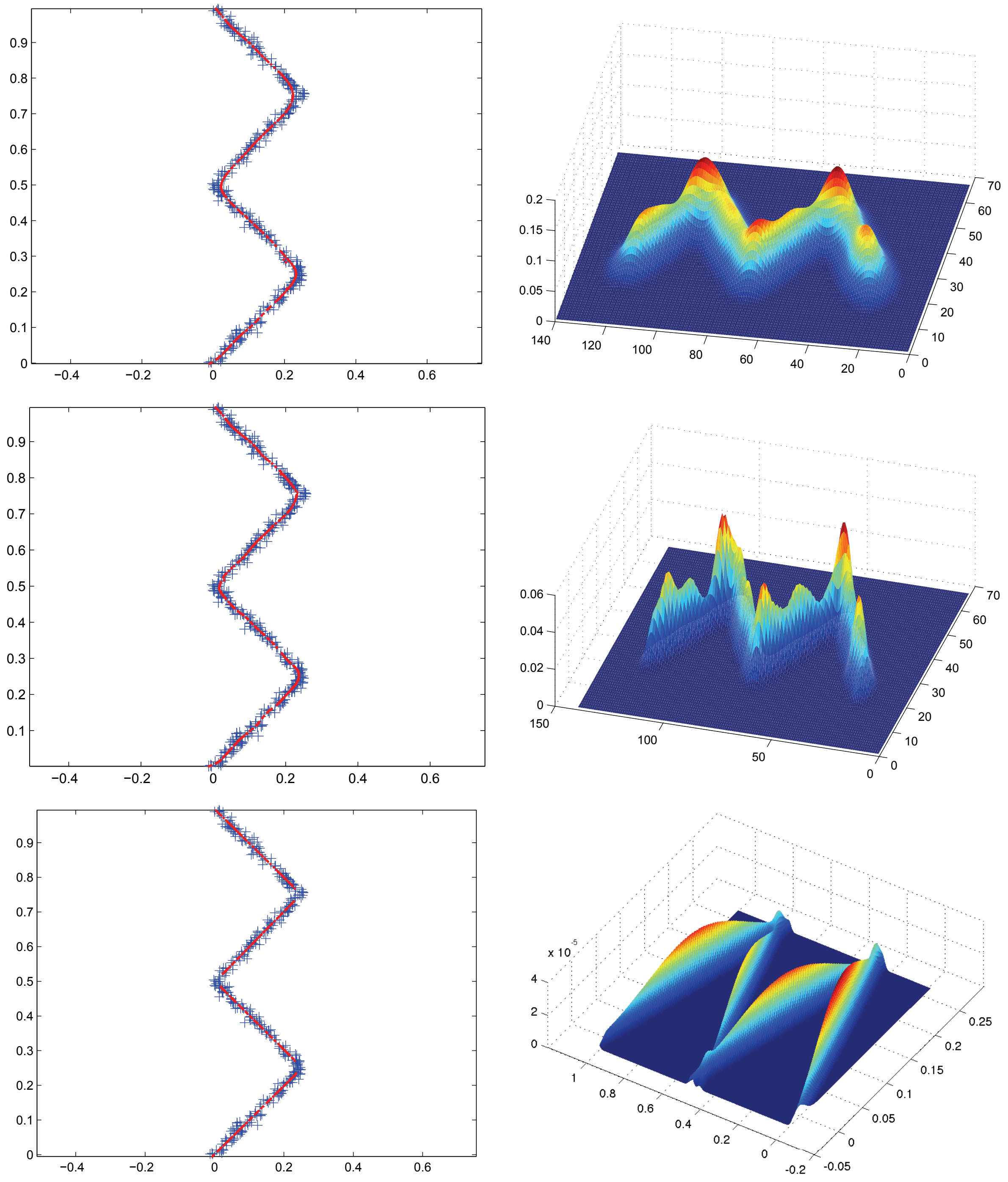


Figure 7: Zig-zag data set, and its principal curve projections obtained for KDE with isotropic constant bandwidth (top), KDE with anisotropic and data-dependent covariance (middle), and Gaussian mixture with 4 components (bottom). The underlying density estimates are shown on the right column.

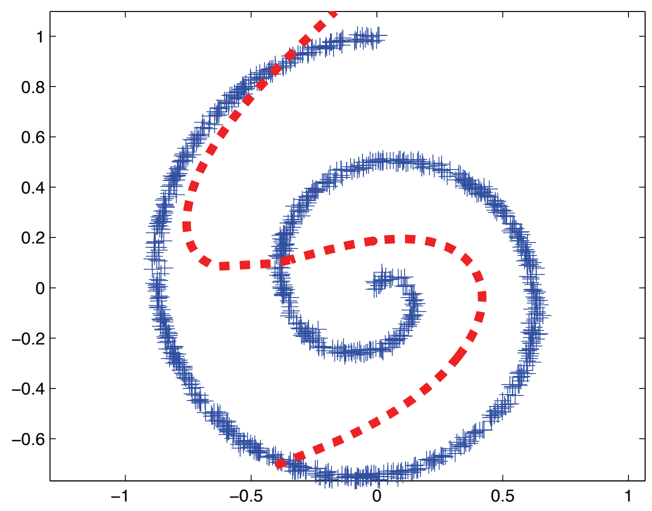


Figure 8: Spiral data set, and Hastie-Stuetzle principal curve

Similar to the previous example, we start with the results of Hastie-Stuetzle algorithm and Kegl's polygonal line algorithm. Figure 8 shows the data samples and the result of Hastie's algorithm. Figure 9 presents the results of Kegl's polygonal line algorithm for different penalty coefficients. The length penalty coefficient is equal to $0.1, 0.2, 0.4$, and 0.5, respectively.

As in the previous example, in SCMS uses the leave-one-out ML kernel bandwidth for this data set. Figure 10 shows the same spiral data set along with the results of KDE-SCMS. Comparing Figure 9 and Figure 10, one can see that both Polygonal Line algorithm—with suitable parameters—and our locally defined principal curve can achieve satisfactory results. Therefore, we create a more challenging scenario, where the spiral this time has some substantial noise around the underlying generating curve and has fewer samples. Figure 11 shows the result of KDE-SCMS, and Figure 12 shows results of Polygonal Line algorithm for different penalty coefficients; $0.05, 0.1, 0.2$, and 0.3.

On the noisy spiral data set, we also provide quantitative results for different noise levels and compare the computation times. At each noise level, we find the principal curve using both methods using the same noisy data set, and afterwards we take another 200 samples from the same generating curve and add same amount of radial noise to use as the test set. We present the MSE between the projection of the test samples and their original points on the noiseless generating curve. Results for KDE-SCMS, and Polygonal Line algorithm are presented in Table 2 along with corresponding running times for 50 Monte Carlo runs of this experiment. Since results are presented for the leave-one-out ML kernel bandwidth, the running times for SCMS include this ML training as well. For the Polygonal Line algorithm we performed a manual parameter tuning for each noise level and best results are presented.

Overall, as the noise level increases, the computation time of SCMS increases, presumably due to more iterations being required for convergence; still, the computation time is much less than that of the Polygonal Line algorithm. In terms of MSE between the estimated and the true curve, SCMS provides similar or better performance as compared to the Polygonal Line algorithm. For

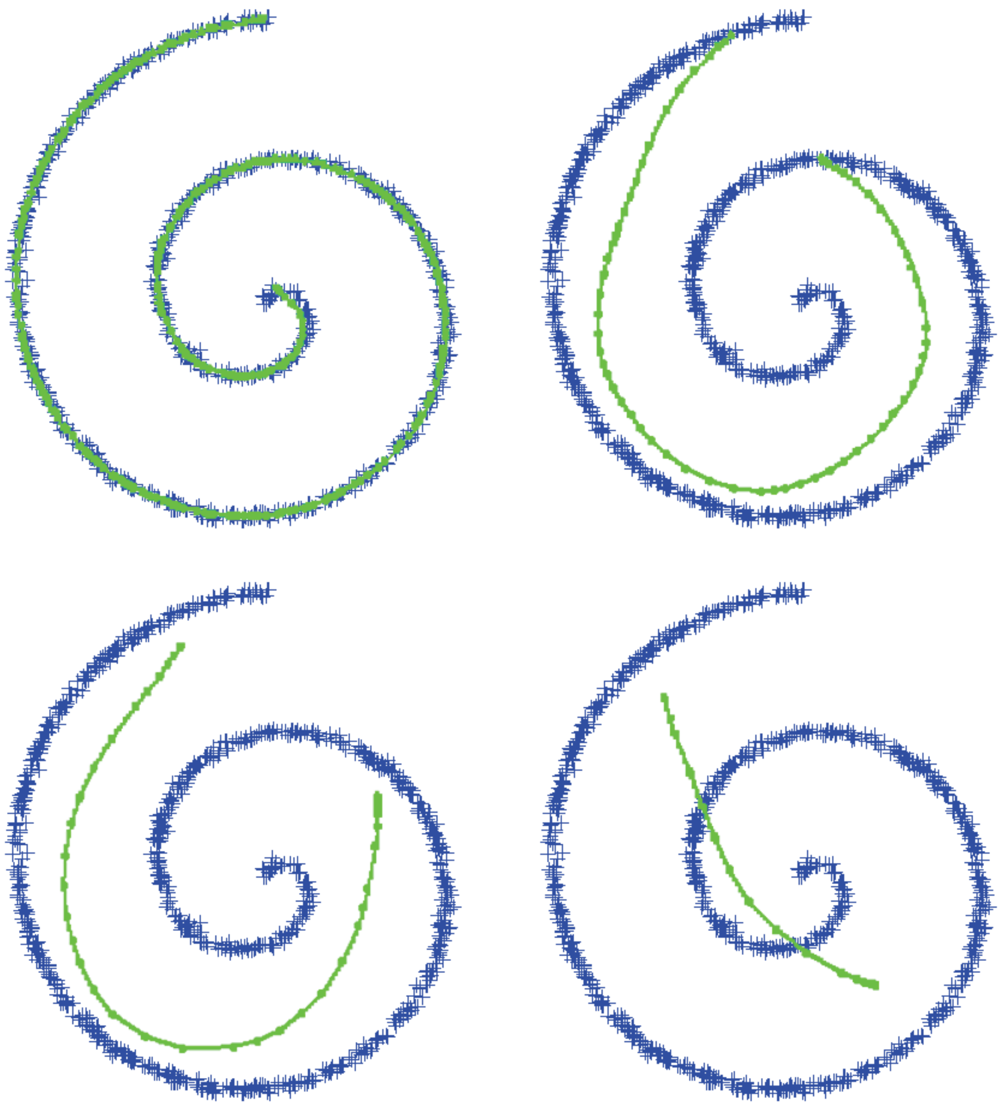


Figure 9: Spiral data set, and result of the Polygonal Line Algorithm

some noise levels the difference in performance is very small; however, note that the real advantage of SCMS is that it provides the similar/better results nonparametrically—as compared to the best result of several runs of the Polygonal Line algorithm with different parameters.

4.1.3 LOOPS, SELF-INTERSECTIONS, AND BIFURCATIONS

Since they are specifically designed to fit *smooth* curves to the data, traditional principal curve fitting approaches in the literature have difficulties if there are loops, bifurcations and self intersections in the data. Perhaps the most efficient algorithm in this context is Kegl's principal graph algorithm (Kegl and Kryzak, 2002), where Kegl modifies his polygonal line algorithm (Kegl et al., 2000) with a table of predefined rules to handle these irregularities. On the other hand, in the presence of

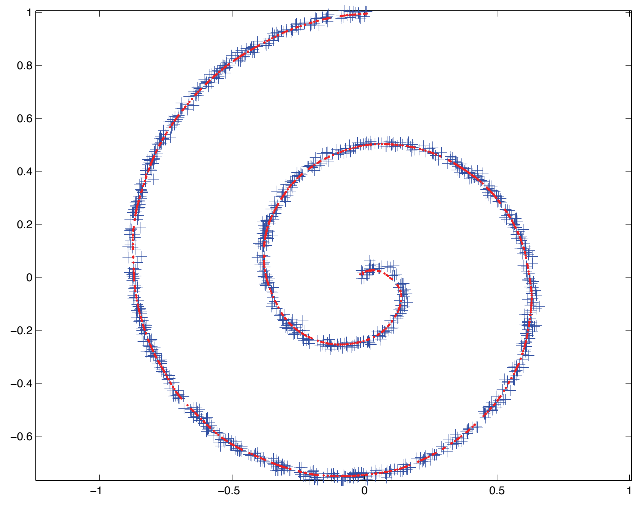


Figure 10: Spiral data set, and KDE-SCMS principal curve

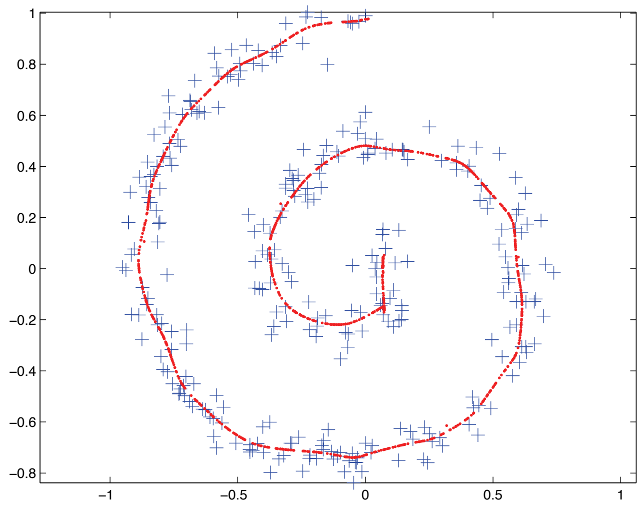


Figure 11: Noisy spiral data set, and KDE-SCMS principal curve

such irregularities, our definition yields a principal graph—a collection of smooth curves. Since the ridges of the pdf can intersect each other, KDE-SCMS can handle such data sets with no additional effort/parameter. Results of KDE-SCMS on a synthetically-created snow crystal data set that has a number of loops, self intersections, and bifurcation points is presented in Figure 13.

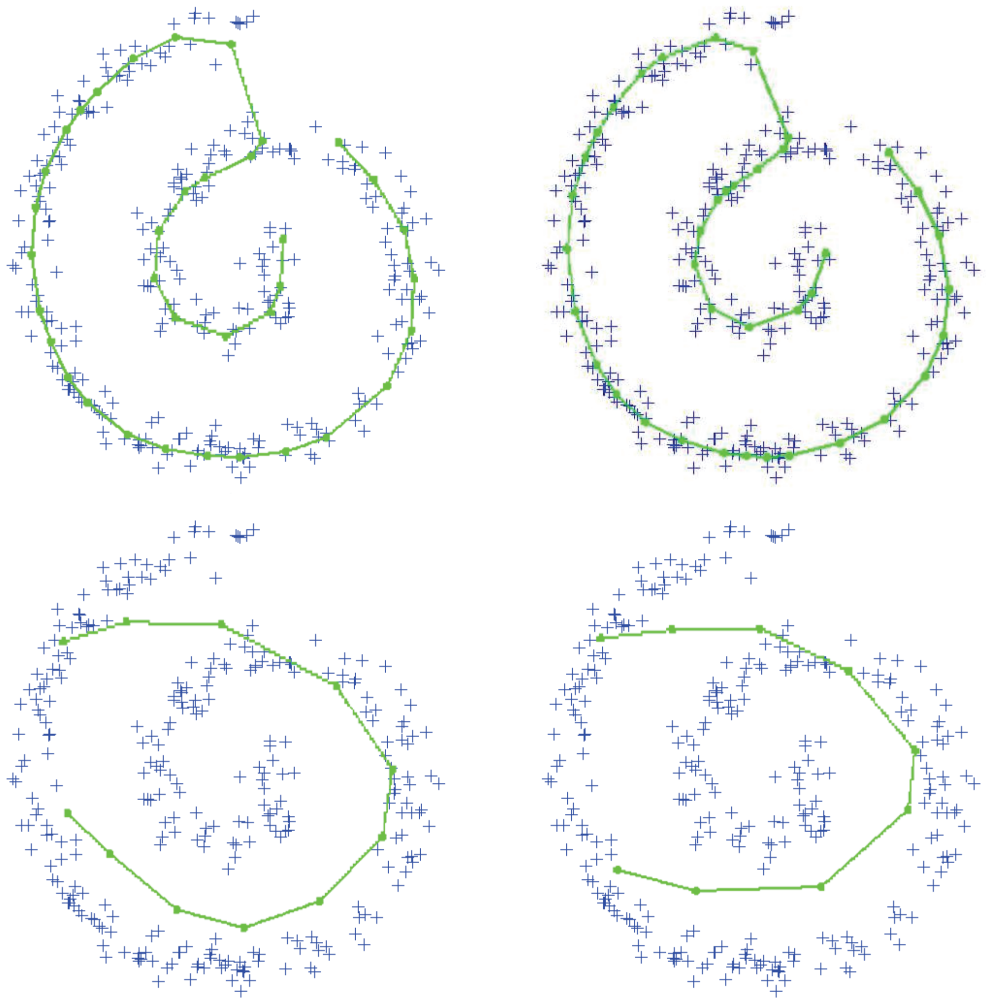


Figure 12: Noisy spiral data set, and result of the Polygonal Line Algorithm

4.1.4 EXTENDING THE DEFINITION TO HIGHER DIMENSIONAL MANIFOLDS

The generalization of principal curves to principal surfaces and higher order manifolds is naturally achieved with our definition. Here we present the results of KDE-SCMS for $d = 1$ and $d = 2$ for a three-dimensional helix data set in Figure 14. (For $d = 2$, we present the surface built by the Delaunay triangulations Delaunay, 1934 of the principal surface projections for better visualization.) Here, note that the covariance of the helix data around the principal curve is not symmetric, and the horizontal dimension has a higher variance (and this is why the principal surface is spanned along this dimension). If the helix had been symmetric around the principal curve, the principal surface would have been ill-defined.

	Computation time	Mean squared projection error	σ_{noise}
SCMS	3.237 sec.	0.003184	0.005
PL	18.422 sec.	0.017677	0.005
SCMS	3.508 sec.	0.011551	0.01
PL	20.547 sec.	0.024497	0.01
SCMS	3.986 sec.	0.062832	0.02
PL	22.671 sec.	0.066665	0.02
SCMS	6.257 sec.	0.194560	0.04
PL	27.672 sec.	0.269184	0.04
SCMS	7.198 sec.	0.433269	0.06
PL	19.093 sec.	0.618819	0.06
SCMS	8.813 sec	0.912748	0.08
PL	19.719 sec	1.883287	0.08

Table 2: Computation Time and MSE Performance Comparisons

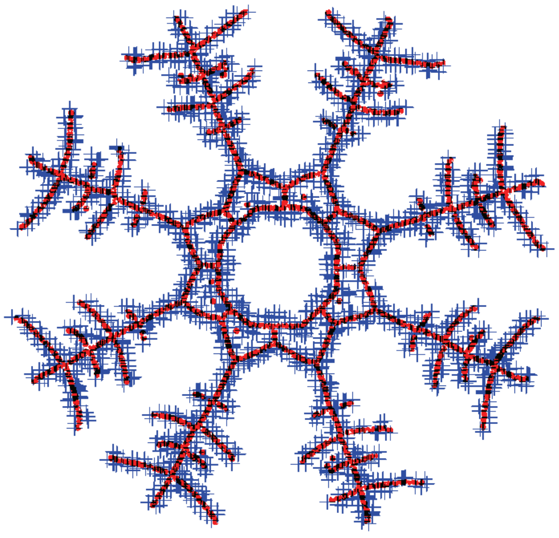
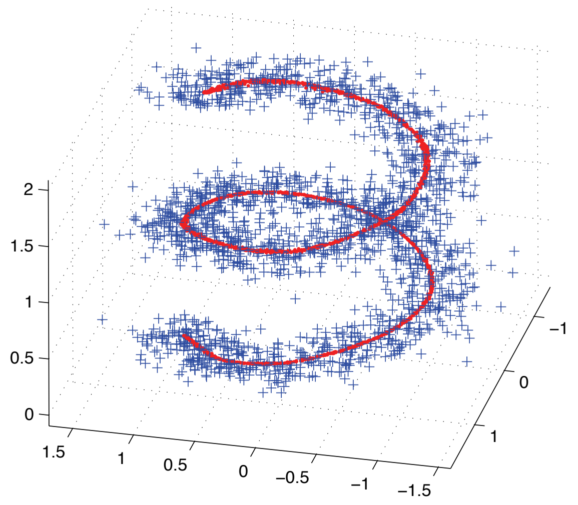


Figure 13: Snow crystal data set, and KDE-based SCMS result

4.2 Applications of Principal Curves

In the following, we will present a number of applications of our approach; on time series denoising, independent components analysis, time-frequency reassignment, channel equalization, and optical character skeletonization.



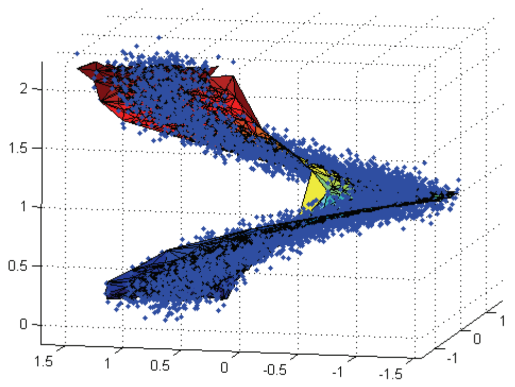


Figure 14: Helix data set, and KDE-based SCMS result for $d = 1$ (top) and $d = 2$ (bottom)

4.2.1 Time Series Signal Denoising

KDE-SCMS finds use in many applications of time series denoising. In general, the feature space for such problems can be constructed using the time index as one of the features, yielding an embedded structure of the—possibly multidimensional—time signal. In such spaces, we show that KDE-SCMS can successfully be used for denoising (Ozertem and Erdogmus, 2009; Ozertem et al., 2008). In the following, first we will briefly mention our previous work on applying KDE-SCMS to signal denoising applications, and proceed with preliminary results in two other application domains.

We proposed to use principal curve projections as a nonparametric denoising filter at the preprocessing stage of time warping algorithms, which in general are prone to noise (Ozertem and Erdogmus, 2009). In this setting, time embedding is used in the scatter plot of the pair of signals that we want to find the time warping function in between. We use a slightly different variant of KDE-based SCMS for this purpose that exploits the application specific case that the time embedding dimension is not a random variable, and shown improvement in time series classification and clustering.

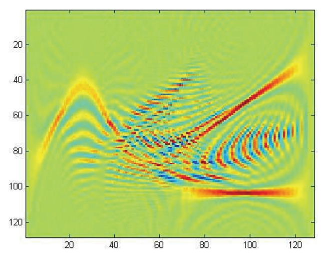
A common problem in signal denoising is that if the signal has a blocky, in other words, a piecewise-smooth structure, traditional frequency domain filtering techniques may lead to oversmoothings in discontinuities. One idea to overcome this is to take discrete wavelet transform (DWT), do the filtering (or thresholding) in this domain and recover the smoothed signal by taking inverse DWT. The shortcoming of this is high frequency artifacts (similar to Gibbs-effect) at both ends of the discontinuities. We show that KDE-SCMS can be used for this purpose (Ozertem et al., 2008). Since at the discontinuities, KDE will not be much affected by the signal samples of the other end of the discontinuity, the algorithms leads to a piecewise-smooth denoising result without introducing oversmoothings or any artifacts at the discontinuities.

4.2.2 NONLINEAR INDEPENDENT COMPONENT ANALYSIS

The proposed principal surface definition can be viewed in a differential geometric framework as follows: at each point $\mathbf{x}$, the solutions to the differential equations that characterize curve whose tangents are the eigenvectors of the local covariance of the pdf form a local curvilinear coordinate system that is isomorphic to an Euclidean space in some open ball around $\mathbf{x}$. The trajectories that take a point $\mathbf{x}$ to its projection on the d-dimensional principal surface can be used to obtain these curvilinear coordinates that specify the point with respect to some reference critical point that can be assumed to be the origin. Consequently, for instance, for $\mathbf{x}$, the lengths of curves during its projection from n-dimensional space to the $(n-1)$-dimensional principal surface, and then subsequently to $(n-2),\dots,1$, and eventually to a local maximum (the one that has been recognized as the origin) could, in some cases when a global manifold unfolding is possible, lead to a nonlinear coordinate vector. This manifold unfolding strategy can be used in many applications including visualization and nonlinear blind source separation. As we do not aim to focus on the manifold unwrapping aspects of the proposed framework in this paper (because that, in principle, requires solving differential equations accurately and the proposed algorithm is not at a desired level of accuracy for that purpose), we simply point out that the definition presented here allows for a principled coordinate unfolding strategy as demonstrated in nonlinear blind source separation (Erdogmus and Ozertem, 2007). Developing fast feedforward approximations (via parametric or nonparametric mappings) to this manifold unwrapping strategy remains as a critical future work.

4.2.3 TIME-FREQUENCY DISTRIBUTION REASSIGNMENT

Time-frequency reassignment is a known problem in signal processing literature and yields another example, where KDE-SCMS can be applied directly. As any other bilinear energy distribution, the spectrogram is faced with an unavoidable trade-off between the reduction of misleading interference terms and a sharp localization of the signal components. To reduce the smoothing effects introduced by the window function in short-term Fourier transform, reassignment methods are used to sharpen the time-frequency representation by using the rate of change of phase of the signal, which finds numerous applications in speech signal processing and signal roughness analysis (Fulop and Fitz, 2007; K. Fitz and L. Haken and P. Christensen, 2000). Parameter envelopes of spectral components are obtained by *following ridges on the smooth time-frequency surface*, using the reassignment


(a) The Wigner-Ville distribution

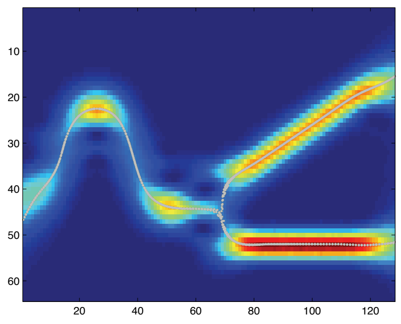

(b) Smoothed Wigner-Ville distribution and its principal curve

Figure 15: Wigner-Ville distribution in time-frequency domain, its smoothed version and principal curve of the smoothed distribution

method (Auger and Flandrin, May 1995) to improve the time and frequency estimates for the envelope breakpoints. Figure 15 shows our preliminary results for a synthetic time frequency surface with multiple components in some time intervals that yield cross interference terms. Wigner-Ville distribution of the signal, and the smoothed Wigner-Ville distribution, where the cross-terms in the original spectogram are eliminated are shown in Figure 15(a). Figure 15(b) shows the principal curve of this time-frequency surface obtained by KDE-SCMS.

Furthermore, in the presence of the *auto-cross terms*, a much more challenging scenario appears (Ozdemir and Arikan, 2000; Ozdemir et al., 2001). In these cases a rotation invariant reassignment method is required and traditional methods that are based on the rate of change of the phase cannot answer this need. KDE-SCMS, on the other hand, is still directly applicable to this problem because it is invariant to rotations in the input data.

4.2.4 TIME-VARYING MIMO CHANNEL EQUALIZATION

Recently, multiple-input multiple-output wireless communication systems have drawn considerable attention, and there are reliable and computationally inexpensive symbol detection algorithms in the literature (Foschini et al., Nov 1999). On the other hand, applications in time-varying environments pose a harder problem to the changing channel state, and some supervised algorithms have been proposed to tackle this issue, where an initialization phase is used in the beginning for training purpose (Rontogiannis et al., May 2006; Karami and Shiva, 2006; Choi et al., Nov. 2005).

Blind channel equalization approaches in the literature are based on clustering (Chen et al., Jul 1993). However, these approaches mostly focus on time-invariant single-input single-output channels. Recently, a spectral clustering technique is proposed that extends the applications into time-varying multiple-input multiple-output channels as well (Van Vaerenbergh et al., 2007; Vaerenbergh and Santamaria, 2008). Van Vaerenbergh and Santamaria introduce the time embedding into the feature space before employing the clustering algorithm to *untangle* the clusters. The same idea proves to be effective in Post-Nonlinear Blind Source Separation as well (Vaerenbergh and Santamaría, 2006).

The original clustering problem in four dimensions presented in Figure 16(a). The fast time-varying nature of the channel poses a very difficult clustering problem with overlapping clusters. With the time embedding, the overlapping clusters become intertwined threads as shown in Figure 16(b) with two three-dimensional subspace projections of the data. Van Vaerenbergh and Santamaria employ a spectral clustering algorithm to solve the channel equalization problem with no supervision. At this point, one can improve noise robustness of the clustering by using the fact that the clusters are curves in the feature space by using the spectral clustering of the principal curve projections instead of the data samples. Figure 17 shows a result of KDE-SCMS for the same data set at signal to noise ratio of 5dB, along with the average normalized MSE (and $\pm$ one standard deviation) between the actual noisefree signal and the principal curve projection over 20 Monte Carlo runs. The principal curve projection result can give a good estimate of the noisefree signal even in signal to noise ratio levels even lower than 0dB—where the noise power is greater than the signal power itself.

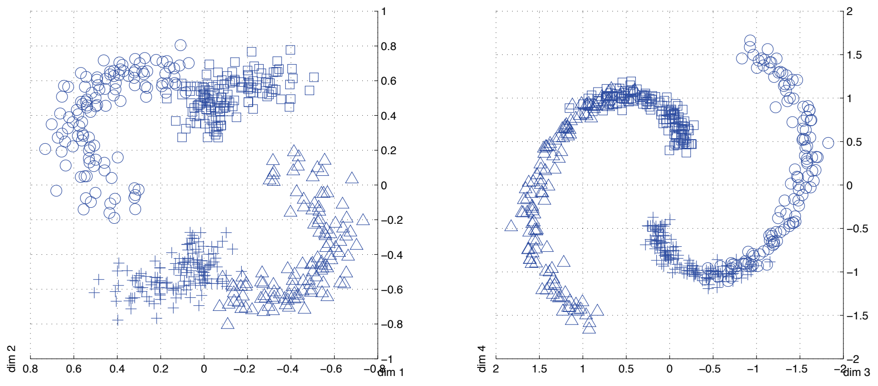
4.2.5 SKELETONIZATION OF OPTICAL CHARACTERS

Optical character skeletonization can be used for two purposes: feature extraction for optical character recognition and compression. Principal curves have been used for this application (Kegl and Kryzak, 2002). One significant problem with applying principal curve algorithms to skeletonization of optical characters is that, by definition, algorithms are seeking for a *smooth curve*. In general, data may have loops, self intersections, and bifurcation points, which is obviously the case for optical characters. Kegl's principal graph algorithm is perhaps the only method in the literature that can successfully handle such irregularities (Kegl and Kryzak, 2002). In this approach, Kegl reshapes his polygonal line algorithm (Kegl et al., 2000) to handle loops, and self intersections by modifying it with a table of rules and adding preprocessing and postprocessing steps. Using the handwritten digits data set provided by Kegl, we show the results of KDE-SCMS. Figure 18 shows the binary images along with the principal curve projection of the pixels. SCMS gives satisfactory results without any rule or model based special treatment for the self intersections.

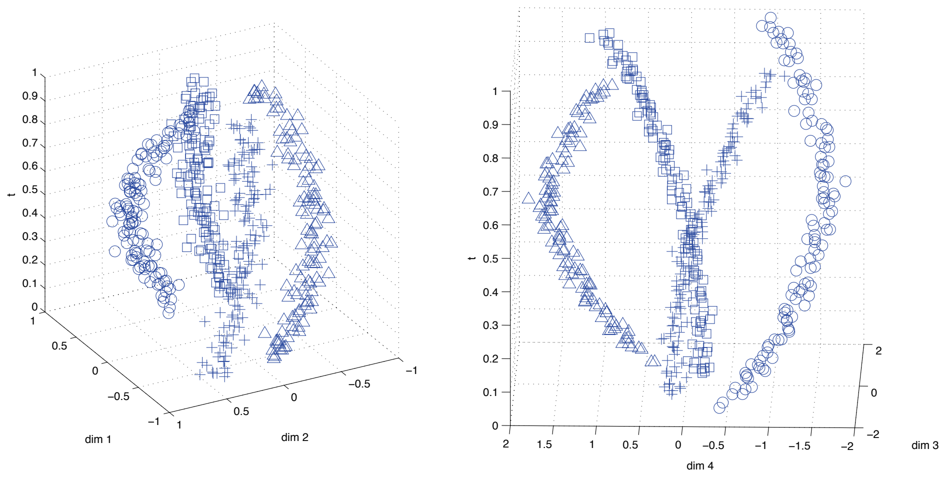
4.3 Limitations, Finite Sample Effects, and the Curse of Dimensionality

Since our principal curve definition assumes the pdf to be given, it depends on the reliability of the preceding density estimation step, which in general may not be an easy task. Stated by Bellman as *the curse of dimensionality* (Bellman, 1961), it is a very well-known fact that density estimation becomes a much harder problem as the dimensionality of the data increases. Therefore, before we move on to applications on real data, in this section we will present the performance of our principal curve fitting results for various density estimates with different number of samples and dimensions.

The first comparison is with principal line estimation based on eigendecomposition of the data covariance, where the true underlying probability distribution is Gaussian. The second comparison examines the model order estimation using a Gaussian mixture model, which in the limiting case, where the number of Gaussian mixtures is equal to the number of samples, converges to KDE. In all comparisons presented below principal curve projections are obtained by the KDE-SCMS algorithm using the leave-one-out ML kernel bandwidth.



(a) Four dimensional data, coming from four symbols

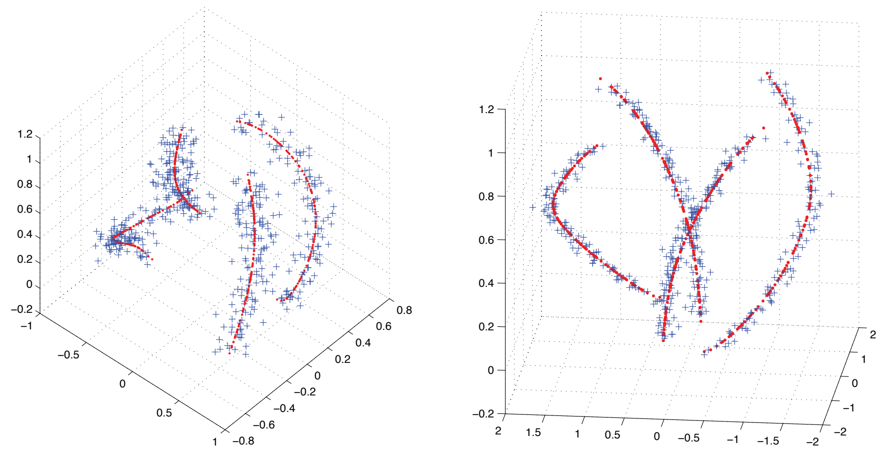


(b) Four dimensional data with time embedding

Figure 16: Symbol clustering problem for a MIMO channel

4.4 Comparison with Eigenvector Estimation

As mentioned before, the reason why we prefer to use KDE is its ability to adapt to different complex shapes that data may take. Indeed, results previously presented in this section show that KDE based principal curve estimation proves to be efficient in adapting to many real-life data distributions of a diverse set of applications. However, one well-known disadvantage of KDE is the required number



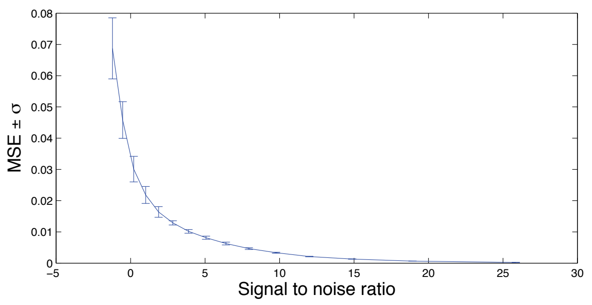


Figure 17: Signal samples and their principal curve projections, normalized MSE vs signal to noise ratio (in dB).

of samples as the dimensionality of the data increases. Here we discuss the case where the true underlying probability density is Gaussian; hence, the claim of *the requirement to adapt to complex shapes in the data* is an obvious overstatement. In this scenario, we will compare the principal line estimator based on PCA to the principal curve based on KDE, for different number of dimensions.

Consider the data set $\{\mathbf{x}_{i=1}^{N}\}$ Gaussian distributed in d-dimensional space, where $\mathbf{v}$ denotes the true principal line of this distribution, and $\mathbf{v}_*$ denotes the principal line obtained by sample PCA. What we are going to compare here is the following:

1. mean squared distance between the projection of the data samples onto the true first eigenvector and the estimated first principal component, $E\{\|\mathbf{v}^T\mathbf{x} - \mathbf{v}_*^T\mathbf{x}\|\}$

2. mean squared distance between the projection of the data samples onto the true eigenvector and the principal curve projection $\widetilde{\mathbf{x}}$, $E\{\|\mathbf{v}^T\mathbf{x} - \widetilde{\mathbf{x}}\|\}$.

Figure 19 presents the MSE of the principal line (dashed curve) and principal curve (solid curve) projections for 2, 3, 4, 5, 10, 20, 30,and 40 dimensions, and average log MSE for 100 Monte Carlo

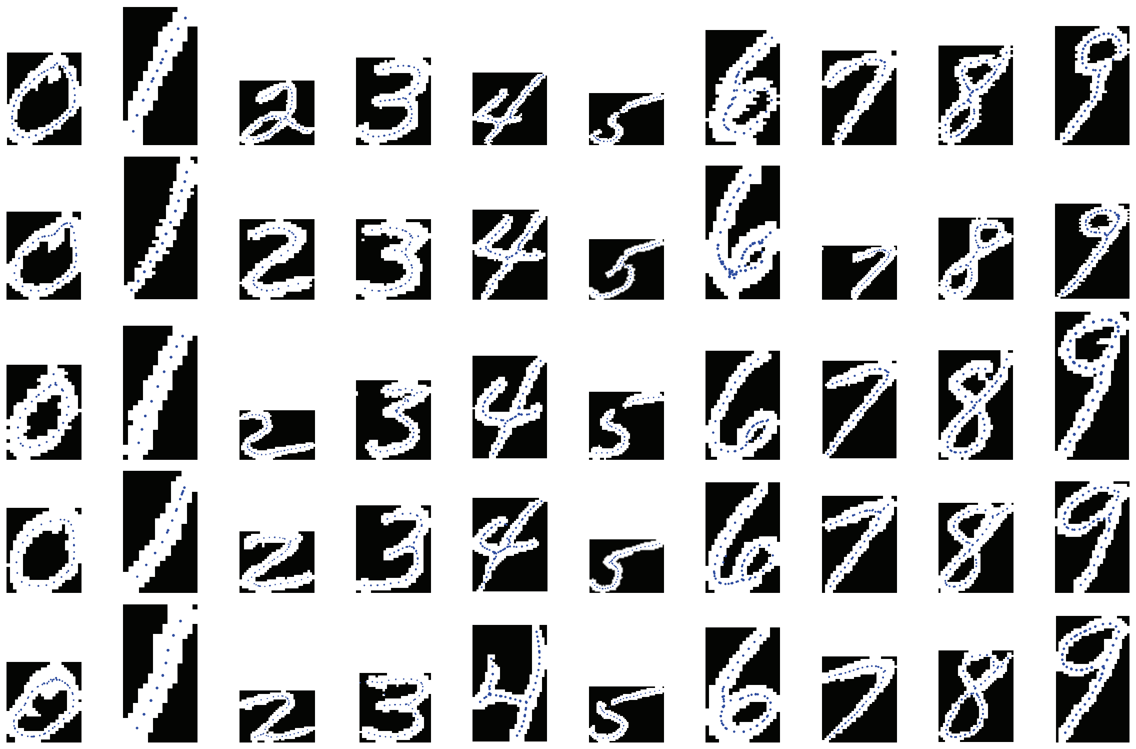


Figure 18: SCMS results in optical characters

simulations is shown. For all cases the MSE decreases for both methods as the number of samples increase. Principal line projection always results in better accuracy and the performance of principal curve projections drop exponentially for increasing dimensions.

4.5 Effects of the Model Order Estimation

An important problem in parametric density estimation is model order selection. In the real applications presented above, we work with KDE-SCMS to provide a general purpose nonparametric algorithm, and to avoid model order selection problems. However, using a parametric model has two main advantages:

1. As opposed to $O(N^2)$ complexity of the KDE-SCMS, the computational complexity of GMM-SCMS is $O(MN)$, where M is the number of mixtures in the Gaussian mixture and N is the number of samples, since typically $M \ll N$.

2. As also implied in the previous section, with the comparison against PCA on a Gaussian data set, a parametric approach with a *suitable* model order, the algorithm would need less samples to achieve good principal curve estimates.

Here we will evaluate the stability of principal curve estimation with GMM-SCMS for improper model order selections in the GMM density estimation step, and compare the principal curve projection results for a Gaussian mixture with 3 components. Since the true underlying density is known to have 3 components, we measure the performance as of principal curve projection results

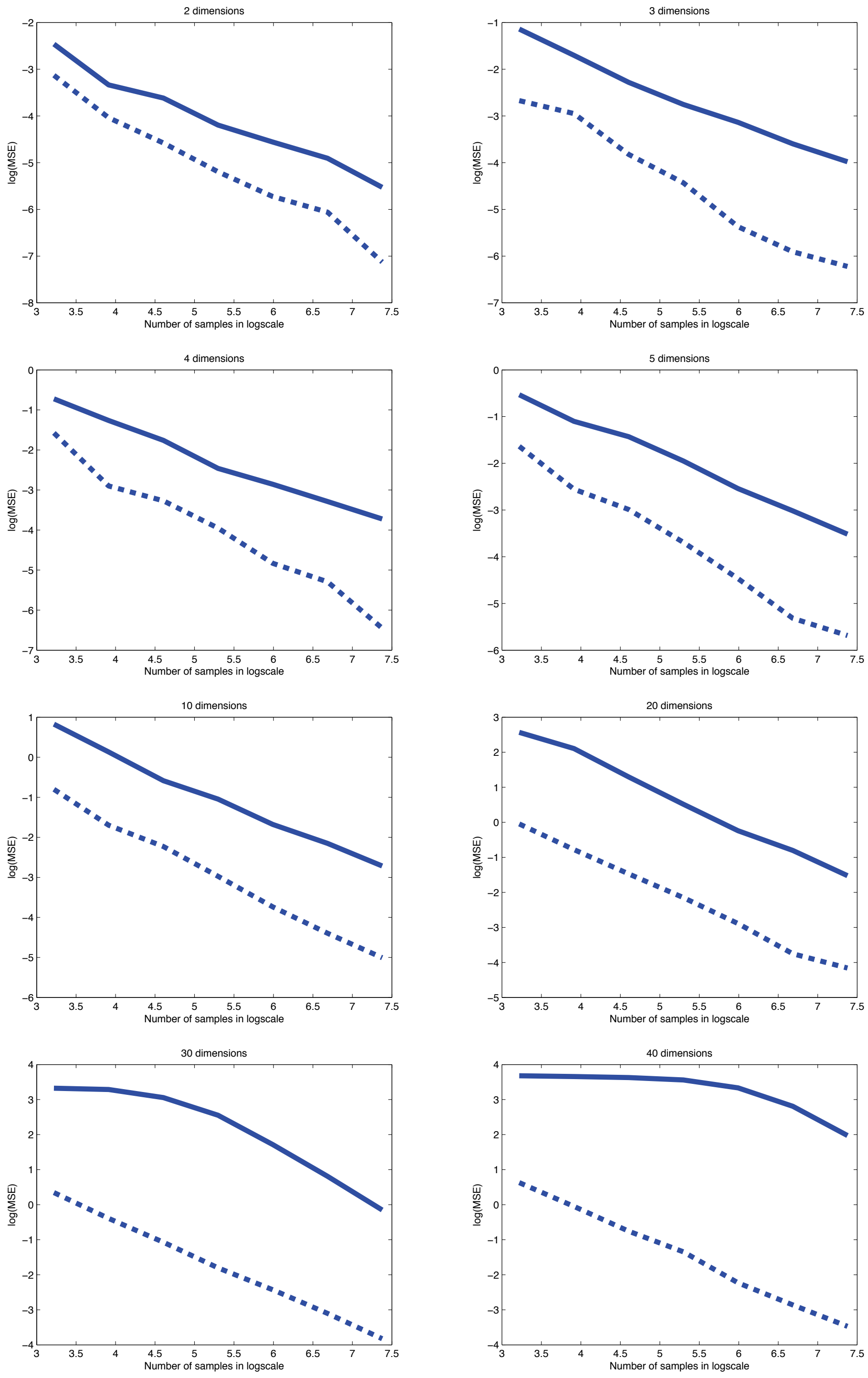


Figure 19: Mean projection error in log_e scale for principal line (dashed) and principal curve (solid). Average of 100 Monte Carlo simulations is shown.

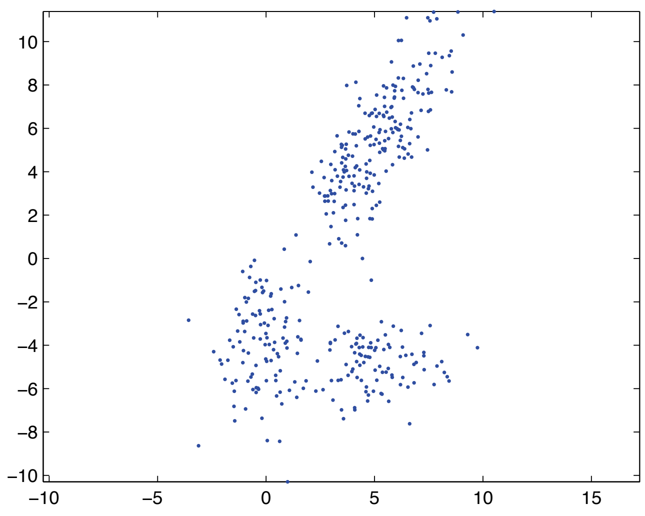


Figure 20: One realization of the 3-component Gaussian mixture data used in performance evaluations

for different number of components in the density estimate as the distance to the principal curve projections obtained with three components

$$\begin{aligned} J_d &= E\{(\tilde{\mathbf{x}}_3(\mathbf{x}) - \tilde{\mathbf{x}}_d(\mathbf{x}))^2\} , \\ where\ d &= 1,2,3,4,5,6,10,15,25,50,100,200,400 . \end{aligned}$$

The data set **x** has 400 samples in 2-dimensional space. Figure 20 shows a realization of the Gaussian mixture, and Figure 21 presents the performance of the principal curve projections for different number of components in the Gaussian mixture estimation, and results of 50 Monte Carlo simulations is shown. Note that for increasing model orders, if the GMM has more number of components than the true underlying distribution, the generalization performance of the principal curve does not change significantly.

5. Discussions

We proposed a novel definition that characterizes the principal curves and surfaces in terms of the gradient and the Hessian of the density estimate. Unlike traditional machine learning papers on manifold learning, which tend to focus on criteria such as reconstruction error of available samples, we focus on the definition of the underlying manifold from a more (differential—though not emphasized here) geometric point of view. There are strong connections between our definition and the literature. If the ridge cross-section is a unimodal and symmetric density, our definition coincides with the original Hastie & Stuetzle definition. There is a strong connection to Kegl's piecewise linear curve proposition, when the underlying density is selected to be a Gaussian mixture. However, the connections are less obvious when considering a principal curve or manifold definition

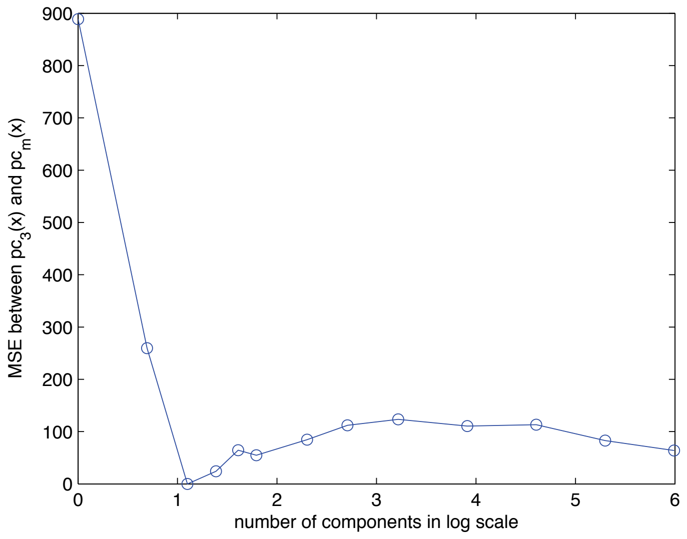


Figure 21: Principal curve projections for different number of components in the density estimate in log_e scale. Specifically, $d = 1, 2, 3, 4, 5, 6, 10, 15, 25, 50, 100, 200, 400$.

that is not explicit (e.g., the principal curve is the solution to some optimization problem without an analytical expression or property).

Providing the definition in terms of the probability density estimate of the data allows us to exclude any smoothness or regularization constraints from the definition, and adopt them from the density estimation literature directly. Although this cannot ultimately answer the question of the trade-off between generalization and overfitting, using the connection to density estimation yields data-driven nonparametric solutions for handling regularization and outlier robustness. In the definition, we also do not assume any parametric model and since the ridges of the pdf can intersect each other, handling self-intersecting data structures requires no additional effort.

An important property of the definition is that it yields a unified framework for clustering, principal curve fitting and manifold learning. Similar to PCA, for an n-dimensional data set, our definition contains all the d-dimensional principal manifolds, where $d < n$. Theoretically, the principal set of $d = 0$ yields the modes of the probability density, which coincides with a widely accepted clustering solution. We accompany this with an algorithmic connection by showing that principal curves can be achieved using the SCMS idea, very similar to the well-known mean-shift clustering algorithm. KDE-based SCMS implementation is significantly faster than the most commonly used method in principal curves literature. Besides, it does not require significantly more time or memory storage as compared to mean shift, which already has been used in many practical application domains.

In high dimensional spaces, density estimation becomes impractical due to the curse of dimensionality. Therefore, similar to existing methods in principal curves literature, the proposed method is not an alternative for proximity graph based manifold learning methods like Isomap, Laplacian eigenmaps etc. Still, we show that there are many real applications in lower dimensional spaces suitable for KDE-based SCMS. We show results on a family of applications in time series signal processing, as well as an earlier proposed application of principal curves (OCR).

Acknowledgments

The authors would like to thank Jose C. Principe, Miguel A. Carreira-Perpinan, Sudhir Rao, Engin Tola, M. Emre Sargin, and Cagri Akargun for valuable discussions, and B. Kegl for providing some of the data sets and implementations of the Polygonal Line algorithm used for the comparisons in the experiments. This work is partially supported by NSF grants ECS-0524835, and ECS-0622239.

References

F. Auger and P. Flandrin. Improving the readability of time-frequency and time-scale representations by the reassignment method. *IEEE Transactions on Signal Processing*, 43(5):1068–1089, May 1995.

J. D. Banfield and A. E. Raftery. Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. *Journal of the American Statistical Association*, 87(417):7–16, 1992.

G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.

R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.

Y. Bengio, H. Larochelle, and P. Vincent. Non-local manifold parzen windows. In *Advances in Neural Information Processing Systems 18*, pages 115–122. MIT Press, 2006.

C. M. Bishop. *Neural Networks for Pattern Recognition, 1st Ed.* Clarendon Press, Oxford, 1997.

M. A. Carreira-Perpinan. Fast nonparametric clustering with gaussian blurring mean-shift. In *ICML '06: Proceedings of the 23rd International Conference on Machine learning*, pages 153–160, New York, NY, USA, 2006. ACM. ISBN 1595933832.

J. E. Chacon, T. Duong, and M. P. Wand. Asymptotics for general multivariate kernel density derivative estimators. *Statistica Sinica, in press*, 2011.

K. Chang and J. Grosh. A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):59–74, 2002.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In R. G. Cowell and Z. Ghahramani, editors, *Proc. of the Tenth Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2005)*, pages 57–64, Barbados, January 6–8 2005.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`.

S. Chen, B. Mulgrew, and P. M. Grant. A clustering technique for digital communications channel equalization using radial basis function networks. *IEEE Transactions on Neural Networks*, 4(4): 570–590, Jul 1993.

Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Analysis and Machine Intelligence,*, 17(8):790–799, 1995.

J. Choi, H. Yu, and Y. H. Lee. Adaptive mimo decision feedback equalization for receivers with time-varying channels. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 53(11):4295–4303, Nov. 2005.

D. Comaniciu. An algorithm for data-driven bandwidth selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):281–288, 2003.

D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

B. N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, (6):793–800, 1934.

P. Delicado. *Principal curves and principal oriented points*. 1998. URL `http://www.econ.upf.es/deehome/what/wpapers/postscripts/309.pdf`.

T. Duchamp and W. Stuetzle. Geometric properties of principal curves in the plane. *Robust Statistics, Data Analysis, and Computer Intensive Methods: In Honor of Peter Huber's 60th Birthday*, 109:135–152, 1996a.

T. Duchamp and W. Stuetzle. Extremal properties of principal curves in the plane. *The Annals of Statistics*, 24(4):1511–1520, 1996b.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

D. Erdogmus and U. Ozertem. Nonlinear coordinate unfolding via principal curve projections with application to bss. In *14th International Conference on Neural Information Processing*, 2007.

L. Fahrmeir and G. Tutz. *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer-Verlag, New York, 1994.

T. S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1 (2):209–230, 1973.

G. J. Foschini, G. D. Golden, R. A. Valenzuela, and P. W. Wolniansky. Simplified processing for high spectral efficiency wireless communication employing multi-element arrays. *IEEE Journal on Selected Areas in Communications*, 17(11):1841–1852, Nov 1999.

K. Fukunaga and D.R. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, 20(2):176–183, 1971.

S. A. Fulop and K. Fitz. Separation of components from impulses in reassigned spectrograms. *Acoustical Society of America Journal*, 121:1510–1517, 2007.

J. Ham, D. Lee, S. Mika, and B. Scholkopf. A kernel view of the dimensionality reduction of manifolds. *In Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 369–376, 2004.

B. E. Hansen. Uniform convergence rates for kernel estimation with dependent data. *Econometric Theory*, 24(03):726–748, June 2008. URL `http://ideas.repec.org/a/cup/etheor/v24y2008i03p726-748_08.html`.

T. Hastie. *Principal Curves and Surfaces*. PhD thesis, Stanford University, 1984.

T. Hastie and W. Stuetzle. Principal curves. *Journal of American Statistical Association*, 84:502–516, 1989.

A. Hegde, J. C. Principe, D. Erdogmus, U. Ozertem, Y. N. Rao, and H. Peddaneni. Perturbation-based eigenvector updates for on-line principal components analysis and canonical correlation analysis. *Journal of VLSI Signal Processing Systems*, 45(1-2):85–95, 2006.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.*, 24, 1933.

J. E. Jackson. *A User's Guide to Principal Components*. John Wiley and Sons, New York, 1991.

I. T. Jolliffe. *Principal Components Analysis*. Springer-Verlag, Berlin, 1986.

M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91(433):401–407, 1996.

K. Fitz and L. Haken and P. Christensen. Transient preservation under transformation in an additive sound model. *Proceedings of International Computer Music Conference*, pages 392–395, 2000.

N. Kambhatla and T. K. Leen. Fast non-linear dimension reduction. In *Neural Information Processing Systems*, pages 152–159, 1994.

N. Kambhatla and T. K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516, 1997.

E. Karami and M. Shiva. Decision-directed recursive least squares mimo channels tracking. *EURASIP Journal of Wireless Communication Networks*, 2006(2):7–7, 2006.

B. Kegl. *Principal Curves: Learning, Design, And Applications*. PhD thesis, Concordia University, Montreal, Canada, 1999.

B. Kegl and A. Kryzak. Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):59–74, 2002.

B. Kegl, A. Kryzak, T. Linder, and K. Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):281–297, 2000.

S. Y. Kung, K. I. Diamantaras, and J. S. Taur. Adaptive principal component extraction (apex) and applications. *IEEE Transactions on Signal Processing*, 42(5):1202–1217, May 1994.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman & Hall/CRC, London, 1989.

P. Meinicke and H. Ritter. Local pca learning with resolution-dependent mixtures of gaussians. *Proceedings of 9th International Conference on Artificial Neural Networks*, pages 497–502, 1999.

A. K. Ozdemir and O. Arikan. A high resolution time frequency representation with significantly reduced cross-terms. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2:693–696, 2000.

A. K. Ozdemir, L. Durak, and O. Arikan. High resolution time-frequency analysis by fractional domain warping. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 6:3553–3556, 2001.

U. Ozertem and D. Erdogmus. Principal curve time warping. *IEEE Transactions on Signal Processing*, 57(6):2041–2049, 2009.

U. Ozertem, D. Erdogmus, and O. Arikan. Piecewise smooth signal denoising via principal curve projections. In *IEEE Int. Conf. on Machine Learning for Signal Processing*, pages 426 – 431, 2008.

E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

V. C. Raykar and R. Duraiswami. Fast optimal bandwidth selection for kernel density estimation. In J. Ghosh, D. Lambert, D. Skillicorn, and J. Srivastava, editors, *Proceedings of the sixth SIAM International Conference on Data Mining*, pages 524–528, 2006.

A.A. Rontogiannis, V. Kekatos, and K. Berberidis. A square-root adaptive v-blast algorithm for fast time-varying mimo channels. *IEEE Signal Processing Letters*, 13(5):265–268, May 2006.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

S. Sandilya and S. R. Kulkarni. Principal curves with bounded turn. *IEEE Transactions on Information Theory*, 48(10):2789–2793, 2002.

B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

John Shawe-Taylor and Yoram Singer, editors. *Regularization and Semi-supervised Learning on Large Graphs*, volume 3120 of *Lecture Notes in Computer Science*, 2004. Springer.

S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(3):683–690, 1991.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, April 1986. ISBN 0412246201.

D. C. Stanford and A. E. Raftery. Finding curvilinear features in spatial point patterns: Principal curve clustering with noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):601–609, 2000.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

R. Tibshirani. Principal curves revisited. *Statistics and Computation*, 2:183–190, 1992.

S. Van Vaerenbergh and I. Santamaría. A spectral clustering approach to underdetermined post-nonlinear blind source separation of sparse sources. *IEEE Transactions on Neural Networks*, 17 (3):811–814, May 2006.

S. Van Vaerenbergh and I. Santamaria. A spectral clustering approach for blind decoding of mimo transmissions over time-correlated fading channels. In E. Hines and M. Martinez, editors, *Intelligent Systems: Techniques and Applications*. Shaker Publishing, 2008.

S. Van Vaerenbergh, E. Estébanez, and I. Santamaría. A spectral clustering algorithm for decoding fast time-varying BPSK MIMO channels. In *15th European Signal Processing Conference*, Poznan, Poland, September 2007.

J. J. Verbeek, N. A. Vlassis, and B. Kröse. A soft k-segments algorithm for principal curves. In *ICANN '01: Proceedings of the International Conference on Artificial Neural Networks*, pages 450–456, London, UK, 2001. Springer-Verlag. ISBN 3-540-42486-5.

J. J. Verbeek, N. Vlassis, and B. Kröse. A k-segments algorithm for finding principal curves. *Pattern Recognition Letters*, 23(8):1009–1017, 2002.

P. Vincent and Y. Bengio. Manifold parzen windows. In *Advances in Neural Information Processing Systems 15*, pages 825–832, 2003.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

A. S. Wong, K. Wong, and C. Wong. A practical sequential method for principal component analysis. *Neural Processing Letters*, 11(2):107–112, 2000.

C. Yang, R. Duraiswami, D. Dementhon, and L. Davis. Mean-shift analysis using quasi-newton methods. In *Proceedings of the International Conference on Image Processing*, pages 447–450, 2003a.

C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis. Improved fast gauss transform and efficient kernel density estimation. In *Ninth IEEE International Conference on Computer Vision*, pages 664–671 vol.1, 2003b.

Journal of Machine Learning Research 12 (2011) 1287-1311 Submitted 8/09; Revised 10/12; Published 4/11

Better Algorithms for Benign Bandits

Elad Hazan EHAZAN@IE.TECHNION.AC.IL
Department of IE&M
Technion–Israel Institute of Technology
Haifa 32000, Israel

Satyen Kale[*] SKALE@YAHOO-INC.COM
Yahoo! Research
4301 Great America Parkway
Santa Clara, CA, USA

Editor: Nicolo Cesa-Bianchi

Abstract

The online multi-armed bandit problem and its generalizations are repeated decision making problems, where the goal is to select one of several possible decisions in every round, and incur a cost associated with the decision, in such a way that the total cost incurred over all iterations is close to the cost of the best fixed decision in hindsight. The difference in these costs is known as the *regret* of the algorithm. The term *bandit* refers to the setting where one only obtains the cost of the decision used in a given iteration and no other information.

A very general form of this problem is the non-stochastic bandit linear optimization problem, where the set of decisions is a convex set in some Euclidean space, and the cost functions are linear. Only recently an efficient algorithm attaining $\tilde{O}(\sqrt{T})$ regret was discovered in this setting.

In this paper we propose a new algorithm for the bandit linear optimization problem which obtains a tighter regret bound of $\tilde{O}(\sqrt{Q})$, where Q is the total variation in the cost functions. This regret bound, previously conjectured to hold in the full information case, shows that it is possible to incur much less regret in a slowly changing environment even in the bandit setting. Our algorithm is efficient and applies several new ideas to bandit optimization such as reservoir sampling.

Keywords: multi-armed bandit, regret minimization, online learning

1. Introduction

Consider a person who commutes to work every day. Each morning, she has a choice of routes to her office. She chooses one route every day based on her past experience. When she reaches her office, she records the time it took her on that route that day, and uses this information to choose routes in the future. She doesn't obtain any information on the other routes she could have chosen to work. She would like to minimize her total time spent commuting in the long run; however, knowing nothing of how traffic patterns might change, she opts for the more pragmatic goal of trying to minimize the total time spent commuting in comparison with the time she would have spent had she full knowledge of the future traffic patterns but had to choose the same fixed route every day. This difference in cost (using time as a metric of cost) measures how much she regrets not knowing traffic patterns and avoiding the hassle of choosing a new path every day.

*. Work done while the author was at Microsoft Research.

This scenario, and many more like it, are modeled by the multi-armed bandit problem and its generalizations. It can be succinctly described as follows: iteratively an online learner has to choose an action from a set of n available actions. She then suffers a cost (or receives a reward) corresponding to the action she took and no other information as to the merit of other available actions. Her goal is to minimize her *regret*, which is defined as the difference between her total cost and the total cost of the best single action knowing the costs of all actions in advance.

Various models of the "unknown" cost functions have been considered in the last half a century. Robbins (1952) pioneered the study of various stochastic cost functions, followed by Hannan (1957), Lai and Robbins (1985) and others. It is hard to do justice to the numerous contributions and studies and we refer the reader to the book of Cesa-Bianchi and Lugosi (2006) for references. In their influential paper, Auer et al. (2003) considered an adversarial non-stochastic model of costs, and gave an efficient algorithm that attains the optimal regret[1] in terms of the number of iterations, T, a bound of $\tilde{O}(\sqrt{T})$.[2] The sublinear (in T) regret bound implies that on average, the algorithm's cost converges to that of the best fixed action in hindsight.

The latter paper (Auer et al., 2003) was followed by a long line of work (Awerbuch and Kleinberg, 2004; McMahan and Blum, 2004; Flaxman et al., 2005; Dani et al., 2008) which considered the more general case of bandit online linear optimization over a convex domain. In this problem, the learner has to choose a sequence of points from the convex domain and obtains their cost from an unknown linear cost function. The objective, again, is to minimize the regret, that is, the difference between the total cost of the algorithm and that of the best fixed point in hindsight. This generality is crucial to allow for *efficient* algorithms for problems with a large decision space, such as online shortest path problem considered at the beginning. This line of work finally culminated in the work of Abernethy et al. (2008), who obtained the first algorithm to give $\tilde{O}(\sqrt{T})$ regret with polynomial running time.

Even though the $\tilde{O}(\sqrt{T})$ dependence on T was a great achievement, this regret bound is weak from the point of view of real-world bandit scenarios. Rarely would we encounter a case where the cost functions are truly adversarial. Indeed, the first work on this problem assumed a stochastic model of cost functions, which is a very restrictive assumption in many cases. One reasonable way to retain the appeal of worst-case bounds while approximating the steady-state nature of the stochastic setting is to consider the *variation* in the cost vectors.

For example, our office commuter doesn't expect the traffic gods to conspire against her every day. She might expect a certain predictability in traffic patterns. Most days the traffic pattern is about the same, except for some fluctuations depending on the day of the week, time of the day, etc. Coming up with a stochastic model for traffic patterns would be simply too onerous. An algorithm that quickly learns the dominant pattern of the traffic, and achieves regret bounded by the (typically small) variability in day-to-day traffic, would be much more desirable. Such regret bounds naturally interpolate between the stochastic models of Robbins and the worst case models of Auer *et al.*

In this paper[3] we present the first such bandit optimization algorithm in the worst-case adversarial setting, with regret bounded by $\tilde{O}(\sqrt{Q})$, where Q is the total observed variation in observed costs, defined as the sum of squared deviations of the cost vectors from their mean. This regret

1. Strictly speaking, here we talk about *expected* regret, as all algorithms that attain non-trivial guarantees must use randomization.
2. We use the notation $\tilde{O}$ to hide all constant terms (such as dependence on the dimension of the problem, or the diameter of the decision set) and other lower order terms which grow at a poly-logarithmic rate with T.
3. A preliminary version of this result was presented in Hazan and Kale (2009a).

degrades gracefully with increasing Q, and in the worst case, we recover the regret bound $\tilde{O}(\sqrt{T})$ of Abernethy et al. (2008). Our algorithm is efficient, running in polynomial time per iteration.

The conjecture that the regret of online learning algorithms should be bounded in terms of the total variation was put forth by Cesa-Bianchi et al. (2007) in the full information model (where the online player is allowed to observe the costs of actions she did not choose). This conjecture was recently resolved on the affirmative in Hazan and Kale (2008), in two important online learning scenarios, viz. online linear optimization and expert prediction. In addition, in Hazan and Kale (2009b), we give algorithms with regret bounds of $O(\log(Q))$ for the Universal Portfolio Selection problem and its generalizations. In this paper, we prove the surprising fact that such a regret bound of $\tilde{O}(\sqrt{Q})$ is possible to obtain even when the only information available to the player is the cost she incurred (in particular, we may not even be able to estimate Q accurately in this model).

To prove our result we need to overcome the following difficulty: all previous approaches for the non-stochastic multi-armed bandit problem relied on the main tool of "unbiased gradient estimator", that is, the use of randomization to extrapolate the missing information (cost function). The variation in these unbiased estimators is unacceptably large even when the underlying cost function sequence has little or no variation.

To overcome this problem we introduce two new tools: first, we use historical costs to construct our gradient estimators. Next, in order to construct these estimators, we need an accurate method of accumulating historical data. For this we use a method from data streaming algorithms known as "reservoir sampling". This method allows us to maintain an accurate "sketch" of history with very little overhead.

An additional difficulty which arises is the fact that a learning rate parameter η needs to be set based on the total variation Q to obtain the $\tilde{O}(\sqrt{Q})$ regret bound. Typically, in other scenarios where square root regret bound in some parameter is desired, a simple η-halving trick works, but requires the algorithm to be able to compute the relevant parameter after every iteration. However, as remarked earlier, even estimating Q is non-trivial problem. We do manage to bypass this problem by using a novel approach that implicitly mimics the η-halving procedure.

2. Preliminaries

Throughout the paper we use the standard asymptotic $O()$ notation to hide dependence on (absolute, problem-independent) constants. For convenience of notation, we use the $\tilde{O}()$ notation to hide dependence on (problem-dependent) constants as well as polylog(T) factors: $g = \tilde{O}(f)$ if $g < cf\log^d(T)$ for some problem-dependent constant $c > 0$ and and a problem-independent constant $d > 0$. Specifically, in the $\tilde{O}()$ notation we also hide terms which depend on the dimension n, since this is fixed: we use this notation in order highlight the dependence on the parameter which grows with time, viz. the quadratic variation. Unless specified otherwise, all vectors live in $\mathbb{R}^n$, and all matrices in $\mathbb{R}^{n\times n}$. The vector norm $\|\cdot\|$ denotes the standard ℓ_2 norm.

We consider the online linear optimization model in which iteratively the online player chooses a point $\mathbf{x}_t \in \mathcal{K}$, where $\mathcal{K} \subseteq \mathbb{R}^n$ is a convex compact set called the *decision set*. After her choice, an adversary supplies a linear cost function $\mathbf{f}_t$, and the player incurs a cost of $\mathbf{f}_t(\mathbf{x}_t)$. In this paper we assume an *oblivious* adversary, which can choose arbitrary cost functions $\mathbf{f}_t$ in advance, with prior knowledge of the player's algorithm, but the adversary does not have access to the random bits used by the player (see Dani and Hayes, 2006 for more details on various models of adversaries).

With some abuse of notation, we use $\mathbf{f}_t$ to also denote the cost vector such that $\mathbf{f}_t(\mathbf{x}) = \mathbf{f}_t^\top \mathbf{x}$. The *only* information available to the player is the cost incurred, that is, the scalar $\mathbf{f}_t(\mathbf{x}_t)$. Denote the total number of game iterations by T. The standard game-theoretic measure of performance is regret, defined as

$$\text{Regret}_T \;=\; \sum_{t=1}^{T} \mathbf{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{K}} \sum_{t=1}^{T} \mathbf{f}_t(\mathbf{x}).$$

We make some normalizations on the cost vectors and the convex domain $\mathcal{K}$ to keep the presentation clean. We assume that the cost vectors are scaled so that their norms are bounded by one, that is, $\|\mathbf{f}_t\| \leq 1$. This scaling only changes the regret bound by a constant factor: if G is a known upper bound on the norms of the cost vectors, we can scale down the cost vectors by G and run the algorithm; the actual regret is G times larger than the bound obtained here. Next, we assume $\mathcal{K}$ is scaled to fit inside the unit ball (in the ℓ_2 norm) centered at the origin, that is, for all $\mathbf{x} \in \mathcal{K}$, we have $\|\mathbf{x}\| \leq 1$. We also assume that for some parameter $\gamma \in (0,1)$, all the vectors $\gamma\mathbf{e}_1, \ldots, \gamma\mathbf{e}_n$, where $\mathbf{e}_i$ is the standard basis vector with 1 in the i-th coordinate and 0 everywhere else, are in the decision set $\mathcal{K}$.

The above assumptions can be met by translating and scaling $\mathcal{K}$ appropriately. This only changes the regret bound by a constant factor: if D is a known upper bound on the diameter of $\mathcal{K}$, then we can translate $\mathcal{K}$ so that it contains the origin and scaled coordinate vectors $\gamma'\mathbf{e}_i$ for some $\gamma' > 0$, and then scale it down by D to make its diameter 1 and run the algorithm; the actual regret is D times larger than the bound obtained here. In certain specific cases, one can certainly obtain tighter constants by using a set of n linearly independent vectors contained inside $\mathcal{K}$, but here we make this simplifying assumption for the sake of cleanliness of presentation.

We denote by Q_T the total quadratic variation in cost vectors, that is,

$$Q_T \;:=\; \sum_{t=1}^{T} \|\mathbf{f}_t - \mu\|^2,$$

where $\mu = \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}_t$ is the mean of all cost vectors.

A symmetric matrix $\mathbf{A}$ is called positive semidefinite (denoted by $\mathbf{A} \succeq \mathbf{0}$ if all its eigenvalues are non-negative. If all eigenvalues of $\mathbf{A}$ are strictly positive, the matrix is called positive definite. For symmetric matrices we denote by $\mathbf{A} \preceq \mathbf{B}$ the fact that the matrix $\mathbf{B} - \mathbf{A}$ is positive semidefinite. For a positive definite matrix $\mathbf{A}$ we denote its induced norm by $\|\mathbf{x}\|_{\mathbf{A}} = \sqrt{\mathbf{x}^\top \mathbf{A}\mathbf{x}}$. We make use of the following simple generalization of the Cauchy-Schwarz inequality:

$$\mathbf{x}^\top \mathbf{y} \;\leq\; \|\mathbf{x}\|_{\mathbf{A}} \cdot \|\mathbf{y}\|_{\mathbf{A}^{-1}}. \tag{1}$$

This inequality follows by applying the usual Cauchy-Schwarz inequality to the vectors $\mathbf{A}^{1/2}\mathbf{x}$ and $(\mathbf{A}^{1/2})^{-1}\mathbf{y}$, where $\mathbf{A}^{1/2}$ is the matrix square root of the positive definite matrix $\mathbf{A}$, that is, a matrix $\mathbf{B}$ which satisfies $\mathbf{B}\mathbf{B} = \mathbf{A}$.

For a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, we denote its gradient by ∇f and its Hessian by $\nabla^2 f$.

2.1 Reservoir Sampling

A crucial ingredient in our algorithm is a sampling procedure ubiquitously used in streaming algorithms known as "reservoir sampling" (Vitter, 1985). In a streaming problem the algorithm gets to see a stream of data in one pass, and not allowed to re-visit previous data. Suppose the elements of the stream are real numbers $\mathbf{f}_1, \mathbf{f}_2, \ldots$ and our goal is to maintain a randomized estimate of the current mean $\mu_t := \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{f}_\tau$. The main constraint which precludes the trivial solution is that we desire a sampling scheme that touches (i.e., observes the value of) very few elements in the stream.

The reservoir sampling method is to maintain a randomly chosen (without replacement) subset S of size k (also called a "reservoir") from the stream, and then use the average of the sample as an estimator. This works as follows. Initialize S by including the first k elements $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k$ in the stream. For every subsequent element $\mathbf{f}_t$, we decide to include it in S with probability $\frac{k}{t}$. If the decision to include it is made, then a random element of S is replaced by $\mathbf{f}_t$.

The following lemma is standard (see Vitter, 1985) and we include a simple inductive proof for completeness:

Lemma 1 *For every $t \geq k$, the set S is random subset chosen without replacement uniformly from $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_t\}$.*

Proof We prove this by induction on t. The statement is trivially true for $t = k$. Assume that the statement is true for some $t \geq m$, and we now show $t+1$. Let S be an arbitrary subset of size k of $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_t\}$. We now show that the probability that the chosen set in the $t+1$-th round is S is $\frac{1}{\binom{t+1}{k}}$. For this, we have two cases: $\mathbf{f}_{t+1} \notin S$ and $\mathbf{f}_{t+1} \in S$. In the first case, the probability that S is the chosen subset at the end of t-th round is $\frac{1}{\binom{t}{k}}$ by the induction hypothesis. The conditional probability that it survives the $t+1$-th is $1 - \frac{k}{t+1}$, so the overall probability that S is the chosen set at the end of $t+1$-th round is $(1 - \frac{k}{t+1}) \cdot \frac{1}{\binom{t}{k}} = \frac{1}{\binom{t+1}{k}}$.

In the second case, S will be the chosen set at the end of $t+1$-th round if the set S' chosen at the end of the t-th round is one of the $t+1-k$ sets obtained from S by replacing $\mathbf{f}_{t+1}$ by an element of $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_t\} \setminus S$, which gets replaced by $\mathbf{f}_{t+1}$ in the $t+1$-th round. The probability of this happening is $\frac{t+1-k}{\binom{t}{k}} \cdot \frac{k}{t+1} \cdot \frac{1}{k} = \frac{1}{\binom{t+1}{k}}$, as required. ■

Now suppose we define $\tilde{\mu}_t$ to be the average of the k chosen numbers in S, then the following lemma is immediate:

Lemma 2 *For every $t \geq k$, we have $\mathbf{E}[\tilde{\mu}_t] = \mu_t$ and $VAR[\tilde{\mu}_t] \leq \frac{1}{kt}\sum_{\tau=1}^{t}(\mathbf{f}_\tau - \mu_t)^2 = \frac{1}{kt}Q_t$.*

The bound on the variance follows because the variance of a single randomly chosen element of the stream is $\frac{1}{t}\sum_{\tau=1}^{t}(\mathbf{f}_\tau - \mu_t)^2$. So the variance of the average of k randomly chosen elements *with* replacement is $\frac{1}{kt}\sum_{\tau=1}^{t}(\mathbf{f}_\tau - \mu_t)^2$. Since we choose the k elements in the sample *without* replacement, the variance is only smaller.

The main reason reservoir sampling is useful in our context is because it samples every element *obliviously*, that is, a decision to sample is made without looking at the element. This implies that the expected number of elements touched by the reservoir sampling based estimation procedure for μ_t is $k + \sum_{t=k+1}^{T}\frac{k}{t} = O(k\log(T))$, which is very small compared to the length of the stream, T, if k is set to some small value, like $O(\log(T))$ as in our applications.

2.2 Self-concordant Functions and the Dikin Ellipsoid

In this section we give a few definition and properties of self-concordant barriers that we will crucially need in the analysis. Our treatment of this subject follows Abernethy et al. (2008), who introduced self-concordant functions to online learning. Self-concordance in convex optimization is a beautiful and deep topic, and we refer the reader to Nesterov and Nemirovskii (1994) and Ben-Tal and Nemirovski (2001) for a thorough treatment on the subject.

Definition 1 *A convex function $\mathcal{R}(\mathbf{x})$ defined on the interior of the convex compact set $\mathcal{K}$, and having three continuous derivatives, is said to be a ϑ-self-concordant barrier (where $\vartheta > 0$ is the self-concordance parameter) if the following conditions hold:*

1. *(Barrier property) $\mathcal{R}(\mathbf{x}_i) \to \infty$ along every sequence of points $\mathbf{x}_i$ in the interior of $\mathcal{K}$ converging to a boundary point of $\mathcal{K}$.*
2. *(Differential properties) $\mathcal{R}$ satisfies*

$$|\nabla^3 \mathcal{R}(\mathbf{x})[\mathbf{h},\mathbf{h},\mathbf{h}]| \;\le\; 2(\mathbf{h}^\top [\nabla^2 \mathcal{R}(\mathbf{x})]\mathbf{h})^{3/2},$$

$$|\nabla \mathcal{R}(\mathbf{x})^\top \mathbf{h}| \;\le\; \vartheta^{1/2} \left[\mathbf{h}^\top \nabla^2 \mathcal{R}(\mathbf{x}) \mathbf{h}\right]^{1/2}.$$

where $\mathbf{x}$ is a point in the interior of $\mathcal{K}$, and $\mathbf{h}$ is an arbitrary vector in $\mathbb{R}^n$. Here, $\nabla \mathcal{R}(\mathbf{x}), \nabla^2 \mathcal{R}(\mathbf{x})$ denote the Gradient and Hessian, respectively, of $\mathcal{R}$ at point $\mathbf{x}$, and

$$\nabla^3 \mathcal{R}(\mathbf{x})[\mathbf{h}_1,\mathbf{h}_2,\mathbf{h}_3] \;=\; \frac{\partial^3}{\partial t_1 \partial t_2 \partial t_3} \mathcal{R}(\mathbf{x} + t_1\mathbf{h}_1 + t_2\mathbf{h}_2 + t_3\mathbf{h}_3)\bigg|_{t_1=t_2=t_3=0}$$

Any n-dimensional closed convex set admits an $O(n)$-self-concordant barrier. However, such a barrier may not necessarily be efficiently computable.

More concretely, the standard logarithmic barrier for a half-space $\mathbf{u}^\top \mathbf{x} \le b$ is given by

$$\mathcal{R}(\mathbf{x}) \;=\; -\log(b - \mathbf{u}^\top \mathbf{x}),$$

and is 1-self-concordant. For polytopes defined by m halfspaces, the standard logarithmic barrier (which is just the sum of all barriers for the defining half-spaces) has the self-concordance parameter $\vartheta = m$.

Definition 2 *For a given $\mathbf{x} \in \mathcal{K}$, and any $\mathbf{h} \in \mathbb{R}^n$, define the norm induced by the Hessian, and its dual norm, to be*

$$\|\mathbf{h}\|_{\mathbf{x}} := \sqrt{\mathbf{h}^\top [\nabla^2 \mathcal{R}(\mathbf{x})]\mathbf{h}}, \;\; \textit{and}$$
$$\|\mathbf{h}\|_{\mathbf{x}}^\star := \sqrt{\mathbf{h}^\top [\nabla^2 \mathcal{R}(\mathbf{x})]^{-1}\mathbf{h}}.$$

Definition 3 *The* Dikin ellipsoid *of radius r centered at $\mathbf{x}$ is the set*

$$W_r(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}} \le r\}.$$

When a radius is unspecified, it is assumed to be 1; so "the Dikin ellipsoid at $\mathbf{x}$" refers to the Dikin ellipsoid of radius 1 centered at $\mathbf{x}$.

Definition 4 *For any two distinct points* $\mathbf{x}$ *and* $\mathbf{y}$ *in the interior of* $\mathcal{K}$*, the* Minkowsky function $\pi_{\mathbf{x}}(\mathbf{y})$ *on* $\mathcal{K}$ *is*

$$\pi_{\mathbf{x}}(\mathbf{y}) = \inf\{t \geq 0 : \mathbf{x} + t^{-1}(\mathbf{y} - \mathbf{x}) \in \mathcal{K}\}.$$

The Minkowsky function measures the distance from $\mathbf{x}$ to $\mathbf{y}$ as a portion of the total distance on the ray from $\mathbf{x}$ to the boundary of $\mathcal{K}$ through the point $\mathbf{y}$, and hence $\pi_{\mathbf{x}}(\mathbf{y}) \in [0,1]$.

The following facts about the Dikin ellipsoid and self concordant barriers will be used in the sequel (we refer to Nemirovskii, 2004 for proofs):

1. $W_1(\mathbf{x}) \subseteq \mathcal{K}$ for any $\mathbf{x} \in \mathcal{K}$. This is crucial for most of our sampling steps (the "ellipsoidal sampling"), since we sample from the Dikin ellipsoid $W_1(\mathbf{x}_t)$. Since $W_1(\mathbf{x}_t)$ is contained in $\mathcal{K}$, the sampling procedure yields feasible points.

2. The lengths of the principal axes of the ellipsoid $W_1(\mathbf{x})$ are $2/\sqrt{\lambda_i}$, where λ_i, for $i = 1, 2, \ldots, n$ are the eigenvalues of $\nabla^2\mathcal{R}(\mathbf{x})$. Thus, the fact that $W_1(\mathbf{x}) \subseteq \mathcal{K}$ and that $\mathcal{K}$ is contained in the unit ball implies that $2/\sqrt{\lambda_i} \leq 2$ for all i, or in other words, $1/\lambda_i \leq 1$ for all i. This implies that $[\nabla^2\mathcal{R}(\mathbf{x})]^{-1} \leq \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Thus, we can relate the $\|\cdot\|_{\mathbf{x}}^{\star}$ norm to the standard ℓ_2 norm $\|\cdot\|$: for any vector $\mathbf{h}$,

$$\|\mathbf{h}\|_{\mathbf{x}}^{\star} = \sqrt{\mathbf{h}^{\top}[\nabla^2\mathcal{R}(\mathbf{x})]^{-1}\mathbf{h}} \leq \sqrt{\mathbf{h}^{\top}\mathbf{I}\mathbf{h}} = \|\mathbf{h}\|. \tag{2}$$

3. In the interior of the Dikin ellipsoid at $\mathbf{x}$, the Hessian of $\mathcal{R}$ is "almost constant": for any $\mathbf{h} \in \mathbb{R}^n$ such that $\|\mathbf{h}\|_{\mathbf{x}} < 1$, we have

$$(1 - \|\mathbf{h}\|_{\mathbf{x}})^2\nabla^2\mathcal{R}(\mathbf{x}) \preceq \nabla^2\mathcal{R}(\mathbf{x}+\mathbf{h}) \preceq (1 - \|\mathbf{h}\|_{\mathbf{x}})^{-2}\nabla^2\mathcal{R}(\mathbf{x}).$$

4. For any ϑ-self-concordant barrier on $\mathcal{K}$, and for any two distinct points $\mathbf{x}$ and $\mathbf{y}$ in the interior of $\mathcal{K}$, it holds that (see Nemirovskii, 2004)

$$\mathcal{R}(\mathbf{y}) - \mathcal{R}(\mathbf{x}) \leq \vartheta \ln\left(\frac{1}{1 - \pi_{\mathbf{x}}(\mathbf{y})}\right). \tag{3}$$

Definition 5 *Let* $\mathbf{x}^{\circ}$ *be the analytic center of* $\mathcal{K}$ *with respect to the self-concordant barrier* $\mathcal{R}$*, that is, the point inside* $\mathcal{K}$ *in which* $\nabla\mathcal{R}(\mathbf{x}^{\circ}) = 0$*. For any* $\delta > 0$*, define the convex body* $\mathcal{K}_{\delta} \subseteq \mathcal{K}$ *by*

$$\mathcal{K}_{\delta} := \{\mathbf{x} | \pi_{\mathbf{x}^{\circ}}(\mathbf{x}) \leq (1 - \delta)\}.$$

The following properties holds for $\mathcal{K}_{\delta}$:

Lemma 3 *For any* $\mathbf{x} \in \mathcal{K}$*, there exists a* $\mathbf{u} \in \mathcal{K}_{\delta}$ *such that* $\|\mathbf{x} - \mathbf{u}\| \leq 2\delta$ *which satisfies*

$$\mathcal{R}(\mathbf{u}) - \mathcal{R}(\mathbf{x}^{\circ}) \leq \vartheta \ln\frac{1}{\delta}.$$

Proof Let $\mathbf{u} = \delta\mathbf{x}^{\circ} + (1-\delta)\mathbf{x}$. Since

$$\mathbf{x} = \mathbf{x}^{\circ} + \frac{\|\mathbf{x} - \mathbf{x}^{\circ}\|}{\|\mathbf{u} - \mathbf{x}^{\circ}\|}(\mathbf{u} - \mathbf{x}^{\circ}),$$

and $\mathbf{x} \in \mathcal{K}$, we have that

$$\pi_{\mathbf{x}^\circ}(\mathbf{u}) \leq \frac{\|\mathbf{u} - \mathbf{x}^\circ\|}{\|\mathbf{x} - \mathbf{x}^\circ\|} = \frac{\|(1-\delta)(\mathbf{x} - \mathbf{x}^\circ)\|}{\|\mathbf{x} - \mathbf{x}^\circ\|} = 1 - \delta,$$

which implies that $\mathbf{u} \in \mathcal{K}_\delta$. Next, note that

$$\|\mathbf{x} - \mathbf{u}\| = \|\delta(\mathbf{x} - \mathbf{x}^\circ)\| \leq 2\delta,$$

since $\mathcal{K}$ is contained in the unit ball.

Finally, since $\mathcal{R}$ is a ϑ-self-concordant barrier, we have by (3)

$$\mathcal{R}(\mathbf{u}) - \mathcal{R}(\mathbf{x}^\circ) \leq \vartheta \log \frac{1}{1 - \pi_{\mathbf{x}^\circ}(\mathbf{u})} \leq \vartheta \log \frac{1}{1-(1-\delta)} = \vartheta \ln \frac{1}{\delta}.$$

■

3. The Main Theorem and Algorithm

Main result. Before describing the algorithm, let us state the main result of this paper formally.

Theorem 4 *Let $\mathcal{K}$ be the underlying decision set in an online linear optimization instance, such that $\mathcal{K}$ admits an efficiently computable ϑ-self-concordant barrier. Then there exists a polynomial time algorithm for this online linear optimization problem (Algorithm 1 below coupled with the halving procedure of Section 5) whose expected regret is bounded as follows. Let Q_T be the total variation of a cost function sequence in the online linear optimization instance. Then*

$$\mathbf{E}[Regret_T] = O\left(n\sqrt{\vartheta Q_T \log T} + n\log^2 T + n\vartheta \log(T)\right).$$

This theorem can be used with the well known logarithmic barrier to derive regret bounds for the online-shortest-paths problem and other linearly constrained problems, and of course applicable much more generally.

The non-stochastic multi-armed bandit problem. A case of particular interest, which has been studied most extensively, is the "basic" multi-armed bandit (MAB) problem where in each iteration the learner pulls the arm of one out of n slot machines and obtains an associated reward, assumed to be in the range $[0,1]$. The learner's objective is to minimize his regret, that is, the difference between his cumulative reward and that of the best fixed slot machine in hindsight.

This is a special case of the more general problem considered earlier and corresponds to taking the convex set $\mathcal{K}$ to be the n-dimensional simplex of probability distributions over the arms. Since the n-dimensional simplex admits a simple n-self-concordant barrier, an immediate corollary of our main theorem is:

Corollary 5 *There exists an efficient algorithm for the multi-armed-bandit problem whose expected regret is bounded by*

$$\mathbf{E}[Regret_T] = O\left(n^2\sqrt{Q_T \log(T)} + n^{1.5}\log^2(T) + n^{2.5}\log(T)\right).$$

The additional factor of $\sqrt{n}$ factor is because our results assume that $\|\mathbf{f}_t\| \leq 1$, and so we need to scale the costs down by $\sqrt{n}$ to apply our bounds. In comparison, the best previously known bounds for this problem is $O(\sqrt{nT})$ (Audibert and Bubeck, 2010). Even though our bound is worse in the dependence on n, the dependence on the parameter which grows, viz. T, is much better.

3.1 Overview of the Algorithm

The underlying scheme of our algorithm follows the recent approach of Abernethy et al. (2008), who use the Follow-The-Regularized-Leader (FTRL) methodology with self concordant barrier functions as a regularization (see also exposition in Hazan and Kale 2008). At the top level, at every iteration this algorithm simply chooses the point that would have minimized the total cost so far, including an additional regularization cost function $\mathcal{R}(\mathbf{x})$, that is, we predict with the point

$$\mathbf{x}_t = \arg\min_{\mathcal{K}} \left[\eta \sum_{\tau=1}^{t-1} \tilde{\mathbf{f}}_\tau^\top(\mathbf{x}) + \mathcal{R}(\mathbf{x}) \right],$$

where η is a learning rate parameter.

Here, $\tilde{\mathbf{f}}_t$ is an estimator for the vector $\mathbf{f}_t$, which is carefully constructed to have low variance. In the full-information setting, when we can simply set $\tilde{\mathbf{f}}_t = \mathbf{f}_t$, such an algorithm can be shown to achieve low regret (see exposition in Abernethy et al., 2008 and references therein). In the bandit setting, a variety of "one-point-gradient-estimators" are used (Flaxman et al., 2005; Abernethy et al., 2008) which produce an unbiased estimator $\tilde{\mathbf{f}}_t$ of $\mathbf{f}_t$ by evaluating $\mathbf{f}_t$ at just one point.

In order to obtain variation based bounds on the regret, we modify the unbiased estimators of previous approaches by incorporating our experience with previous cost vector as a "prior belief" on the upcoming cost vector. Essentially, we produce an unbiased estimator of the *difference between the average cost vector in the past and the current cost vector*.

This brings out the issue that the past cost vectors are unfortunately also unknown. However, since we had many opportunities to learn about the past and it is an aggregate of many functions, our knowledge about the past cumulative cost vector is much better than the knowledge of any one cost vector in particular. We denote by $\tilde{\mu}_t$ our estimator of $\frac{1}{t}\sum_{\tau=1}^{t} \mathbf{f}_\tau$. The straightforward way of maintaining this estimator would be to average all previous estimators $\tilde{\mathbf{f}}_t$. However, this estimator is far from being sufficiently accurate for our purposes.

Instead, we use the reservoir sampling idea of Section 2.1 to construct this $\tilde{\mu}_t$. For each coordinate $i \in [n]$, we maintain a reservoir of size k, $S_{i,1}, S_{i,2}, \ldots, S_{i,k}$. The estimator for $\mu_t(i)$ is then $\tilde{\mu}_t(i) = \frac{1}{k}\sum_{j=1}^{k} S_{i,j}$. The first nk rounds we devote to initialize these reservoirs with samples from the stream. This increases the overall regret of our algorithm by a constant of nk.

Our current approach is to use separate exploration steps in order to construct $\tilde{\mu}_t$. While it is conceivable that there are more efficient methods of integrating exploration and exploitation, as done by the algorithm in the other iterations, reservoir sampling turns out to be extremely efficient and incurs only a logarithmic penalty in regret.

The general scheme is given in Algorithm 1. It is composed of exploration steps, called SIMPLEXSAMPLE steps, and exploration-exploitation steps, called ELLIPSOIDSAMPLE steps. Note that we use the notation $\mathbf{y}_t$ for the actual point in $\mathcal{K}$ chosen by the algorithm in either of these steps.

It remains to precisely state the SIMPLEXSAMPLE and ELLIPSOIDSAMPLE procedures. The SIMPLEXSAMPLE procedure is the simpler of the two. It essentially performs reservoir sampling on all the coordinates with a reservoir of size k. The initial nk time iterations are used to initialize this reservoir to have correct expectation (i.e., in these iterations we sample with probability one and fill all buckets S_{ij}), and incur an additional additive regret of at most nk.

Now, the SIMPLEXSAMPLE procedure is invoked with probability $\frac{nk}{t}$ for any time period $t > nk$. Once invoked, it samples a coordinate $i_t \in [n]$ with the uniform distribution. The point $\mathbf{y}_t$ chosen

Algorithm 1 Bandit Online Linear Optimization

```
1: Input: $\eta > 0$, $\vartheta$-self-concordant $\mathcal{R}$, reservoir size parameter $k$
2: Initialization: for all $i \in [n], j \in [k]$, set $S_{i,j} = 0$. Set $\mathbf{x}_1 = \arg\min_{\mathbf{x}\in\mathcal{K}}[\mathcal{R}(\mathbf{x})]$ and $\tilde{\mu}_0 = 0$. Let
   $\pi : \{1, 2 \dots, nk\} \to \{1, 2, \dots, nk\}$ be a random permutation.
3: for $t = 1$ to $T$ do
4:   Set $r = 1$ with probability $\min\left\{\frac{nk}{t}, 1\right\}$, and 0 with probability $1 - \min\left\{\frac{nk}{t}, 1\right\}$.
5:   if $r = 1$ then
6:     // SIMPLEXSAMPLE step
7:     if $t \le nk$ then
8:       Set $i_t = (\pi(t) \mod n) + 1$.
9:     else
10:      Set $i_t$ uniformly at random from $\{1, 2, \dots, n\}$.
11:    end if
12:    Set $\tilde{\mu}_t \leftarrow$ SIMPLEXSAMPLE$(i_t)$.
13:    Set $\tilde{\mathbf{f}}_t = 0$.
14:  else
15:    // ELLIPSOIDSAMPLE step
16:    Set $\tilde{\mu}_t = \tilde{\mu}_{t-1}$.
17:    Set $\tilde{\mathbf{f}}_t \leftarrow$ ELLIPSOIDSAMPLE$(\mathbf{x}_t, \tilde{\mu}_t)$.
18:  end if
19:  Update $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{K}} \underbrace{\left[\eta \sum_{\tau=1}^{t} \tilde{\mathbf{f}}_\tau^\top \mathbf{x} + \mathcal{R}(\mathbf{x})\right]}_{\Phi_t(\mathbf{x})}$
20: end for
```

by the algorithm is the corresponding vertex $\gamma\mathbf{e}_{i_t}$ of the (γ-scaled) n-dimensional simplex (which is assumed to be contained inside of $\mathcal{K}$) to obtain the coordinate $\mathbf{f}_t(i_t)$ as the cost.

It then chooses one of the samples $S_{i_t,1}, S_{i_t,2}, \dots, S_{i_t,k}$ uniformly at random and replaces it with the value $\mathbf{f}_t(i_t)$, and updates $\tilde{\mu}_t$. This exactly implements the reservoir sampling for each coordinate, and detailed in Algorithm 2.

As for the ELLIPSOIDSAMPLE procedure, it is a modification of the sampling procedure of Abernethy et al. (2008). The point $\mathbf{y}_t$ chosen by the algorithm is uniformly at random chosen from the endpoints of the principal axes of the Dikin ellipsoid $W_1(\mathbf{x}_t)$ centered at $\mathbf{x}_t$. The analysis of Abernethy et al. (2008) already does the hard work of making certain that the ellipsoidal sampling is unbiased and has low variation with respect to the regularization. However, to take advantage of the low variation in the data, we incorporate the previous information in the form of $\tilde{\mu}$. This modification seems to be applicable more generally, not only to the algorithm of Abernethy et al. (2008). However, plugged into this recent algorithm we obtain the best possible regret bounds and also an efficient algorithm.

Before we proceed with the analysis, let us make a small formal claim that our estimates of μ_t are unbiased for $t \ge nk$:

Algorithm 2 SIMPLEXSAMPLE(i_t)

```
1: Predict $\mathbf{y}_t = \gamma \mathbf{e}_{i_t}$, that is, the $i_t$-th standard basis vector scaled by $\gamma$.
2: Observe the cost $\mathbf{f}_t^\top \mathbf{y}_t = \mathbf{f}_t(i_t)$.
3: if some bucket for $i_t$ is empty then
4:     Set $j$ to the index of the empty bucket.
5: else
6:     Set $j$ uniformly at random from $\{1,\ldots,k\}$.
7: end if
8: Update the sample $S_{i_t,j} = \frac{1}{\gamma}\mathbf{f}_t(i_t)$.
9: if $t < nk$ then
10:    Return $\tilde{\mu}_t = 0$.
11: else
12:    Return $\tilde{\mu}_t$ defined as: $\forall i \in \{1,2,\ldots,n\}$, set $\tilde{\mu}_t(i) := \frac{1}{k}\sum_{j=1}^{k} S_{i,j}$.
13: end if
```

Algorithm 3 ELLIPSOIDSAMPLE($\mathbf{x}_t, \tilde{\mu}_t$)

```
1: Let $\{\mathbf{v}_1,\ldots,\mathbf{v}_n\}$ and $\{\lambda_1,\ldots,\lambda_n\}$ be the set of orthogonal eigenvectors and eigenvalues of $\nabla^2\mathcal{R}(\mathbf{x}_t)$.
2: Choose $i_t$ uniformly at random from $\{1,\ldots,n\}$ and $\varepsilon_t = \pm 1$ with probability $1/2$.
3: Predict $\mathbf{y}_t = \mathbf{x}_t + \varepsilon_t \lambda_{i_t}^{-1/2}\mathbf{v}_{i_t}$.
4: Observe the cost $\mathbf{f}_t^\top \mathbf{y}_t$.
5: Return $\tilde{\mathbf{f}}_t$ defined as:
       $$\tilde{\mathbf{f}}_t = \tilde{\mu}_t + \tilde{\mathbf{g}}_t$$
   Where $\tilde{\mathbf{g}}_t := n\left(\mathbf{f}_t^\top \mathbf{y}_t - \tilde{\mu}_t^\top \mathbf{y}_t\right)\varepsilon_t \lambda_{i_t}^{1/2}\mathbf{v}_{i_t}$.
```

Claim 1 *For all $t \geq nk$, and for all $i = 1,2,\ldots,n$, the reservoir for i, $S_i = \{S_{i,1}, S_{i,2},\ldots,S_{i,k}\}$ is a random subset of size k chosen without replacement from $\{\mathbf{f}_1(i), \mathbf{f}_2(i),\ldots,\mathbf{f}_t(i)\}$. Hence, we have $\mathbf{E}[\tilde{\mu}_t] = \mu_t$.*

Proof For $t = nk$ the claim follows because the choice of the random permutation π ensures that the set of times $\{t : (\pi(t) \mod n) + 1 = i\}$ is a random subset of size k chosen without replacement from $\{1,2,\ldots,nk\}$.

For $t > nk$ the claim follows from the properties of reservoir sampling, as we show now. This is because SIMPLEXSAMPLE simulates reservoir sampling. We just showed that at time $t = nk$, the claim is true. Then, at time $t = nk+1$ and onwards, reservoir sampling performs select-and-replace with probability $\frac{k}{t}$ (i.e., it selects $\mathbf{f}_t(i)$ with probability $\frac{k}{t}$ and replaces a random element of the previous S_i with it). The algorithm does exactly the same thing: SIMPLEXSAMPLE is invoked with probability $\frac{nk}{t}$, and with probability $\frac{1}{n}$, we have $i_t = i$. Thus, the overall probability of sample-and-replace is $\frac{k}{t}$, exactly as in reservoir sampling. ■

4. Analysis

In this section, we prove a regret bound, in a slightly easier setting where we know an upper bound Q on the total variation Q_T. The main theorem proved here is the following:

Theorem 6 *Let Q be an estimated upper bound on Q_T. Suppose that Algorithm 1 is run with $\eta = \min\left\{\sqrt{\frac{\log T}{n^2 Q}}, \frac{1}{25n}\right\}$ and $k = \log(T)$. Then, if $Q_T \leq Q$, the expected regret is bounded as follows:*

$$\mathbf{E}[Regret_T] \;=\; O\left(n\sqrt{\vartheta Q \log T} + n\log^2(T) + n\vartheta\log(T)\right).$$

Although this bound requires an estimate of the total variation, we show in the Section 5 how to remove this dependence, thereby proving Theorem 4. In this section we sketch the simpler proof of Theorem 6 and give precise proofs of the main lemmas involved.

Proof For clarity, we present the proof as a series of lemmas whose complete proofs appear after this current proof.

We first relate the expected regret of Algorithm 1 which plays the points $\mathbf{y}_t$, for $t = 1,2,\ldots$ with the $\mathbf{f}_t$ cost vectors to the expected regret of another algorithm that plays the points $\mathbf{x}_t$ with the $\tilde{\mathbf{f}}_t$ cost vectors.

Lemma 7 *For any $\mathbf{u} \in \mathcal{K}$,*

$$\mathbf{E}\left[\sum_{t=1}^{T} \mathbf{f}_t^\top(\mathbf{y}_t - \mathbf{u})\right] \;\leq\; \mathbf{E}\left[\sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{u})\right] + 2n\log^2(T).$$

Intuitively, this bound holds since in every ELLIPSOIDSAMPLE step, the expectation of $\tilde{\mathbf{f}}_t$ and $\mathbf{y}_t$ (conditioned on all previous randomization) are $\mathbf{f}_t$ and $\mathbf{x}_t$ respectively, the expected costs for both algorithms is the same in such rounds. In the SIMPLEXSAMPLE steps, we have $\tilde{\mathbf{f}}_t = 0$ and we can bound $|\mathbf{f}_t^\top(\mathbf{y}_t - \mathbf{u})|$ by 2. The expected number of such steps is $O(nk\log(T)) = O(n\log^2(T))$, which yields the extra additive term.

We therefore turn to bounding $\sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{u})$. For this, we apply standard techniques (originally due to Kalai and Vempala 2005) which bounds the regret of any follow-the-leader type algorithm by terms which depend on the stability of the algorithm, measured by how close the successive predictions $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ are:

Lemma 8 *For any sequence of cost vectors $\tilde{\mathbf{f}}_1, \ldots, \tilde{\mathbf{f}}_T \in \mathbb{R}^n$, the FTRL algorithm with a ϑ-self concordant barrier $\mathcal{R}$ has the following regret guarantee: for any $\mathbf{u} \in \mathcal{K}$, we have*

$$\sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{u}) \;\leq\; \sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{2}{\eta}\vartheta\log T.$$

We now turn to bounding the term $\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1})$. The following main lemma gives such bounds, and forms the main part of the theorem. We go into detail of its proof in the next section, as it contains the main new ideas.

Lemma 9 *Let t be an* ELLIPSOIDSAMPLE *step. Then we have*

$$\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) \;\leq\; 64\eta n^2\|\mathbf{f}_t - \mu_t\|^2 + 64\eta n^2\|\mu_t - \tilde{\mu}_t\|^2 + 2\mu_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}). \quad (4)$$

A similar but much easier statement can be made for SIMPLEXSAMPLE steps. Trivially, since we set $\tilde{\mathbf{f}}_t = 0$ in such steps, we have $\mathbf{x}_t = \mathbf{x}_{t+1}$. Thus, we have

$$\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) \;=\; 0 \;=\; 2\mu_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}).$$

By adding the non-negative term $64\eta n^2\|\mathbf{f}_t - \mu_t\|^2$, we get that for any SIMPLEXSAMPLE step t, we have

$$\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) \;\le\; 64\eta n^2\|\mathbf{f}_t - \mu_t\|^2 + 2\mu_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}). \tag{5}$$

Let T_E be the set of all ELLIPSOIDSAMPLE steps t. Summing up either (4) or (5), as the case may be, over all time periods t we get

$$\sum_{t=1}^{T}\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) \;\le\; 64\eta n^2\sum_{t=1}^{T}\|\mathbf{f}_t - \mu_t\|^2 + 64\eta n^2\sum_{t\in T_E}\|\mu_t - \tilde{\mu}_t\|^2 + 2\sum_{t=1}^{T}\mu_t(\mathbf{x}_t - \mathbf{x}_{t+1}) \tag{6}$$

We bound each term of the inequality (6) above separately. The first term can be easily bounded by the total variation, even though it is the sum of squared deviations from changing means. Essentially, the means don't change very much as time goes on.

Lemma 10 $\sum_{t=1}^{T}\|\mathbf{f}_t - \mu_t\|^2 \;\le\; Q_T$.

The second term, in expectation, is just the variance of the estimators $\tilde{\mu}_t$ of μ_t, which can be bounded in terms of the size of the reservoir and the total variation (see Lemma 2).

Lemma 11 $\mathbf{E}\left[\sum_{t\in T_E}\|\mu_t - \tilde{\mu}_t\|^2\right] \le \frac{\log T}{k}Q_T$.

The third term can be bounded by the sum of successive differences of the means, which, in turn, can be bounded the logarithm of the total variation.

Lemma 12 $\sum_{t=1}^{T}\mu_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) \;\le\; 2\log(Q_T+1)+4$.

Let $Q \ge Q_T$ be a given upper bound. Plugging the bounds from Lemmas 10, 11, and 12 into (6), and using the value $k = \log(T)$, we obtain

$$\sum_{t=1}^{T}\mathbf{f}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) \;\le\; 128\eta n^2 Q + 4\log(Q_T+1) + 8.$$

where we will choose $\eta \ge \frac{\log(Q_T+1)}{8n^2Q}$ so that $\log(Q_T+1) \le 8\eta n^2 Q$. Hence, via Lemmas 8 and 7, we have for any $\mathbf{u} \in \mathcal{K}$,

$$\mathbf{E}\left[\sum_{t=1}^{T}\mathbf{f}_t^\top(\mathbf{y}_t - \mathbf{u})\right] \;\le\; 128\eta n^2 Q + \frac{2\vartheta}{\eta}\log T + 2n\log^2(T) + 4\log(Q_T+1) + 8.$$

Now, choosing $\eta = \min\left\{\sqrt{\frac{\vartheta\log(T)}{n^2Q}}, \frac{1}{25n}\right\}$, for the upper bound $Q \ge Q_T$, and we get the following regret bound:

$$\mathbf{E}\left[\sum_{t=1}^{T}\mathbf{f}_t^\top(\mathbf{y}_t - \mathbf{u})\right] \;\le\; O\left(n\sqrt{\vartheta Q\log T} + n\vartheta\log(T) + n\log^2(T)\right).$$

Here, we absorb the lower order terms $4\log(Q_T+1)+8$ in the other terms using the $O(\cdot)$ notation. The restriction that $\eta \le \frac{1}{25n}$ arises from the proof of Lemma 13 below. ■

4.1 Proof of Main Lemmas

Proof [Lemma 7]
Let t be an ELLIPSOIDSAMPLE step. We first show that $\mathbf{E}[\tilde{\mathbf{f}}_t] = \mathbf{f}_t$. We condition on all the randomness prior to this step, thus, $\tilde{\mu}_t$ is fixed. In the following, $\mathbf{E}_t$ denotes this conditional expectation. Now, condition on the choice i_t and average over the choice of ε_t:

$$\mathbf{E}_t[\tilde{\mathbf{g}}_t|i_t] \;=\; \sum_{\varepsilon_t\in\{1,-1\}} \frac{1}{2}n\left((\mathbf{f}_t-\tilde{\mu}_t)^\top(\mathbf{x}_t+\varepsilon_t\lambda_{i_t}^{-1/2}\mathbf{v}_{i_t})\right)\lambda_{i_t}^{1/2}\varepsilon_t\mathbf{v}_{i_t} = n((\mathbf{f}_t-\tilde{\mu}_t)^\top\mathbf{v}_{i_t})\mathbf{v}_{i_t}.$$

Hence,

$$\mathbf{E}_t[\tilde{\mathbf{g}}_t] = \sum_{i=1}^{n}\frac{1}{n}\cdot n((\mathbf{f}_t-\tilde{\mu}_t)^\top\mathbf{v}_i)\mathbf{v}_i = \mathbf{f}_t-\tilde{\mu}_t,$$

since the $\mathbf{v}_i$ form an orthonormal basis. Thus, $\mathbf{E}_t[\tilde{\mathbf{f}}_t] = \mathbf{E}_t[\tilde{\mathbf{g}}_t]+\tilde{\mu}_t = \mathbf{f}_t$.

Furthermore, it is easy to see that $\mathbf{E}_t[\mathbf{y}_t] = \mathbf{x}_t$, since $\mathbf{y}_t$ is drawn from a symmetric distribution centered at $\mathbf{x}_t$ (namely, the uniform distribution on the endpoints of the principal axes of the Dikin ellipsoid centered at $\mathbf{x}_t$). Thus, we conclude that

$$\mathbf{E}_t[\mathbf{f}_t^\top(\mathbf{y}_t-\mathbf{u})] \;=\; \mathbf{f}_t^\top(\mathbf{x}_t-\mathbf{u}) \;=\; \mathbf{E}_t[\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t-\mathbf{u})],$$

and hence, taking expectation over all the randomness, we have

$$\mathbf{E}[\mathbf{f}_t^\top(\mathbf{y}_t-\mathbf{u})] \;=\; \mathbf{E}[\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t-\mathbf{u})].$$

Now, let t be a SIMPLEXSAMPLE step or alternatively $t\le nk$. In this case, we have $|\mathbf{f}_t^\top(\mathbf{y}_t-\mathbf{u})\| \le \|\mathbf{f}_t\|\|\mathbf{y}_t-\mathbf{u}\| \le 2$, and $\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t-\mathbf{u}) = 0$ since $\tilde{\mathbf{f}}_t = 0$. Thus,

$$\mathbf{E}[\mathbf{f}_t^\top(\mathbf{y}_t-\mathbf{u})] \;\le\; \mathbf{E}[\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t-\mathbf{u})]+2.$$

Overall, if X is the number of SIMPLEXSAMPLE sampling steps or initialization steps, we have

$$\mathbf{E}[\mathbf{f}_t^\top(\mathbf{y}_t-\mathbf{u})] \;\le\; \mathbf{E}_t[\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t-\mathbf{u})]+2\mathbf{E}[X].$$

Finally, using the fact that $\mathbf{E}[X] = nk+\sum_{t=nk+1}^{T}\frac{nk}{t} \le nk(\log(T)+1) \le 2n\log^2(T)$, the proof is complete. ■

Proof [Lemma 8]
By Lemma 15 (see Section 4.2) applied to the sequence $\{\mathbf{x}_t\}$ as defined in (19), for any $\mathbf{u}\in\mathcal{K}$

$$\sum_{t=1}^{T}\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t-\mathbf{u}) \;\le\; \sum_{t=1}^{T}\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t-\mathbf{x}_{t+1})+\frac{1}{\eta}[\mathcal{R}(\mathbf{u})-\mathcal{R}(\mathbf{x}_1)].$$

By Lemma 3, there exists a vector $\mathbf{u}_1 \in \mathcal{K}_\delta \subseteq \mathcal{K}$ for $\delta = \frac{1}{T}$, such that $\|\mathbf{u}_1 - \mathbf{u}\| \leq \frac{2}{T}$ and in addition, $\mathcal{R}(\mathbf{u}_1) - \mathcal{R}(\mathbf{x}_1) \leq \vartheta \log(T)$. Hence,

$$\begin{aligned}
\sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{u}) &\leq \sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{u}_1) + \sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top (\mathbf{u}_1 - \mathbf{u}) \\
&\leq \sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{1}{\eta}[\mathcal{R}(\mathbf{u}_1) - \mathcal{R}(\mathbf{x}_1)] + \sum_{t=1}^{T} \|\mathbf{f}_t\| \|\mathbf{u}_1 - \mathbf{u}\| \\
&\leq \sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{\vartheta}{\eta} \log T + \sum_{t=1}^{T} \frac{2}{T} \\
&\leq \sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{2\vartheta}{\eta} \log T.
\end{aligned}$$

In the last step, we upper bound $\sum_{t=1}^{T} \frac{2}{T} \leq \frac{\vartheta}{\eta} \log T$, which is valid for $\eta < 1/4$, say. ■

Now we turn to proving Lemma 9. We first develop some machinery to assist us. Lemmas 13 and 14 are essentially generalizations of similar lemmas from Abernethy et al. (2008) to the case in which we have both sampling and ellipsoidal steps.

Lemma 13 *For any time period $t \geq nk$, the next minimizer $\mathbf{x}_{t+1}$ is "close" to $\mathbf{x}_t$:*

$$\mathbf{x}_{t+1} \in W_{\frac{1}{2}}(\mathbf{x}_t).$$

Proof If t is a SIMPLEXSAMPLE step, then $\mathbf{x}_t = \mathbf{x}_{t+1}$ and the lemma is trivial. So assume that t is an ELLIPSOIDSAMPLE step. Now, recall that

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x} \in \mathcal{K}} \Phi_t(\mathbf{x}) \quad \text{and} \quad \mathbf{x}_t = \arg\min_{\mathbf{x} \in \mathcal{K}} \Phi_{t-1}(\mathbf{x}),$$

where $\Phi_t(\mathbf{x}) = \eta \sum_{s=1}^{t} \tilde{\mathbf{f}}_t^\top \mathbf{x} + \mathcal{R}(\mathbf{x})$. Since the barrier function $\mathcal{R}$ goes to infinity as we get close to the boundary, the points $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ are both in the interior of $\mathcal{K}$. We now show that all points on the boundary of $W_{\frac{1}{2}}(\mathbf{x}_t)$ have higher Φ_t value than $\Phi_t(\mathbf{x}_t)$, and since $\mathbf{x}_{t+1}$ is the minimizer of the strictly convex function Φ_t, we conclude that $\mathbf{x}_{t+1}$ must lie in the interior of $W_{\frac{1}{2}}(\mathbf{x}_t)$.

First, note that since $\mathbf{x}_t$ is in the interior of $\mathcal{K}$, the first order optimality condition gives $\nabla\Phi_{t-1}(\mathbf{x}_t) = 0$, and we conclude that $\nabla\Phi_t(\mathbf{x}_t) = \eta\tilde{\mathbf{f}}_t$. Now consider any point in $\mathbf{z}$ on the boundary of $W_{\frac{1}{2}}(\mathbf{x}_t)$, that is, $\mathbf{y} = \mathbf{x}_t + \mathbf{h}$ for some vector $\mathbf{h}$ such that $\|\mathbf{h}\|_{\mathbf{x}_t} = \frac{1}{2}$. Using the multi-variate Taylor expansion, we get

$$\Phi_t(\mathbf{y}) = \Phi_t(\mathbf{x}_t + \mathbf{h}) = \Phi_t(\mathbf{x}_t) + \nabla\Phi_t(\mathbf{x}_t)^\top \mathbf{h} + \frac{1}{2}\mathbf{h}^\top \nabla^2\Phi_t(\xi)\mathbf{h} = \Phi_t(\mathbf{x}_t) + \eta\tilde{\mathbf{f}}_t^\top \mathbf{h} + \frac{1}{2}\mathbf{h}^\top \nabla^2\Phi_t(\xi)\mathbf{h} \tag{7}$$

for some ξ on the line segment between $\mathbf{x}_t$ and $\mathbf{x}_t + \mathbf{h}$. This latter fact also implies that $\|\xi - \mathbf{x}_t\|_{\mathbf{x}_t} \leq \|\mathbf{h}\|_{\mathbf{x}_t} \leq \frac{1}{2}$. Hence, by (3),

$$\nabla^2\mathcal{R}(\xi) \succeq (1 - \|\xi - \mathbf{x}_t\|_{\mathbf{x}_t})^2 \nabla^2\mathcal{R}(\mathbf{x}_t) \succeq \frac{1}{4}\nabla^2\mathcal{R}(\mathbf{x}_t).$$

Thus $\mathbf{h}^\top \nabla^2 \mathcal{R}(\xi)\mathbf{h} \;\geq\; \frac{1}{4}\|\mathbf{h}\|_{\mathbf{x}_t} = \frac{1}{8}$. Next, we bound $|\tilde{\mathbf{f}}_t^\top \mathbf{h}|$ as follows:

$$|\tilde{\mathbf{f}}_t^\top \mathbf{h}| \;\leq\; \|\tilde{\mathbf{f}}_t\|^\star_{\mathbf{x}_t}\|\mathbf{h}\|_{\mathbf{x}_t} \;\leq\; \frac{1}{2}\|\tilde{\mathbf{f}}_t\|^\star_{\mathbf{x}_t}.$$

Claim 2 $\|\tilde{\mathbf{f}}_t\|^\star_{\mathbf{x}_t} \leq 3n$.

Proof We have $\tilde{\mathbf{f}}_t = \tilde{\mu}_t + \tilde{\mathbf{g}}_t$, where $\tilde{\mathbf{g}}_t = n\left((\mathbf{f}_t - \tilde{\mu}_t)^\top \mathbf{y}_t\right)\varepsilon_t \lambda_{i_t}^{1/2}\mathbf{v}_{i_t}$. We have

$$\begin{aligned}\|\tilde{\mathbf{g}}_t\|^{\star 2}_{\mathbf{x}_t} &= \left[n\left((\mathbf{f}_t - \tilde{\mu}_t)^\top \mathbf{y}_t\right)\varepsilon_t \lambda_{i_t}^{1/2}\mathbf{v}_{i_t}\right]^\top [\nabla^2\mathcal{R}(\mathbf{x}_t)]^{-1}\left[n\left((\mathbf{f}_t - \tilde{\mu}_t)^\top \mathbf{y}_t\right)\varepsilon_t \lambda_{i_t}^{1/2}\mathbf{v}_{i_t}\right] \\ &= n^2\left((\mathbf{f}_t - \tilde{\mu}_t)^\top \mathbf{y}_t\right)^2,\end{aligned}$$

since $\mathbf{v}_{i_t}^\top [\nabla^2\mathcal{R}(\mathbf{x}_t)]^{-1}\mathbf{v}_{i_t} = 1/\lambda_{i_t}$. Hence,

$$\|\tilde{\mathbf{f}}_t\|^\star_{\mathbf{x}_t} \;\leq\; \|\tilde{\mu}_t\|^\star_{\mathbf{x}_t} + \|\tilde{\mathbf{g}}_t\|^\star_{\mathbf{x}_t} \;\leq\; \|\tilde{\mu}_t\| + n|(\mathbf{f}_t - \tilde{\mu}_t)^\top \mathbf{y}_t| \;\leq\; 3n,$$

since $\|\tilde{\mu}_t\|^\star_{\mathbf{x}_t} \leq \|\tilde{\mu}_t\| \leq 1$. We also used the facts that $\|\mathbf{y}_t\| \leq 1$ and $\|\mathbf{f}_t - \tilde{\mu}_t\| \leq 2$. ■

Hence, from (7) we get

$$\Phi_t(\mathbf{y}) \;\geq\; \Phi_t(\mathbf{x}_t) - \eta \cdot \frac{3n}{2} + \frac{1}{16} \;>\; \Phi_t(\mathbf{x}_t),$$

since $\eta \leq \frac{1}{25n}$. This concludes the proof that all boundary points of $W_{\frac{1}{2}}(\mathbf{x}_t)$ have higher Φ_t value than $\Phi_t(\mathbf{x}_t)$. ■

Lemma 14 *For any time period $t \geq nk$, we have*

$$\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2_{\mathbf{x}_t} \;\leq\; 4\eta\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}).$$

Proof Applying the Taylor series expansion to the function Φ_t around the point $\mathbf{x}_t$, we get that for some point $\mathbf{z}_t$ on the line segment joining $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$, we have

$$\Phi_t(\mathbf{x}_t) \;=\; \Phi_t(\mathbf{x}_{t+1}) + \nabla\Phi_t(\mathbf{x}_{t+1})^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) + (\mathbf{x}_{t+1} - \mathbf{x}_t)^\top \nabla^2\Phi_t(\mathbf{z}_t)(\mathbf{x}_{t+1} - \mathbf{x}_t) \;=\; \Phi_t(\mathbf{x}_{t+1}) + \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2_{\mathbf{z}_t},$$

because $\nabla\Phi_t(\mathbf{x}_{t+1}) = 0$ since $\mathbf{x}_{t+1}$, the minimizer of Φ_t, is in the interior of $\mathcal{K}$. We also used the fact that $\nabla^2\Phi_t(\mathbf{z}_t) = \nabla^2\mathcal{R}(\mathbf{z}_t)$. Thus, we have

$$\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2_{\mathbf{z}_t} \;=\; \Phi_t(\mathbf{x}_t) - \Phi_t(\mathbf{x}_{t+1}) \;=\; \Phi_{t-1}(\mathbf{x}_t) - \Phi_{t-1}(\mathbf{x}_{t+1}) + \eta\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) \;\leq\; \eta\tilde{\mathbf{f}}_t^\top(\mathbf{x}_t - \mathbf{x}_{t+1}),$$

since $\mathbf{x}_t$ is the minimizer of Φ_{t-1} in $\mathcal{K}$. It remains to show that $\frac{1}{4}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2_{\mathbf{x}_t} \leq \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2_{\mathbf{z}_t}$, for which it suffices to show $\frac{1}{4}\nabla^2\mathcal{R}(\mathbf{x}_t) \preceq \nabla^2\mathcal{R}(\mathbf{z}_t)$.

By Lemma 13 we have $\mathbf{x}_{t+1} \in W_{1/2}(\mathbf{x}_t)$, and hence $\mathbf{z}_t \in W_{1/2}(\mathbf{x}_t)$ (since $\mathbf{z}_t$ is on the line segment between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$). Therefore, using (3) we have $\frac{1}{4}\nabla^2\mathcal{R}(\mathbf{x}_t) \preceq \nabla^2\mathcal{R}(\mathbf{z}_t)$ as required. ■

Proof [Lemma 9]
First, we have

$$\begin{aligned}
(\tilde{\mathbf{f}}_t - \mu_t)^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) &\leq \|\tilde{\mathbf{f}}_t - \mu_t\|^\star_{\mathbf{x}_t} \cdot \|\mathbf{x}_t - \mathbf{x}_{t+1}\|_{\mathbf{x}_t} && \text{(by (1))} \\
&\leq \|\tilde{\mathbf{f}}_t - \mu_t\|^\star_{\mathbf{x}_t} \cdot \sqrt{4\eta \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1})} && \text{(Lemma 14)} \\
&\leq 2\eta \|\tilde{\mathbf{f}}_t - \mu_t\|^{\star 2}_{\mathbf{x}_t} + \frac{1}{2}\tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}).
\end{aligned}$$

The last inequality follows using the fact that $ab \leq \frac{1}{2}(a^2 + b^2)$ for real numbers a, b. Simplifying, we get that

$$\begin{aligned}
\tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) &\leq 4\eta \|\tilde{\mathbf{f}}_t - \mu_t\|^{\star 2}_{\mathbf{x}_t} + 2\mu_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \\
&\leq 8\eta \left(\|\tilde{\mathbf{f}}_t - \tilde{\mu}_t\|^{\star 2}_{\mathbf{x}_t} + \|\mu_t - \tilde{\mu}_t\|^{\star 2}_{\mathbf{x}_t} \right) + 2\mu_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \\
&\leq 32\eta \left(\|\tilde{\mathbf{g}}_t\|^{\star 2}_{\mathbf{x}_t} + \|\mu_t - \tilde{\mu}_t\|^2 \right) + 2\mu_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}).
\end{aligned}$$

The last inequality is because $\|\cdot\|^\star_{\mathbf{x}} \leq 2\|\cdot\|$ from (2) and the assumption that $\mathcal{K}$ is contained inside the unit ball.

Using the definition of $\tilde{\mathbf{g}}_t$ from Algorithm 3, we get that

$$\begin{aligned}
\|\tilde{\mathbf{g}}_t\|^{\star 2}_{\mathbf{x}_t} &= n^2 \left((\mathbf{f}_t - \tilde{\mu}_t)^\top \mathbf{y}_t \right)^2 \lambda_{i_t} \cdot \left(\mathbf{v}_{i_t}^\top [\nabla^2 \mathcal{R}(\mathbf{x}_t)]^{-1} \mathbf{v}_{i_t} \right) \\
&= n^2 \left((\mathbf{f}_t - \tilde{\mu}_t)^\top \mathbf{y}_t \right)^2 \\
&\leq n^2 \|\mathbf{f}_t - \tilde{\mu}_t\|^2 \\
&\leq 2n^2 [\|\mathbf{f}_t - \mu_t\|^2 + \|\mu_t - \tilde{\mu}_t\|^2].
\end{aligned}$$

The first inequality follows by applying Cauchy-Schwarz and using the fact that $\|\mathbf{y}_t\| \leq 1$. Plugging this bound into the previous bound we conclude that

$$\tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \leq 64\eta n^2 \|\mathbf{f}_t - \mu_t\|^2 + 64\eta n^2 \|\mu_t - \tilde{\mu}_t\|^2 + 2\mu_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}).$$

■

Proof [Lemma 10]

Recall that $\mu_t = \arg\min_\mu \sum_{\tau=1}^t \|\mathbf{f}_\tau - \mu\|^2$. As a first step, we show that

$$\sum_{\tau=1}^T \|\mathbf{f}_t - \mu_t\|^2 \leq \sum_{\tau=1}^T \|\mathbf{f}_t - \mu_T\|^2.$$

This is proved by induction on t. For $T = 1$ the inequality is trivial; we actually have equality. Assume correctness for some $T-1$. Moving to T, we have

$$\begin{aligned}
\sum_{t=1}^{T} \|\mathbf{f}_t - \mu_t\|^2 &= \sum_{t=1}^{T-1} \|\mathbf{f}_t - \mu_t\|^2 + \|\mathbf{f}_T - \mu_T\|^2 \\
&\leq \sum_{t=1}^{T-1} \|\mathbf{f}_t - \mu_{T-1}\|^2 + \|\mathbf{f}_T - \mu_T\|^2 && \text{(By inductive hypothesis)} \\
&\leq \sum_{t=1}^{T-1} \|\mathbf{f}_t - \mu_T\|^2 + \|\mathbf{f}_T - \mu_T\|^2 && (\mu_{T-1} = \arg\min_{\mathbf{x}} \textstyle\sum_{t=1}^{T-1} \|\mathbf{f}_t - \mathbf{x}\|^2) \\
&= \sum_{t=1}^{T} \|\mathbf{f}_t - \mu_T\|^2.
\end{aligned}$$

Hence,

$$\sum_{\tau=1}^{T} \|\mathbf{f}_t - \mu_t\|^2 \leq \sum_{\tau=1}^{T} \|\mathbf{f}_t - \mu\|^2 = Q_T.$$

■

Proof [Lemma 11]
Any ELLIPSOIDSAMPLE step t must have $t \geq nk$, so by Claim 1 the algorithm exactly implements reservoir sampling with a reservoir of size k for each of the n coordinates.

Now, for any coordinate i, $\tilde{\mu}_t(i)$ is the average of a k samples chosen *without* replacement from F_t. Thus, we have $\mathbf{E}[\tilde{\mu}_t(i)] = \mu_t(i)$, and hence $\mathbf{E}[(\tilde{\mu}_t(i) - \mu_t(i))^2] = \mathrm{VAR}[\tilde{\mu}_t(i)]$.

Now consider another estimator $\nu_t(i)$, which averages k samples chosen *with* replacement from F_t. It is a well-known statistical fact (see, e.g., Rice, 2001) that $\mathrm{VAR}[\tilde{\mu}_t(i)] \leq \mathrm{VAR}[\nu_t(i)]$. Thus, we bound $\mathrm{VAR}[\nu_t(i)]$ instead.

Suppose $t > nk$. Let $\mu = \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}_t$. Since $\mathbf{E}[\nu_t(i)] = \mu_t(i)$, we have

$$\begin{aligned}
\mathrm{VAR}[\nu_t(i)] = \mathbf{E}[(\nu_t(i) - \mu_t(i))^2] &\leq \mathbf{E}[(\nu_t(i) - \mu(i))^2] \\
&= \frac{1}{k}\sum_{\tau=1}^{t} \frac{1}{t}(\mathbf{f}_\tau(i) - \mu(i))^2
\end{aligned}$$

Summing up over all coordinates i, we get

$$\mathbf{E}[\|\tilde{\mu}_t - \mu_t\|^2] \leq \sum_i \mathrm{VAR}[\nu_t(i)] \leq \frac{1}{kt}Q_t \leq \frac{1}{kt}Q_T.$$

Summing up over all ELLIPSOIDSAMPLE steps t, we get

$$\mathbf{E}\left[\sum_{t \in T_E} \|\tilde{\mu}_t - \mu_t\|^2\right] \leq \sum_{t \in T_E} \frac{1}{kt}Q_T \leq \frac{\log(T)}{k}Q_T.$$

■

Proof [Lemma 12]
We have

$$\sum_{t=1}^{T} \mu_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) = \sum_{t=1}^{T} \mathbf{x}_{t+1}(\mu_{t+1} - \mu_t) + \mu_1 \mathbf{x}_1 - \mathbf{x}_{T+1}\mu_{T+1}.$$

Thus, since $\|\mathbf{x}_t\| \leq 1$ and $\|\mu_t\| \leq 1$, we have

$$\sum_{t=1}^{T} \mu_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \ \leq \ \sum_{t=2}^{T} \|\mu_{t+1} - \mu_t\| + 4.$$

To proceed, we appeal to Lemma 16 (see Section 4.2), and apply it for $x_t := \|\mathbf{f}_t - \mu_t\|$. Let $\mu = \frac{1}{T}\sum_{t=1}^{T} \mathbf{f}_t$. Arguing as in Lemma 10, we have

$$\sum_t x_t^2 \ = \ \sum_{t=1}^{T} \|\mathbf{f}_t - \mu_t\|^2 \ \leq \ \sum_{t=1}^{T} \|\mathbf{f}_t - \mu\|^2 \ \leq \ Q_T.$$

Notice that

$$\mu_t - \mu_{t-1} = \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{f}_\tau - \frac{1}{t-1}\sum_{\tau=1}^{t-1} \mathbf{f}_\tau = \frac{1}{t}\mathbf{f}_t + (\frac{1}{t} - \frac{1}{t-1})\sum_{\tau=1}^{t-1} \mathbf{f}_\tau = \frac{1}{t}(\mathbf{f}_t - \mu_{t-1}).$$

Hence,

$$\|\mu_t - \mu_{t-1}\| \ = \ \frac{1}{t}\|\mathbf{f}_t - \mu_{t-1}\| \ \leq \ \frac{1}{t}x_t + \frac{1}{t}\|\mu_t - \mu_{t-1}\|,$$

from which we conclude that for all $t \geq 2$ we have $\|\mu_t - \mu_{t-1}\| \leq \frac{t^{-1}}{1-t^{-1}}x_t \leq \sum_t \frac{2}{t}x_t$. Now, we apply Lemma 16 to conclude that

$$\sum_{t=2}^{T} \|\mu_{t+1} - \mu_t\| + 4 \quad \leq \ 2\log(Q_T + 1) + 4.$$

■

4.2 Auxiliary Lemmas

In this section, we give a number of auxiliary lemmas that are independent of the analysis of the algorithm. These lemmas give useful bounds that are used in the main analysis.

The first lemma gives a general regret bound for any follow-the-regularized-leader style algorithm. The proof of this bound is essentially due to Kalai and Vempala (2005).

Lemma 15 *Consider an online linear optimization instance over a convex set $\mathcal{K}$, with a regularization function $\mathcal{R}$ and a sequence $\{x_t\}$ defined by*

$$\mathbf{x}_t = \arg\min_{\mathbf{x}\in\mathcal{K}} \left\{ \sum_{\tau=1}^{t-1} \mathbf{f}_\tau^\top \mathbf{x} + \mathcal{R}(\mathbf{x}) \right\}.$$

For every $\mathbf{u} \in \mathcal{K}$, the sequence $\{\mathbf{x}_t\}$ satisfies the following regret guarantee

$$\sum_{t=1}^{T} \mathbf{f}_t^\mathsf{T}(\mathbf{x}_t - \mathbf{u}) \quad \leq \quad \sum_{t=1}^{T} \mathbf{f}_t^\mathsf{T}(\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{1}{\eta}[\mathcal{R}(\mathbf{u}) - \mathcal{R}(\mathbf{x}_1)].$$

Proof For convenience, denote by $\mathbf{f}_0 = \frac{1}{\eta}\mathcal{R}$, and assume we start the algorithm from $t = 0$ with an arbitrary $\mathbf{x}_0$. The lemma is now proved by induction on T.

In the base case, for $T = 1$, by definition we have that $\mathbf{x}_1 = \arg\min_{\mathbf{x}}\{\mathcal{R}(\mathbf{x})\}$, and thus $\mathbf{f}_0(\mathbf{x}_1) \leq \mathbf{f}_0(\mathbf{u})$ for all $\mathbf{u}$, thus $\mathbf{f}_0(\mathbf{x}_0) - \mathbf{f}_0(\mathbf{u}) \leq \mathbf{f}_0(\mathbf{x}_0) - \mathbf{f}_0(\mathbf{x}_1)$.

Now assume that that for some $T \geq 1$, we have

$$\sum_{t=0}^{T} \mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{u}) \quad \leq \quad \sum_{t=0}^{T} \mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{x}_{t+1}).$$

We now prove the claimed inequality for $T+1$. Since $\mathbf{x}_{T+2} = \arg\min_{\mathbf{x}}\{\sum_{t=0}^{T+1} \mathbf{f}_t(\mathbf{x})\}$ we have:

$$\begin{aligned}
\sum_{t=0}^{T+1} \mathbf{f}_t(\mathbf{x}_t) - \sum_{t=0}^{T+1} \mathbf{f}_t(\mathbf{u}) &\leq \sum_{t=0}^{T+1} \mathbf{f}_t(\mathbf{x}_t) - \sum_{t=0}^{T+1} \mathbf{f}_t(\mathbf{x}_{T+2}) \\
&= \sum_{t=0}^{T} (\mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{x}_{T+2})) + \mathbf{f}_{T+1}(\mathbf{x}_{T+1}) - \mathbf{f}_{T+1}(\mathbf{x}_{T+2}) \\
&\leq \sum_{t=0}^{T} (\mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{x}_{t+1})) + \mathbf{f}_{T+1}(\mathbf{x}_{T+1}) - \mathbf{f}_{T+1}(\mathbf{x}_{T+2}) \\
&= \sum_{t=0}^{T+1} \mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{x}_{t+1}).
\end{aligned}$$

In the third line we used the induction hypothesis for $\mathbf{u} = \mathbf{x}_{T+2}$. We conclude that

$$\begin{aligned}
\sum_{t=1}^{T} \mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{u}) &\leq \sum_{t=1}^{T} \mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{x}_{t+1}) + [-\mathbf{f}_0(\mathbf{x}_0) + \mathbf{f}_0(\mathbf{u}) + \mathbf{f}_0(\mathbf{x}_0) - \mathbf{f}_0(\mathbf{x}_1)] \\
&= \sum_{t=1}^{T} \mathbf{f}_t(\mathbf{x}_t) - \mathbf{f}_t(\mathbf{x}_{t+1}) + \frac{1}{\eta}[\mathcal{R}(\mathbf{u}) - \mathcal{R}(\mathbf{x}_1)].
\end{aligned}$$

■

Lemma 16 *Suppose we have real numbers $x_1, x_2, \ldots, x_T$ such that $0 \leq x_t \leq 1$ and $\sum_t x_t^2 \leq Q$. Then*

$$\sum_{t=1}^{T} \frac{1}{t} x_t \ \leq\ \log(Q+1) + 1.$$

Proof By Lemma 17 below, the values of x_t that maximize $\sum_{t=1}^{T} \frac{1}{t} x_t$ must have the following structure: there is a k such that for all $t \leq k$, we have $x_t = 1$, and for any index $t > k$, we have $x_{k+1}/x_t \geq \frac{1}{k}/\frac{1}{t}$, which implies that $x_t \leq k/t$. We first note that $k \leq Q$, since $Q \geq \sum_{t=1}^{k} x_t^2 = k$. Now, we can bound the value as follows:

$$\sum_{t=1}^{T} \frac{1}{t} x_t \ \leq\ \sum_{t=1}^{k} \frac{1}{t} + \sum_{t=k+1}^{T} \frac{k}{t^2} \ \leq\ \log(k+1) + k \cdot \frac{1}{k} \ \leq\ \log(Q+1) + 1.$$

■

Lemma 17 *Let $a_1 \geq a_2 \geq \dots a_T > 0$. Then the optimal solution of*

$$\max \sum_i a_i x_i \text{ subject to}$$
$$\forall i:\ 0 \leq x_i \leq 1$$
$$\sum_i x_i^2 \leq Q$$

has the following properties: $x_1 \geq x_2 \geq \dots x_T$, and for any pair of indices i, j, with $i < j$, either $x_i = 1$, $x_i = 0$ or $x_i/x_j \geq a_i/a_j$.

Proof The fact that in the optimal solution $x_1 \geq x_2 \geq \dots x_T$ is obvious, since otherwise we could permute the x_i's to be in decreasing order and increase the value.

The second fact follows by the Karush-Kuhn-Tucker (KKT) optimality conditions, which imply the existence of constants $\mu, \lambda_1, \dots, \lambda_T, \rho_1, \dots, \rho_T$ for which the optimal solution satisfies

$$\forall i:\ -a_i + 2\mu x_i + \lambda_i + \rho_i = 0.$$

Furthermore, the complementary slackness condition says that the constants λ_i, ρ_i are equal to zero for all indices of the solution which satisfy $x_i \notin \{0, 1\}$. For such x_i, the KKT equation is

$$-a_i + 2\mu x_i = 0,$$

which implies the lemma. ■

5. Tuning the Learning Rate: Proof of Theorem 4

Theorem 6 requires *a priori* knowledge of a good bound Q on the total quadratic variation Q_T. This may not be possible in many situations. Typically, in online learning scenarios where a regret bound of $O(\sqrt{A_T})$ for some quantity A_T which grows with T is desired, one first gives an online learning algorithm $L(\eta)$ where $\eta \leq 1$ is a learning rate parameter which obtains a regret bound of

$$\text{Regret}_T \ \leq\ \eta A_T + O(1/\eta).$$

Then, we can obtain a master online learning algorithm whose regret grows like $O(\sqrt{A_T})$ as follows. We start with $\eta = 1$, and run the learning algorithm $L(\eta)$. Then, the master algorithm tracks how A_T grows with T. As soon as A_T quadruples, the algorithm resets η to half its current value, and restarts with $L(\eta)$. This simple trick can be shown to obtain $O(\sqrt{A_T})$ regret.

Unfortunately, this trick doesn't work in our case, where $A_T = Q_T$, since we cannot even compute Q_T accurately in the bandit setting. For this reason, obtaining a regret bound of $\tilde{O}(\sqrt{Q_T})$ becomes quite non-trivial. In this section, we give a method to obtain such a regret bound. At its heart, we still make use of the η-halving trick, but in a subtle way. We assume that we know a good bound on $\log(T)$ in advance. This is not a serious restriction, it can be circumvented by standard tricks, but we make this assumption in order to simplify the exposition.

We design our master algorithm in the following way. Let $L(\eta)$ be Algorithm 1 with the given parameter η and $k = \log(T)$. We initialize $\eta_0 = \frac{1}{25n}$. The master algorithm then runs in phases indexed by $i = 0, 1, 2, \ldots$. In phase i, the algorithm runs $L(\eta_i)$ where $\eta_i = \eta_0/2^i$. The decision to end a phase i and start phase $i+1$ is taken in the following manner: let t_i be first period of phase i, and let t be the current period. We start phase $i+1$ as soon as

$$\sum_{\tau=t_i}^{t} \tilde{\mathbf{f}}_\tau^\top (\mathbf{x}_\tau - \mathbf{x}_{\tau+1}) \;\geq\; \frac{2}{\eta_i}\vartheta \log(T).$$

Thus, phase i ends at time period $t-1$, and the point $\mathbf{x}_t$ computed by $L(\eta_i)$ is discarded by the master algorithm since $L(\eta_{i+1})$ starts at this point and $\mathbf{x}_t$ is reset to the initial point of $L(\eta_{i+1})$. Note that this sum can be computed by the algorithm, and hence the algorithm is well-defined. This completes the description of the master algorithm.

5.1 Analysis

Define $I_i = \{t_i, t_i+1, \ldots, t_{i+1}-1\}$, that is, the interval of time periods which constitute phase i.

By Lemma 8, for any $\mathbf{u} \in \mathcal{K}$, we have

$$\sum_{t\in I_i} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{u}) \;\leq\; \sum_{t\in I_i} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) + \frac{2}{\eta_i}\vartheta\log(T) \;\leq\; \frac{4}{\eta_i}\vartheta\log(T).$$

Note that this inequality uses the fact that the sum $\sum_{\tau=t_i}^{t} \tilde{\mathbf{f}}_\tau^\top (\mathbf{x}_\tau - \mathbf{x}_{\tau+1})$ is a monotonically increasing as t increases, since by Lemma 14, we have that $\tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \geq 0$.

Let $i^\star$ be the index of the final phase. Summing up this bound over all phases, we have

$$\sum_{t=1}^{T} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{u}) \;\leq\; \sum_{i=0}^{i^\star} \frac{4}{\eta_i}\vartheta\log(T) \;\leq\; \frac{8}{\eta_{i^\star}}\vartheta\log(T).$$

Then, using Lemma 7 we get that the expected regret of this algorithm is bounded by

$$\mathbf{E}\left[\sum_{t=1}^{T} \mathbf{f}_t^\top (\mathbf{y}_t - \mathbf{u})\right] \;\leq\; \mathbf{E}\left[\frac{1}{\eta_{i^\star}}\right] \cdot (8\vartheta\log(T)) + O(n\log^2(T)). \tag{8}$$

We now need to bound $\mathbf{E}\left[\frac{1}{\eta_{i^\star}}\right]$. If the choice of the randomness in the algorithm is such that $i^\star = 0$, then $\frac{1}{\eta_{i^\star}} \leq 25n$ is an upper bound.

Otherwise, $i^\star > 0$, and so the phase $i^\star - 1$ is well-defined. For brevity, let $J = I_{i^\star-1} \cup \{t_{i^\star}\}$, and let J_E be the ELLIPSOIDSAMPLE steps in J. For this interval, we have (here, $\mathbf{x}_{t_{i^\star}}$ is the point computed by $L(\eta_{i^\star-1})$, which is discarded by the master algorithm when phase $i^\star$ starts):

$$\sum_{t\in J} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \;\geq\; \frac{2}{\eta_{i^\star-1}}\vartheta\log(T) \;=\; \frac{1}{\eta_{i^\star}}\vartheta\log(T).$$

Applying the bound (6), and using the fact that $\eta_{i^\star-1} = 2\eta_{i^\star}$, we get

$$\sum_{t\in J} \tilde{\mathbf{f}}_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}) \;\leq\; 128\eta_{i^\star} n^2 \sum_{t\in J} \|\mathbf{f}_t - \mu_t\|^2 + 128\eta_{i^\star} n^2 \sum_{t\in J_E} \|\mu_t - \tilde{\mu}_t\|^2 + 2\sum_{t\in J} \mu_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}).$$

Putting these together, and dividing by $\eta_{i^\star}$, we get

$$\frac{1}{\eta_{i^\star}^2}\vartheta\log(T) \;\leq\; 128n^2 \sum_{t\in J} \|\mathbf{f}_t - \mu_t\|^2 + 128n^2 \sum_{t\in J_E} \|\mu_t - \tilde{\mu}_t\|^2 + \frac{2}{\eta_{i^\star}} \sum_{t\in J} \mu_t^\top (\mathbf{x}_t - \mathbf{x}_{t+1}). \tag{9}$$

Lemmas 10 and 12 give us the following upper bounds:

$$\sum_{t\in J}\|\mathbf{f}_t-\mu_t\|^2 \;\le\; Q_T \quad\text{ and }\quad \sum_{t\in J}\mu_t^\top(\mathbf{x}_t-\mathbf{x}_{t+1}) \;\le\; 2\log(Q_T+1)+4.$$

Denote the expectation of a random variable conditioned on all the randomness before phase $i^\star-1$ by $\mathbf{E}_{i^\star-1}$. By Lemma 11 we have the bound

$$\mathbf{E}_{i^\star-1}\left[\sum_{t\in J_E}\|\mu_t-\tilde{\mu}_t\|^2\right] \;\le\; \frac{\log(T)}{k}Q_T.$$

Taking the expectation conditioned on all the randomness before phase $i^\star-1$ on both sides of inequality (9) and applying the above bounds, and using $k=\log(T)$, we get

$$\frac{1}{\eta_{i^\star}^2}\vartheta\log(T) \;\le\; 256n^2Q_T+\frac{4\log(Q_T+1)+8}{\eta_{i^\star}}.$$

Hence, one of $256n^2Q_T$ or $\frac{2}{\eta_{i^\star}}\log(Q_T)$ must be at least $\frac{1}{2\eta_{i^\star}^2}\vartheta\log(T)$. In the first case, we get the bound $\frac{1}{\eta_{i^\star}}\le 25n\sqrt{\frac{Q_T}{\vartheta\log(T)}}$. In the second case, we get the bound $\frac{1}{\eta_{i^\star}}\le\frac{8\log(Q_T+1)+16}{\vartheta\log(T)}$.

In all cases (including the case when $i^\star=0$), we have $\frac{1}{\eta_{i^\star}}\le O\left(n\sqrt{\frac{Q_T}{\vartheta\log(T)}}+n\right)$, and hence we can bound

$$\mathbf{E}\left[\frac{1}{\eta_{i^\star}}\right]\cdot(\vartheta\log(T)) \;=O\left(n\sqrt{\vartheta Q_T\log T}+n\vartheta\log(T)\right).$$

Plugging this into (8), and for $k=\log(T)$, we get that the expected regret is bounded by

$$\mathbf{E}\left[\sum_{t=1}^T\mathbf{f}_t^\top(\mathbf{y}_t-\mathbf{u})\right] \;=\; O\left(n\sqrt{\vartheta Q_T\log(T)}+n\log^2(T)+n\vartheta\log(T)\right).$$

6. Conclusions and Open Problems

In this paper, we gave the first bandit online linear optimization algorithm whose regret is bounded by the square-root of the total quadratic variation of the cost vectors. These bounds naturally interpolate between the worst-case and stochastic models of the problem.[4]

This algorithm continues a line of work which aims to prove variation-based regret bounds for any online learning framework. So far, such bounds have been obtained for four major online learning scenarios: expert prediction, online linear optimization, portfolio selection (and exp-concave cost functions), and bandit online linear optimization in this paper.

The concept of variational regret bounds in the setting of the ubiquitous multi-armed bandit problem opens many interesting directions for further research and open questions:

1. Improve upon the bounds presented in this paper by removing the dependence on the number of iterations completely - that is, remove the poly$(\log(T))$ terms in the regret bound.

4. In the stochastic multi-armed bandit setting, the regret is known to be bounded by a logarithm in the number of iterations rather than square root (Auer et al., 2002). However, note that the regret is defined differently in the stochastic case, which makes the logarithmic dependency even possible. In this paper we consider a stronger notion of worst-case regret.

2. For the special case of the classic non-stochastic MAB problem, obtain regret bounds which depend on the variation of the best action in hindsight (vs. the total variation).

3. Is it possible to improve regret for the classic non-stochastic multi-armed bandit problem without using the self-concordance methodology (perhaps by extending the algorithm in Hazan and Kale (2008) to the bandit setting)?

References

J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *The 21st Annual Conference on Learning Theory (COLT).*, 2008.

J. Audibert and S. Bubeck. Regret bounds and minimax policies under partial monitoring. *J. Mach. Learn. Res.*, 9999:2785–2836, December 2010. ISSN 1532-4435. URL `http://portal.acm.org/citation.cfm?id=1953011.1953023`.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1013689704352.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2003. ISSN 0097-5397.

B. Awerbuch and R. D. Kleinberg. Adaptive routing with end-to-end feedback: distributed learning and geometric approaches. In *STOC*, pages 45–53, 2004. ISBN 1-58113-852-0.

A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, volume 2 of *MPS/SIAM Series on Optimization*. SIAM, Philadelphia, 2001.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi, Y. Mansour, and G. Stoltz. Improved second-order bounds for prediction with expert advice. *Mach. Learn.*, 66(2-3):321–352, 2007. ISSN 0885-6125.

V. Dani and T. P. Hayes. Robbing the bandit: less regret in online geometric optimization against an adaptive adversary. In *SODA*, pages 937–943, 2006. ISBN 0-89871-605-5.

V. Dani, T. Hayes, and S. Kakade. The price of bandit information for online optimization. In *NIPS*, 2008.

A. D. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *SODA*, pages 385–394, 2005. ISBN 0-89871-585-7.

J. Hannan. Approximation to bayes risk in repeated play. *In M. Dresher, A. W. Tucker, and P. Wolfe, editors, Contributions to the Theory of Games, volume III*, pages 97–139, 1957.

E. Hazan and S. Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. In *The 21st Annual Conference on Learning Theory (COLT).*, 2008.

E. Hazan and S. Kale. Better algorithms for benign bandits. In *SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 38–47, Philadelphia, PA, USA, 2009a. Society for Industrial and Applied Mathematics.

E. Hazan and S. Kale. On stochastic and worst-case models for investing. In *Advances in Neural Information Processing Systems (NIPS) 22*, 2009b.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, March 1985.

H. B. McMahan and A. Blum. Online geometric optimization in the bandit setting against an adaptive adversary. In *COLT*, pages 109–123, 2004.

A.S. Nemirovskii. Interior point polynomial time methods in convex programming, 2004. Lecture Notes.

Y. E. Nesterov and A. S. Nemirovskii. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM, Philadelphia, 1994.

J. A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, April 2001. ISBN 0534399428. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0534399428`.

H. Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58(5): 527–535, 1952.

J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

Journal of Machine Learning Research 12 (2011) 1313-1347 Submitted 3/10; Revised 10/10; Published 4/11

A Family of Simple Non-Parametric Kernel Learning Algorithms

Jinfeng Zhuang ZHUA0016@NTU.EDU.SG
Ivor W. Tsang IVORTSANG@NTU.EDU.SG
Steven C.H. Hoi CHHOI@NTU.EDU.SG
School of Computer Engineering
Nanyang Technological University
50 Nanyang Avenue, Singapore 639798

Editor: Tony Jebara

Abstract

Previous studies of *Non-Parametric Kernel Learning* (NPKL) usually formulate the learning task as a Semi-Definite Programming (SDP) problem that is often solved by some general purpose SDP solvers. However, for N data examples, the time complexity of NPKL using a standard interior-point SDP solver could be as high as $O(N^{6.5})$, which prohibits NPKL methods applicable to real applications, even for data sets of moderate size. In this paper, we present a family of efficient NPKL algorithms, termed "**SimpleNPKL**", which can learn non-parametric kernels from a large set of pairwise constraints efficiently. In particular, we propose two efficient SimpleNPKL algorithms. One is SimpleNPKL algorithm with linear loss, which enjoys a *closed-form* solution that can be efficiently computed by the *Lanczos* sparse eigen decomposition technique. Another one is SimpleNPKL algorithm with other loss functions (including square hinge loss, hinge loss, square loss) that can be re-formulated as a saddle-point optimization problem, which can be further resolved by a fast iterative algorithm. In contrast to the previous NPKL approaches, our empirical results show that the proposed new technique, maintaining the same accuracy, is significantly more efficient and scalable. Finally, we also demonstrate that the proposed new technique is also applicable to speed up many kernel learning tasks, including *colored maximum variance unfolding*, *minimum volume embedding*, and *structure preserving embedding*.

Keywords: non-parametric kernel learning, semi-definite programming, semi-supervised learning, side information, pairwise constraints

1. Introduction

Kernel methods have been successfully applied in various real applications ranging from data mining, computer vision and bioinformatics, and often show the state-of-the-art performance (refer to Hofmann, Schölkopf, and Smola, 2008 and references therein). Empirical evidences show that the generalization performance of kernel methods is often dominated by the chosen kernel function. Inappropriate kernels could lead to sub-optimal or even poor results. Therefore, the choice of an effective kernel plays a crucial role in many kernel based machine learning methods. Typically, traditional kernel methods, for example, Support Vector Machines (SVMs), often adopt a predefined kernel function that is empirically chosen from a pool of parametric kernel functions, such as polynomial and Gaussian kernels. One major limitation of such an approach is that choosing an appropriate kernel function manually may require a certain level of expert knowledge, which may be difficult in some situations. Another limitation lies in the difficulty of tuning optimal parameters for the predefined parametric kernel functions.

To address these limitations, a bunch of research on learning effective kernels from data automatically has been actively explored recently. An example technique is *Multiple Kernel Learning* (MKL) (Lanckriet et al., 2004; Bach et al., 2004), which aims at learning a convex combination of several predefined parametric kernels in order to identify a good target kernel for the applications. MKL has been actively studied in many applications, including bio-informatics (Sonnenburg et al., 2006a,b), computer vision (Duan et al., 2009; Sun et al., 2009; Vedaldi et al., 2009), and natural language processing (Mao and Tsang, 2011), etc. Despite some encouraging results reported, these techniques often assume the target kernel function is of some *parametric* forms, which limits their capacity of fitting diverse patterns in real complex applications.

Instead of assuming some parametric forms for the target kernel, an emerging group of kernel learning studies are devoted to *Non-Parametric Kernel Learning* (NPKL) methods, which aim to learn a Positive Semi-Definite (PSD) kernel matrix directly from the data. Example techniques include Cristianini et al. (2002), Lanckriet et al. (2004), Zhu et al. (2005), Zhang and Ando (2006), Kulis et al. (2006), Hoi et al. (2007), Kulis et al. (2009) and Li et al. (2009); Mao and Tsang (2010). NPKL provides a flexible learning scheme of incorporating prior/side information into the known similarity measures such that the learned kernel can exhibit better ability to characterize the data similarity. However, due to the PSD constraint, the resulting optimization task of NPKL is often in the form of Semi-Definite Programing (SDP). Many existing studies have simply solved such SDP problems by some general purpose SDP solvers, which often have the time complexity of $O(N^{6.5})$, making the NPKL solution infeasible to real world large-scale applications.

In this paper, we aim at addressing the efficiency and scalability issues related to the NPKL techniques proposed by Hoi et al. (2007) and Zhuang et al. (2009), which have shown the state-of-the-art empirical performance in several applications (Zhuang and Hoi, 2010). In particular, the main contributions of this paper include:

1. We propose a family of Simple Non-Parametric Kernel Learning (SimpleNPKL) algorithms for efficient and scalable non-parametric kernel learning.

2. We present the first SimpleNPKL algorithm with linear loss function to learn non-parametric kernels from pairwise constraints. The algorithm enjoys a *closed-form* solution that can be computed efficiently by sparse eigen-decomposition techniques, for example, the *Lanczos* algorithm.

3. To achieve more robust performance, we propose the second SimpleNPKL algorithm that has other loss functions (including square hinge loss, hinge loss and square loss), which can be re-formulated as a *mini-max optimization* problem. This optimization can be solved by an efficient iterative projection algorithm that mainly involves the computation of sparse eigen decomposition.

4. To further speed up the SimpleNPKL algorithm of other loss functions, we investigate some active constraint selection techniques to reduce the computation cost at each iteration step.

5. We conducted extensive experiments, which show that SimpleNPKL is significantly more efficient than existing NPKL methods. With the same linear loss function, SimpleNPKL is

on average 40 times faster than the original NPKL using a standard SDP solver. This makes the NPK learning techniques practical to large-scale applications.

6. We extend the proposed SimpleNPKL scheme to resolve other non-parametric kernel learning problems, including *colored maximum variance unfolding* (Song et al., 2008), *minimum volume embedding* (Shaw and Jebara, 2007), and *structure preserving embedding* (Shaw and Jebara, 2009). The encouraging results show that our technique is able to speed up the existing non-parametric kernel learning solutions significantly for several real-world applications.

The rest of this paper is organized as follows. Section 2 presents some background of kernel learning, briefly reviews some representative work on kernel learning research, and indicates the motivations of our work. Section 3 introduces a framework of Non-parametric Kernel Learning (NPKL) from pairwise constraints proposed by Hoi et al. (2007). Section 4 describes our proposed SimpleNPKL algorithms, which aim to resolve the NPKL task efficiently. Section 5 discusses some implementation issues for developing a fast solver in practice. Section 6 extends our technique to speed up other kernel learning methods. Section 7 gives our empirical results and Section 8 concludes this work.

2. Background Review and Related Work

In this Section, we review some backgrounds of kernel methods, and related work on kernel learning research.

2.1 Notations

For the notation throughout the paper, we adopt bold upper case letter to denote a matrix, for example, $\mathbf{A} \in \mathbb{R}^{m\times n}$, and A_{ij} to denote the entry at the ith row and jth column of the matrix $\mathbf{A}$, and bold lower case letter to denote a vector, for example, $\mathbf{x} \in \mathbb{R}^d$. We use $\mathbf{0}$ and $\mathbf{1}$ to denote the column vectors with all zeros and all ones, respectively, and $\mathbf{I}$ to denote an identity matrix. For some algebraic operations:

- $\mathbf{x}'$ denotes the transpose of $\mathbf{x}$;
- $[\mathbf{x}]_i$ denotes the ith element of $\mathbf{x}$;
- $\mathbf{x}^p$ denotes the element-wise power of $\mathbf{x}$ with degree p;
- $|\mathbf{x}|$ denotes the vector with entries equal to the absolute value of the entries of $\mathbf{x}$;
- $\|\mathbf{x}\|_p$ denotes p-norm of $\mathbf{x}$, that is, $\sqrt[p]{\sum_i [\mathbf{x}^p]_i}$;
- $\mathbf{x}_i \circ \mathbf{x}_j$ denotes the element-wise multiplication between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$;
- $\mathbf{x} \geq \mathbf{0}$ means all entries in $\mathbf{x}$ is larger than or equal to 0;
- $\mathbf{K} \succeq \mathbf{0}$ denotes a matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ that is symmetric and positive semi-definite;
- $\mathbf{K}^p$ denotes the power of a symmetric matrix $\mathbf{K}$ with degree p;
- $\mathrm{tr}\,\mathbf{K} = \sum_i K_{ii}$ denotes the trace of a matrix $\mathbf{K}$;
- $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}\,\mathbf{AB} = \sum_{ij} A_{ij}B_{ij}$ computes the inner product between two square matrices $\mathbf{A}$ and $\mathbf{B}$. We also use it to denote general inner product of two square matrices.
- $\|\mathbf{K}\|_F = \sqrt{\sum_{ij} K_{ij}^2} = \sqrt{\mathrm{tr}\,\mathbf{KK}}$ denotes the Frobenius norm of a matrix $\mathbf{K}$;
- $\mathbf{A} \circ \mathbf{B}$ denotes the element-wise multiplication between two matrices $\mathbf{A}$ and $\mathbf{B}$.

2.2 Kernel Methods

In general, kernel methods work by embedding data in some Hilbert spaces, and searching for linear relations in the Hilbert spaces. The embedding is often done implicitly by only specifying inner products between any pair of examples (Hofmann et al., 2008). More formally, given an input space $\mathcal{X}$, and an embedding space $\mathcal{F}$, we can define a mapping $\Phi : \mathcal{X} \to \mathcal{F}$. For any two examples $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{x}_j \in \mathcal{X}$, the function k that returns the inner product between the two embedded examples in the space $\mathcal{F}$ is known as the kernel function, that is,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle.$$

Given the kernel function k, a matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ is called a *kernel matrix*, also known as *gram matrix*, if $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for a collection of examples $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$. Note that the choice of kernel plays a central role for the success of kernel methods. However, the selection of proper kernels is nontrivial. An inappropriate kernel could result in sub-optimal or even poor performances. Therefore, learning kernels from data has become an active research topic.

2.3 Kernel Learning

We refer the term *kernel learning* to the problem of learning a kernel function or a kernel matrix from given data, corresponding to the inductive and transductive learning setting, respectively. Due to the large volume of works on this topic, we do not intend to make this Section encyclopedic. Instead, we summarize some key ideas behind representative kernel learning schemes. We discuss the strengths and limitations of existing NPKL methods, which motivates our efficient SimpleNPKL solution.

2.3.1 MULTIPLE KERNEL LEARNING AND BEYOND

Multiple kernel learning (MKL), initiated by Lanckriet et al. (2004), has been widely studied in classical supervised learning tasks. The goal is to learn both the associated kernel of a Reproducing Kernel Hilbert Space (RKHS) and the classifier in this space simultaneously:

$$\begin{aligned} \min_{\mathbf{K} \in \mathcal{K}} \max_{\alpha} \quad & \alpha'\mathbf{1} - \frac{1}{2}\langle (\alpha \circ \mathbf{y})(\alpha \circ \mathbf{y})', \mathbf{K} \rangle \\ \text{s.t.} \quad & \alpha'\mathbf{y} = 0, \;\; 0 \le \alpha_i \le C, \end{aligned} \tag{1}$$

where the solution space $\mathcal{K}$ is assumed to be in a convex hull spanned from m basic kernels: $\mathbf{K} = \{\sum_i p_i \mathbf{K}_i : 0 \le p_i \le 1, i = 1, \ldots, m\}$. Thus the optimization over $\mathbf{K}$ is reduced to optimizing the weight vector $\mathbf{p}$. Many studies have been focused on how to efficiently solve the optimization in (1) (Bach et al., 2004; Sonnenburg et al., 2006b; Rakotomamonjy et al., 2008; Xu et al., 2008).

The assumption of MKL on the target kernel $\mathbf{K} = \sum_i p_i \mathbf{K}_i$ implies to concatenate the mapped feature spaces. Therefore, MKL is a natural choice where the data has multiple views or heterogeneous representations. Apparently, there is "no free lunch" for kernel selection. Based on different assumptions about the optimization domain $\mathcal{K}$, one can propose different objective functions. For example, generating a series of base kernels by varying the free kernel parameters could make the cardinality $|\mathcal{K}|$ arbitrarily large. Argyriou et al. (2005) and Gehler and Nowozin (2008) discussed some interesting techniques for such situation. Other variants of MKL techniques can also be found in Lewis et al. (2006), Gönen and Alpaydin (2008), Varma and Babu (2009) and Zhuang et al. (2011).

One basic motivation of kernel learning is to further relax the optimization domain $\mathcal{K}$ such that the learned kernel can be as flexible as possible to fit the complex data. This motivates *Non-Parametric Kernel Learning* (NPKL) methods, which do not assume any parametric form of the target kernel functions/matrices.

2.3.2 NON-PARAMETRIC KERNEL LEARNING

By simply relaxing the optimization domain $\mathcal{K}$, one can turn a regular kernel learning scheme to some NPKL formulations. For example, a recent approach, called *indefinite kernel learning* (Luss and d'Aspremont, 2008; Chen and Ye, 2008; Ying et al., 2010), extends the MKL formulation to learn non-parametric kernels from an indefinite kernel $\mathbf{K}_0$, which does not assume the convex hull assumption. The indefinite kernel learning rewrites the objective function of (1) as follows:

$$\min_{\mathbf{K}\succeq\mathbf{0}} \max_{\alpha} \quad \alpha'\mathbf{1} - \frac{1}{2}\langle(\alpha\circ\mathbf{y})(\alpha\circ\mathbf{y})', \mathbf{K}\rangle + \gamma\|\mathbf{K}-\mathbf{K}_0\|_F^2, \tag{2}$$

where $\mathbf{K}_0$ is an initial kernel, which could be indefinite.

The kernel learning formulation discussed above aims to optimize both the classifier and the kernel matrix simultaneously. Some theoretical foundations, such as existence and uniqueness of the target kernel, were given in Micchelli and Pontil (2005) and Argyriou et al. (2005).

Another line of kernel learning research mainly focuses on optimizing the kernel only with respect to some criteria under some prior constraints or heuristics. An important technique is the kernel target alignment criterion proposed in Cristianini et al. (2002), which guides the kernel learning task to optimize the kernel by maximizing the alignment between the training data examples and the class labels of the training examples:

$$\max_{\mathbf{K}\succeq\mathbf{0}} \quad \frac{\langle\mathbf{K}_{N_l}, \mathbf{T}\rangle}{\sqrt{\langle\mathbf{K}_{N_l}, \mathbf{K}_{N_l}\rangle\langle\mathbf{T}, \mathbf{T}\rangle}}, \tag{3}$$

where $\mathbf{T} = \mathbf{y}\mathbf{y}'$ is the outer product of labels, $\mathbf{K}_{N_l}$ is the sub-matrix of which the entry values are kernel evaluation on N_l labeled data examples. Note that $\mathbf{T}$ could be obtained by empirical experiments and more general than class labels. The objective (3) only involves the labeled data. A popular assumption is to treat $\mathbf{K}$ to be spanned by the eigen-vectors of some known kernel defined over both the labeled and unlabeled data (Chapelle et al., 2003; Zhu et al., 2005; Hoi et al., 2006; Zhang and Ando, 2006; Johnson and Zhang, 2008): $\mathcal{K} = \{\sum_i \lambda_i \mathbf{v}\mathbf{v}' : \lambda_i \geq 0\}$. Thus the optimization variables are reduced from the entire kernel matrix $\mathbf{K}$ to the kernel spectrum λ.

Recently Hoi et al. (2007) proposed an NPKL technique that aims to learn a fully non-parametric kernel matrix from pairwise constraints. The target kernel is maximally aligned to the constraint matrix $\mathbf{T}$ and minimally aligned to the graph Laplacian. The objective can be deemed as a form of kernel target alignment without normalization. Since our proposed family of SimpleNPKL algorithms follows this framework, we will discuss the details of this formulation in Section 3.

Besides the normalized inner product, which measures the similarity between $\mathbf{K}$ and the target $\mathbf{T}$, researchers have also proposed dissimilarity based criteria. In fact, the preceding indefinite kernel learning (2) employs the Euclidean distance to measure the dissimilarity between kernels. Besides, another example in Kulis et al. (2006) employed the Bregman divergence to measure distance between $\mathbf{K}$ and a known kernel $\mathbf{K}_0$:

$$\min_{\mathbf{K}\succeq\mathbf{0}} \quad D_\phi(\mathbf{K}, \mathbf{K}_0) \triangleq \operatorname{tr}\mathbf{K}\mathbf{K}_0^{-1} - \log\det(\mathbf{K}\mathbf{K}_0^{-1}) - N, \tag{4}$$

where D_ϕ is a Bregman divergence (Kulis et al., 2006).

The optimization over the above learning objective function (3) or (4) will simply return the trivial solution $\mathbf{K}_0$ without additional constraints, which would make NPKL meaningless. In practice, some prior knowledge about the target kernel will be added to constrain the solution space $\mathcal{K}$. Most of the existing constraints over the entries of $\mathbf{K}$ could be expressed by $\operatorname{tr}\mathbf{KT} \leq b$. For example, as discussed in Kwok and Tsang (2003), the square of distance between two data examples $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space can be expressed by $\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|_2^2 = K_{ii} + K_{jj} - 2K_{ij} = \operatorname{tr}\mathbf{KT}_{ij}$, where $\mathbf{T}_{ij}$ is a matrix of $N \times N$ only taking non-zeros at $T_{ii} = T_{jj} = 1, T_{ij} = T_{ji} = -1$. Moreover, one can introduce slack variables for soft constraints.

Besides, some regularization terms over kernel $\mathbf{K}$ are often included during the optimization phase. For example, fixing the trace $\operatorname{tr}\mathbf{K} = 1$ is rather common in SDP solvers.

At last, we summarize the typical schemes of existing NPKL methods:

- To encourage the similarity (e.g., kernel target alignment) or penalize the distance (e.g., Bregman divergence) to some prior similarity information;
- To enforce some constraints to the kernel $\mathbf{K}$ with prior heuristics, such as distance constraint $K_{ii} + K_{jj} - 2K_{ij} = d_{ij}^2$, or side information, etc; and
- To include regularization terms over $\mathbf{K}$ to control capacity, such as $\operatorname{tr}\mathbf{K} = 1$.

By the above steps, NPKL provides a flexible scheme to incorporate more prior information into the target kernel. Due to the non-parametric nature, the solution space $\mathcal{K}$ is capable of fitting diverse empirical data such that the learned kernel $\mathbf{K}$ can be more effective and powerful to achieve better empirical performance than traditional parametric kernel functions.

2.3.3 OPTIMIZATION ASPECTS

Despite the powerful capacity achieved by NPKL, one key challenge with NPKL is the difficulty of the resulting optimization problem, in which

- the whole gram matrix $\mathbf{K}$ is treated as the optimization variable, that is, $O(N^2)$ variables;
- the kernel matrix $\mathbf{K}$ must be positive semi-definite.

As a result, NPKL is often turned into a Semi-Definite Programming (SDP) problem. For instance, a NPKL problem to learn a kernel matrix $\mathbf{K}$ with m linear constraints is written as follows:

$$\max_{\mathbf{K}\succeq\mathbf{0}} \operatorname{tr}\mathbf{CK} \quad : \quad \operatorname{tr}\mathbf{T}_{ij}\mathbf{K} = b_{ij}, \tag{5}$$

where $\mathbf{C}$ and $\mathbf{T}_{ij}$'s are $N \times N$ symmetric matrices and b_{ij}'s are scalars, and its dual problem can be rewritten as:

$$\min_{\mathbf{y}} \mathbf{b}'\mathbf{y} \quad : \quad \mathbf{C} - \sum_{(i,j)} \mathbf{T}_{ij} y_{ij} \succeq \mathbf{0}, \tag{6}$$

where $\mathbf{y}$ is the vector of dual variables y_{ij}'s for the linear constraints in (5) and $\mathbf{b}$ is the vector of b_{ij}'s.

Typically, this SDP problem of NPKL is usually solved by applying a general purpose SDP solver. Among various SDP solvers, the *interior-point algorithm* is one of the state-of-the-art solutions (Boyd and Vandenberghe, 2004). From Lobo et al. (1998), the time complexity per iteration of the SDP problem (6) is $O(m^2N^2)$. Using the primal-dual method for solving this SDP, the accuracy of a given solution can be improved by an absolute constant factor in $O(\sqrt{N})$ iterations (Nesterov and Nemirovskii, 1994). When m approaches to $O(N^2)$, the overall computation complexity is often as high as $O(N^{6.5})$, which makes NPKL inapplicable to real applications.

In this work, we focus on solving the efficiency and scalability issues of NPKL (Zhuang et al., 2009). In particular, we propose a family of efficient SimpleNPKL algorithms that can solve large-scale NPKL problems efficiently. Moreover, we also show that the proposed algorithms are rather general, which can be easily extended to solving other kernel learning applications, including dimensionality reduction and data embedding applications.

3. A Framework of Non-Parametric Kernel Learning from Pairwise Constraints

In this Section, we introduce the framework of Non-Parametric Kernel Learning (NPKL) (Hoi et al., 2007; Zhuang et al., 2009), which aims to learn non-parametric kernels from side information, which is presented in a collection of must-link and cannot-link pairs.

3.1 Side / Label Information

Let $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ denote the entire data collection, where each data point $\mathbf{x}_i \in \mathcal{X}$. Consider a set of N_l labeled data examples, $\mathcal{L} = \{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_{N_l}, y_{N_l})\}$, one can use $y_i y_j$ as the similarity measurement for any two patterns $\mathbf{x}_i$ and $\mathbf{x}_j$. Sometimes, it is possible that the class label information is not readily available, while it is easier to obtain a collection of similar (positive) pairwise constraints $\mathcal{S}$ (known as "must-links", that is, the data pairs share the same class) and a collection of dissimilar (negative) pairwise constraints $\mathcal{D}$ (known as "cannot-links", that is, the data pairs have different classes). These pairwise constraints are often referred to as *side information*.

In general, kernel learning with labeled data can be viewed as a special case of kernel learning with side information (Kwok and Tsang, 2003; Kulis et al., 2006; Hoi et al., 2007), that is, one can construct the sets of pairwise constraints $\mathcal{S}$ and $\mathcal{D}$ from $\mathcal{L}$. In real applications, it is often easier to detect pairwise constraint while the class label is difficult to obtain. For example, in bioinformatics, the interaction between two proteins can be identified by empirical experiments. These interactions are expressed naturally by pairwise constraints. However, it could be very difficult to judge the protein function, which corresponds to class labels. In the sequel, we focus on learning kernels from pairwise constraints.

Given $\mathcal{S}$ and $\mathcal{D}$, we construct a similarity matrix $\mathbf{T} \in \mathbb{R}^{N \times N}$ to represent the pairwise constraints, that is,

$$T_{ij} = \begin{cases} +1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S} \\ -1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

A straightforward and intuitive principle for kernel learning is that the kernel entry K_{ij} should be aligned with the side information T_{ij} as much as possible (Cristianini et al., 2002), that is, the alignment $T_{ij} K_{ij}$ of each kernel entry is maximized.

3.2 Locality Preserving Regularization

In addition to side/label information, preserving the intrinsic geometric structure of the data have also been explored to improve the performance of kernel learning. Typically, most existing kernel learning approaches (Kulis et al., 2006; Hoi et al., 2007; Hoi and Jin, 2008) adopt the low dimensional data manifold (Sindhwani et al., 2005) for preserving the locality of the data in kernel learning. The following reviews an approach for exploring low dimensional data manifold in kernel learning (Hoi et al., 2007).

Let us denote by $f(\mathbf{x},\mathbf{x}')$ a similarity function that measures the similarity between any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\mathbf{S} \in \mathbb{R}^{N\times N}$ is a similarity matrix where each element $S_{ij} = f(\mathbf{x}_i,\mathbf{x}_j) \geq 0$. Note that $f(\cdot,\cdot)$ does not have to be a kernel function that satisfies the Mercer's condition. For a given N data examples, a kernel matrix $\mathbf{K}$ can be expressed as $\mathbf{K} = \mathbf{V}'\mathbf{V} \succeq \mathbf{0}$, where $\mathbf{V} = [\mathbf{v}_1,\ldots,\mathbf{v}_N]$ is the matrix of the embedding of the N data examples. The regularizer of the kernel matrix $\mathbf{K}$, which captures the local dependency between the embedding of $\mathbf{v}_i$ and $\mathbf{v}_j$ (i.e., the low dimensional embedding of similar data examples should be similar w.r.t. the similarity S_{ij}), can be defined as:

$$\begin{aligned}\Omega(\mathbf{V},\mathbf{S}) &= \frac{1}{2}\sum_{i,j=1}^{N} S_{ij}\left\| \frac{\mathbf{v}_i}{\sqrt{D_i}} - \frac{\mathbf{v}_j}{\sqrt{D_j}} \right\|_2^2 \\ &= \mathrm{tr}\,(\mathbf{V}\mathbf{L}\mathbf{V}') = \mathrm{tr}\,(\mathbf{L}\mathbf{K}), \qquad (8)\end{aligned}$$

where $\mathbf{L}$ is the graph Laplacian matrix defined as:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}, \qquad (9)$$

where $\mathbf{D} = \mathrm{diag}(D_1, D_2, \ldots, D_N)$ is a diagonal matrix with the diagonal elements defined as $D_i = \sum_{j=1}^{N} S_{ij}$.

3.3 Formulation of Non-Parametric Kernel Learning

Taking into consideration of both the side information in (7) and the regularizer in (8), the NPKL problem is then formulated into the *loss + regularization* framework (Hoi et al., 2007) as follows:

$$\min_{\mathbf{K}\succeq\mathbf{0}} \quad \mathrm{tr}\,\mathbf{L}\mathbf{K} + C\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \ell\big(T_{ij}K_{ij}\big), \qquad (10)$$

which generally belongs to a Semi-Definite Programming (SDP) problem (Boyd and Vandenberghe, 2004). Here, $C > 0$ is a tradeoff parameter to control the empirical loss[1] $\ell(\cdot)$ of the alignment $T_{ij}K_{ij}$ of the target kernel and the dependency among data examples with respect to the intrinsic data structure.

4. SimpleNPKL: Simple Non-Parametric Kernel Learning

In this Section, we present a family of efficient algorithms for solving the NPKL problem in (10). We refer to the proposed efficient algorithms as "SimpleNPKL" for short.

4.1 Regularization on K

As aforementioned, the solution space of NPKL has been relaxed to boost its flexibility and capacity of fitting diverse patterns. However, arbitrarily relaxing the solution space $\mathcal{K}$ could result in over-fitting. To alleviate this problem, we introduce a regularization term:

$$\mathrm{tr}\,(\mathbf{K}^p), \qquad (11)$$

1. The common choice of the loss function $\ell(\cdot)$ can be hinge loss, square hinge loss or linear loss.

where $p \geq 1$ is a parameter to regulate the capacity of $\mathbf{K}$. Similar regularization terms on $\mathbf{K}$ have also been adopted in previous kernel learning studies. For example, Cristianini et al. (2002) used $\|\mathbf{K}\|_F = \sqrt{\langle \mathbf{K}, \mathbf{K} \rangle} = \sqrt{\text{tr}\,(\mathbf{K}^2)}$ in the objective to penalize the complexity of $\mathbf{K}$; while Lanckriet et al. (2004) proposed to adopt a hard constraint $\text{tr}\,(\mathbf{K}) \leq B$, where $B > 0$ a constant, to control the capacity of $\mathbf{K}$.

We refer the modified NPK learning problem with the regularization term (11) either in the objective or in the constraint to as Simple Non-Parametric Kernel Learning (SimpleNPKL), which can be solved efficiently without engaging any standard SDP solvers. Next we present two SimpleNPKL algorithms that adopt several different types of loss functions.

4.2 SimpleNPKL with Linear Loss

First of all, we consider a linear loss function $\ell(f) = -f$, and rewrite the formulation of (10) as the SimpleNPKL formulation:

$$\min_{\mathbf{K}} \text{tr} \left(\left(\mathbf{L} - C \sum_{(i,j) \in (\mathcal{S} \cup \mathcal{D})} \mathbf{T}_{ij} \right) \mathbf{K} \right) \; : \; \mathbf{K} \succeq \mathbf{0}, \; \text{tr}\, \mathbf{K}^p \leq B, \tag{12}$$

where $\mathbf{T}_{ij}$ is the matrix of setting the (i, j)-th entry to T_{ij} and other entries to 0. To solve this problem, we first present a proposition below.

Proposition 1 *Given $\mathbf{A}$ is any symmetric matrix such that $\mathbf{A} = \mathbf{P} diag(\sigma) \mathbf{P}'$, where $\mathbf{P}$ contains columns of orthonormal eigenvectors of $\mathbf{A}$ and σ is a vector of the corresponding eigenvalues, and B is any positive constant, the optimal solution $\mathbf{K}^*$ to the following SDP problem for $p > 1$:*

$$\max_{\mathbf{K}} tr\, \mathbf{AK} \; : \; \mathbf{K} \succeq \mathbf{0}, \; tr\, \mathbf{K}^p \leq B, \tag{13}$$

can be expressed as the following closed-form solution:

$$\mathbf{K}^* = \left(\frac{B}{tr\, \mathbf{A}_+^{\frac{p}{p-1}}} \right)^{\frac{1}{p}} \mathbf{A}_+^{\frac{1}{p-1}} \tag{14}$$

where $\mathbf{A}_+ = \mathbf{P} diag(\sigma_+) \mathbf{P}'$, and σ_+ is a vector with entries equal to $\max(0, [\sigma]_i)$.

For $p = 1$, the optimal solution $\mathbf{K}^*$ can be expressed as the following closed-form solution:

$$\mathbf{K}^* = B\mathbf{A}_1$$

where $\mathbf{A}_1 = \mathbf{P} diag(\sigma_1) \mathbf{P}'$, and σ_1 is a vector with entries equal to $\frac{1}{\sum_{i:[\sigma]_i = \max_i [\sigma]_i} 1}$ for all i that $[\sigma]_i = \max_i [\sigma]_i$; otherwise, the entries are zeros.

Proof By introducing a dual variable $\gamma \geq 0$ for the constraint $\text{tr}\, \mathbf{K}^p \leq B$, and $\mathbf{Z} \in \mathcal{S}_+^n$ ($\mathcal{S}_+^n$ is self-dual) for the constraint $\mathbf{K} \succeq \mathbf{0}$, we have the Lagrangian of (13):

$$\mathcal{L}(\mathbf{K}; \gamma, \mathbf{Z}) = \text{tr}\, \mathbf{AK} + \gamma(B - \text{tr}\, \mathbf{K}^p) + \text{tr}\, \mathbf{KZ}.$$

By the Karush-Kuhn-Tucker (KKT) conditions, we have:

$$\mathbf{A} - \gamma p \mathbf{K}^{p-1} + \mathbf{Z} = \mathbf{0} \quad \text{and} \quad \text{tr}\, \mathbf{KZ} = 0.$$

First, we show that tr $(\mathbf{KZ}) = 0$ is equivalent to $\mathbf{KZ} = \mathbf{ZK} = \mathbf{0}$. Since $\mathbf{K} \succeq \mathbf{0}, \mathbf{Z} \succeq \mathbf{0}$, we have tr $(\mathbf{KZ}) =$ tr $(\mathbf{K}^{1/2}\mathbf{K}^{1/2}\mathbf{Z}^{1/2}\mathbf{Z}^{1/2}) = \|\mathbf{K}^{1/2}\mathbf{Z}^{1/2}\|_F^2$. Thus, tr $(\mathbf{KZ}) = 0$ follows that $\mathbf{K}^{1/2}\mathbf{Z}^{1/2} = \mathbf{0}$. Pre-multiplying by $\mathbf{K}^{1/2}$ and post-multiplying by $\mathbf{Z}^{1/2}$ yields $\mathbf{KZ} = \mathbf{0}$, which in turn implies $\mathbf{KZ} = \mathbf{0} = (\mathbf{KZ})' = \mathbf{ZK}$. Hence, $\mathbf{K}$ and $\mathbf{Z}$ can be simultaneously diagonalized by the same set of orthonormal eigenvectors (Alizadeh et al., 1997). From the first KKT condition we have $\mathbf{A} = \gamma p \mathbf{K}^{p-1} - \mathbf{Z}$. Consequently, $\mathbf{A}$ can also be diagonalized with the same eigenvectors as $\mathbf{K}$ and $\mathbf{Z}$.

Assume $\mathbf{A} = \mathbf{P}\text{diag}(\sigma)\mathbf{P}'$, where $\mathbf{P}$ contains columns of orthonormal eigenvectors of $\mathbf{A}$, and σ is the vector of the corresponding eigenvalues. Then, $\mathbf{K} = \mathbf{P}\text{diag}(\lambda)\mathbf{P}'$ and $\mathbf{Z} = \mathbf{P}\text{diag}(\mu)\mathbf{P}'$, where $\lambda \geq \mathbf{0}$ and $\mu \geq \mathbf{0}$ denote the vector of the eigenvalues of $\mathbf{K}$ and $\mathbf{Z}$ respectively. Therefore, we have

$$\begin{aligned} \text{tr}\,\mathbf{K}^p &= \|\lambda\|_p^p \leq B, & (15)\\ \text{tr}\,\mathbf{AK} &= \lambda'\sigma, & (16)\\ \sigma &= \gamma p \lambda^{p-1} - \mu, & (17)\\ \lambda'\mu &= 0. & (18) \end{aligned}$$

Together with $\lambda \geq \mathbf{0}$ and $\mu \geq \mathbf{0}$, and from (18), $[\lambda]_i$ and $[\mu]_i$ cannot be both non-zeros. Hence, from (17), we know $\sigma_+ = \gamma p \lambda^{p-1}$ contains all positive components of σ. Moreover, from (16) and $\lambda \geq \mathbf{0}$, together with the constraint (15), the SDP problem (13) is reduced to

$$\max_{\lambda} \lambda'\sigma_+ \;:\; \|\lambda\|_p^p \leq B.$$

By Hölder inequality, we have $\lambda'\sigma_+ \leq \|\lambda\|_p \|\sigma_+\|_q$, where it holds for $1/p + 1/q = 1$. The equality is achieved if and only if $|\lambda|^p$ and $|\sigma_+|^q$ are linearly dependent. Thus we can scale $\mathbf{K}$ satisfying (15) to arrive at the closed-form solution of $\mathbf{K}$ in (14) for $p > 1$.

For $p = 1$, from Equations (15) and (16), the optimization task is simplified as $\max \lambda'\sigma : \lambda \geq 0, \|\lambda\|_1 \leq B$. Due to the linearity, the maximum objective value is obtained by choosing $[\lambda]_i = B / \sum_{i:[\sigma]_i = \max_i [\sigma]_i} 1$ for all i that $[\sigma]_i = \max_i [\sigma]_i$; otherwise, $[\lambda]_i = 0$. ■

Based on Proposition 1, we can easily solve the SimpleNPKL problem. In particular, by setting $\mathbf{A} = C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \mathbf{T}_{ij} - \mathbf{L}$, we can directly compute the optimal $\mathbf{K}^*$ to SimpleNPKL of (12) using sparse eigen-decomposition as in (14). Thus the computation cost of SimpleNPKL with linear loss is dominated by eigen-decomposition. It is clear that this can significantly reduce the time cost for the NPKL tasks. Alternatively, we add tr $(\mathbf{K}^p)$ directly into the objective, and arrive at the following formulation:

$$\min_{\mathbf{K}} \text{tr}\left(\left(\mathbf{L} - C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \mathbf{T}_{ij}\right)\mathbf{K}\right) + \frac{G}{p}\,\text{tr}\,\mathbf{K}^p \;:\; \mathbf{K} \succeq \mathbf{0},$$

where $G > 0$ is a tradeoff parameter. To solve this problem, we first present a proposition below.

Proposition 2 *Given* $\mathbf{A}$ *is any symmetric matrix such that* $\mathbf{A} = \mathbf{P}\,diag(\sigma)\mathbf{P}'$, *where* $\mathbf{P}$ *contains columns of orthonormal eigenvectors of* $\mathbf{A}$ *and* σ *is a vector of the corresponding eigenvalues, and* B *is any positive constant, the optimal solution* $\mathbf{K}^*$ *to the following SDP problem for* $p > 1$*:*

$$\max_{\mathbf{K}} tr\,\mathbf{AK} - \frac{G}{p} tr\,\mathbf{K}^p \;:\; \mathbf{K} \succeq \mathbf{0}, \qquad (19)$$

can be expressed as the following closed-form solution:

$$\mathbf{K}^* = \left(\frac{1}{G}\mathbf{A}_+\right)^{\frac{1}{p-1}} \tag{20}$$

where $\mathbf{A}_+ = \mathbf{P}diag(\sigma_+)\mathbf{P}'$, *and* σ_+ *is a vector with entries equal to* $\max(0, [\sigma]_i)$.

Following the techniques in the proof of Proposition 1, we obtain (20) immediately. If we set $G = \left(\frac{1}{B}\text{tr}\, \mathbf{A}_+^{\frac{p}{p-1}}\right)^{\frac{p-1}{p}}$, these two formulations result in exactly the same solution. Moreover, if we set $B = \text{tr}\, \mathbf{A}_+^{\frac{p}{p-1}}$, it means we just use the projection $\mathbf{A}_+$ as $\mathbf{K}$. No re-scaling of $\mathbf{A}_+$ is performed. In the sequel, we consider the regularization tr $\mathbf{K}^p$ with $p = 2$ for its simplicity and smoothness.

4.3 SimpleNPKL with Square Hinge Loss

Although the formulation with linear loss in (12) gives rise to a closed-form solution for the NPKL, one limitation of the NPKL formulation with linear loss is that it may be sensitive to noisy data due to the employment of the linear loss function. To address this issue, in this section, we present another NPKL formulation that uses (square) hinge loss $\ell(f) = (\max(0, 1-f))^d/d$, which sometimes can be more robust, where $d = 1$ (hinge loss) or 2 (square hinge loss). We first focus on the NPKL formulation with square hinge loss, which can be written into the following constrained optimization:

$$\min_{\mathbf{K},\varepsilon_{ij}} \quad \text{tr}\, \mathbf{LK} + \frac{C}{2}\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \varepsilon_{ij}^2 \tag{21}$$

$$\text{s.t.} \quad \forall (i,j) \in (\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} \geq 1-\varepsilon_{ij}, \tag{22}$$
$$\mathbf{K}\succeq \mathbf{0}, \text{tr}\, \mathbf{K}^p \leq B.$$

Note that we ignore the constraints $\varepsilon_{ij} \geq 0$ since they can be satisfied automatically. However, (21) is not in the form of (13), and thus there is no longer a closed-form solution for $\mathbf{K}$.

4.3.1 DUAL FORMULATION: THE SADDLE-POINT MINIMAX PROBLEM

By Lagrangian theory, we introduce dual variables α_{ij}'s ($\alpha_{ij} \geq 0$) for the constraints in (22), and derive a partial Lagrangian of (21):

$$\text{tr}\, \mathbf{LK} + \frac{C}{2}\sum_{(i,j)} \varepsilon_{ij}^2 - \sum_{(i,j)} \alpha_{ij}(T_{ij}K_{ij} - 1 + \varepsilon_{ij}). \tag{23}$$

For simplicity, we use $\sum_{(i,j)}$ to replace $\sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})}$ in the sequel. By setting the derivatives of (23) w.r.t. the primal variables ε_{ij}'s to zeros, we have

$$\forall (i,j) \in (\mathcal{S}\cup\mathcal{D}),\ C\varepsilon_{ij} = \alpha_{ij} \geq 0$$

and substituting them back into (23), we arrive at the following saddle-point minimax problem $J(\mathbf{K}, \alpha)$:

$$\max_{\alpha}\min_{\mathbf{K}} \quad \text{tr}\left(\left(\mathbf{L} - \sum_{(i,j)} \alpha_{ij}\mathbf{T}_{ij}\right)\mathbf{K}\right) - \frac{1}{2C}\sum_{(i,j)} \alpha_{ij}^2 + \sum_{(i,j)} \alpha_{ij} \tag{24}$$
$$\text{s.t.} \quad \mathbf{K}\succeq \mathbf{0},\ \text{tr}\, \mathbf{K}^p \leq B,\ \forall (i,j) \in \mathcal{S}\cup\mathcal{D},\ \alpha_{ij} \geq 0,$$

where $\alpha = [a_{ij}]$ denotes a matrix of dual variables α_{ij}'s for $(i,j) \in \mathcal{S} \cup \mathcal{D}$, and other entries are zeros. This problem is similar to the optimization problem of DIFFRAC (Bach and Harchaoui, 2008), in which K and α can be solved by an iterative manner.

4.3.2 ITERATIVE ALGORITHM

In this subsection, we present an iterative algorithm which follows the similar update strategy in Boyd and Xiao (2005): 1) For a fixed α_{t-1}, we can let $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^{t-1} \mathbf{T}_{ij} - \mathbf{L}$. Based on Proposition 1, we can compute the closed form solution $\mathbf{K}_t$ to (24) using (14); 2) For a fixed $\mathbf{K}_t$, we can update α_t using $\alpha_t = (\alpha_{t-1} + \eta_t \nabla J_t)_+$; 3) Step 1) and 2) are iterated until convergence. Here J denotes the objective function (24), ∇J_t abbreviates the derivative of J at α_t, and $\eta_t > 0$ is a step size parameter. The following Lemma guarantees the differentiable properties of the optimal value function (Bonnans and Shapiro, 1996; Ying et al., 2010):

Lemma 3 *Let $\mathcal{X}$ be a metric space and $\mathcal{U}$ be a normed space. Suppose that for all $x \in \mathcal{X}$ the function $f(x,\cdot)$ is differentiable and that $f(x,u)$ and $\nabla_u f(x,u)$ are continuous on $\mathcal{X} \times \mathcal{U}$, and Q be a compact subset of $\mathcal{X}$. Then the optimal value function $f(u) := \inf_{x \in Q} f(x,u)$ is differentiable. When the minimizer $x(u)$ of $f(\cdot,u)$ is unique, the gradient is given by $\nabla f(u) = \nabla_u f(u, x(u))$.*

From Proposition 1, we see that the minimizer $\mathbf{K}(\alpha)$ is unique for some fixed α. Together with the above lemma, we compute the gradient at the point α by:

$$\nabla J_{ij} = 1 - \mathrm{tr}\, \mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}, \tag{25}$$

where $\mathbf{K} = \left(\frac{B}{\mathrm{tr}\, \mathbf{A}_+^{\frac{p}{p-1}}} \right)^{\frac{1}{p}} \mathbf{A}_+^{\frac{1}{p-1}}$, $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^t \mathbf{T}_{ij} - \mathbf{L}$.

Similarly, for the another formulation:

$$\begin{aligned} \min_{\mathbf{K},\varepsilon_{ij}} \quad & \mathrm{tr}\, \mathbf{L}\mathbf{K} + \frac{C}{2} \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \varepsilon_{ij}^2 + \frac{G}{p} \mathrm{tr}\, \mathbf{K}^p \\ \text{s.t.} \quad & \forall (i,j) \in (\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} \geq 1 - \varepsilon_{ij}, \end{aligned} \tag{26}$$

we can derive the corresponding saddle-point minimax problem of (26):

$$\begin{aligned} \max_{\alpha} \min_{\mathbf{K}} \quad & \mathrm{tr} \left(\left(\mathbf{L} - \sum_{(i,j)} \alpha_{ij} \mathbf{T}_{ij} \right) \mathbf{K} \right) - \frac{1}{2C} \sum_{(i,j)} \alpha_{ij}^2 + \sum_{(i,j)} \alpha_{ij} + \frac{G}{p} \mathrm{tr}\, \mathbf{K}^p \\ \text{s.t.} \quad & \mathbf{K} \succeq \mathbf{0},\ \forall (i,j) \in \mathcal{S}\cup\mathcal{D},\ \alpha_{ij} \geq 0. \end{aligned}$$

Again, from the Proposition 2, we observe that the minimizer $\mathbf{K}(\alpha)$ is unique for some fixed α. Together with Lemma 3, we compute the gradient at the point α^t in the same way as in (25) by setting $\mathbf{K} = \left(\frac{1}{G}\mathbf{A}_+\right)^{\frac{1}{p-1}}$, $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^t \mathbf{T}_{ij} - \mathbf{L}$. The alternative optimization algorithm is summarized in Algorithm 1.

4.3.3 ESTIMATING THE RANK OF K

According to Proposition 1 or Proposition 2, we are required to locate the positive spectrums of $\mathbf{A}$, which can be achieved by full eigen-decomposition of $\mathbf{A}$. However, this can be computationally

Algorithm 1 SimpleNPKL with (square) hinge loss.

```
Input: Pairwise constraint matrix $\mathbf{T}$, parameters $C$ and $B$ (or G), $k$
Output: $\alpha$ and $\mathbf{K}$.
1: Construct graph Laplacian $\mathbf{L}$ using $k$ nearest neighbors;
2: Initialize $\alpha^0$;
3: repeat
4:     Set $\mathbf{A} = \sum_{(i,j)} \alpha_{ij}^{t-1}\mathbf{T}_{ij} - \mathbf{L}$;
5:     Compute the closed-form solution $\mathbf{K}_t = \left(B/\mathrm{tr}\,\mathbf{A}_+^{p/(p-1)}\right)^{1/p}\mathbf{A}_+^{1/(p-1)}$
       //For the formulation (19), use $\mathbf{K}_t = \left(\mathbf{A}_+/G\right)^{1/(p-1)}$ instead;
6:     Compute the gradient $\nabla J_{ij} = 1 - \mathrm{tr}\,\mathbf{T}_{ij}\mathbf{K}_t - \frac{1}{C}\alpha_{ij}$;
7:     Determine a step size $\eta_t$, update $\alpha_{ij}^t$ using $\alpha_{ij}^t = \left(\alpha_{ij}^{t-1} + \eta_t \nabla J_{ij}\right)_+$;
8: until convergence
```

prohibitive for large scale data sets. Moreover, the computation on the negative eigen-vectors of $\mathbf{A}$ should be avoided. The following proposition (Pataki, 1995) bounds the rank of matrix $\mathbf{K}$ in a general SDP setting.

Proposition 4 *The rank r of $\mathbf{K}$ in the SDP problem:* $\max_{\mathbf{K}\succeq\mathbf{0}}$ *tr* $(\mathbf{A}_0\mathbf{K})$ *with m linear constraints on K, follows the bound* $\begin{pmatrix} r+1 \\ 2 \end{pmatrix} \leq m$.

Moreover, from the empirical study in Alizadeh et al. (1997), the rank r is usually much smaller than this bound. This implies that the full decomposition of matrix $\mathbf{A}_0$ is not required. Our formulation (21) has an additional constraint: $\mathrm{tr}\,\mathbf{K}^2 \leq B$ for $p=2$. This condition equivalently constraints $\mathrm{tr}\,(\mathbf{K})$, which is a common assumption in SDP problems (Krishnan and Mitchell, 2006). To show this, we have $B \geq \mathrm{tr}\,\mathbf{KK} = \frac{1}{N}\sum_i \lambda_i^2 N \geq \frac{1}{N}(\sum_i \lambda_i \cdot 1)^2 = \frac{1}{N}(\mathrm{tr}\,\mathbf{K})^2$, where the second inequality is resulted from the Cauchy inequality. Hence, we have $\mathrm{tr}\,\mathbf{K} \leq \sqrt{BN}$. Therefore, we can make use of the r estimated from Proposition 4 as a suggestion to estimate the rank of $\mathbf{K}$.

4.3.4 DETERMINING THE CONVERGENCE PROPERTIES

When the η_t is small enough or a universal choice of $\eta_t = O(1/t)$ is used, the whole optimization problem is guaranteed to converge (Boyd and Xiao, 2005). Practically, the value of η plays an important role for the convergence speed. Therefore, it is worth studying the influence of η on the convergence rate, which requires to lower bound the increment of J_{α_t} at each step. We first establish the Lipschitz property of $\nabla J(\alpha)$.

Lemma 5 *Assume we use the formulation of Proposition 2 at each iteration of Algorithm 1, then the gradient of the objective function given by (25) is Lipschitz continuous with Lipschitz constant $L = \frac{m}{G} + \frac{1}{C}$, where $m = |\mathcal{S}\cup\mathcal{D}|$ is the number of nonzeros in $\mathbf{T}$. That is,*

$$\|\nabla J(\alpha_1) - \nabla J(\alpha_2)\|_F \leq \left(\frac{m}{G} + \frac{1}{C}\right)\|\alpha_1 - \alpha_2\|_F.$$

Proof For an α_t, we use $\mathbf{K}_t$ denote the corresponding minimizer of J computed by (14). For a spectral function λ defined on $\mathbb{S}_+$, which is Lipschitz continuous with Lipschitz constant κ, we have

$$\|\lambda(\mathbf{K}_1) - \lambda(\mathbf{K}_2)\|_F \leq \kappa\|\mathbf{K}_1 - \mathbf{K}_2\|_F.$$

For our case, the p.s.d. projection is defined by $\lambda(\mathbf{K}) = \sum_i \max(0, \lambda_i)^2$. The Lipschitz constant κ of this function is 1. Therefore, for any $\mathbf{K}_1$ and $\mathbf{K}_2$ given by (14), we have

$$\begin{aligned}
\|\mathbf{K}_1 - \mathbf{K}_2\|_F &= \|\mathbf{A}_{1+} - \mathbf{A}_{2+}\|_F \\
&\leq \Big\| \frac{1}{G}\Big(\sum_{(i,j)} \alpha_{ij}^{(1)} \mathbf{T}_{ij} - \mathbf{L}\Big) - \frac{1}{G}\Big(\sum_{(i,j)} \alpha_{ij}^{(2)} \mathbf{T}_{ij} - \mathbf{L}\Big) \Big\|_F \\
&= \frac{1}{G} \Big\| \sum_{(i,j)} (\alpha_{ij}^{(1)} - \alpha_{ij}^{(2)}) \mathbf{T}_{ij} \Big\|_F \\
&\leq \frac{1}{G} \|\alpha_1 - \alpha_2\|_F \|\mathbf{T}\|_F = \frac{\sqrt{m}}{G} \|\alpha_1 - \alpha_2\|_F.
\end{aligned}$$

Consequently, we have,

$$\begin{aligned}
\|\nabla J(\alpha_1) - \nabla J(\alpha_2)\|_F &= \sqrt{\sum_{(i,j)} \Big(\big(1 - \operatorname{tr} \mathbf{T}_{ij}\mathbf{K}_1 - \frac{1}{C}\alpha_{ij}^{(1)}\big) - \big(1 - \operatorname{tr} \mathbf{T}_{ij}\mathbf{K}_2 - \frac{1}{C}\alpha_{ij}^{(2)}\big) \Big)^2} \\
&= \sqrt{\sum_{(i,j)} \Big(\operatorname{tr} \mathbf{T}_{ij}\big(\mathbf{K}_2 - \mathbf{K}_1\big) + \frac{1}{C}\big(\alpha_{ij}^{(2)} - \alpha_{ij}^{(1)}\big) \Big)^2} \\
&\leq \|\mathbf{T}\|_F \|\mathbf{K}_1 - \mathbf{K}_2\|_F + \frac{1}{C}\|\alpha_1 - \alpha_2\|_F \\
&\leq \big(\frac{m}{G} + \frac{1}{C}\big) \|\alpha_1 - \alpha_2\|_F.
\end{aligned}$$

■

With the Lipschitz property of ∇J, we can further show each iteration of Algorithm 1 makes progress towards the optimal solution. Interestingly, we are aware that the proof is very similar to the analysis of indefinite kernel learning, which is proposed very recently by Ying et al. (2010). This result is developed based on non-smooth optimization algorithm of Nesterov (2005). To make the paper complete, we expose the detailed proof in the following proposition.

Proposition 6 *Assume we use the formulation of Proposition 2, and* $\eta \geq \frac{m}{G} + \frac{1}{C}$ *at each iteration of Algorithm 1. The iteration sequence* $\{\alpha_t\}$ *generated in Algorithm 1 satisfy:*

$$J(\alpha_{t+1}) \geq J(\alpha_t) + \frac{\eta}{2}\|\alpha_{t+1} - \alpha_t\|_F^2,$$

and

$$\max_\alpha J(\alpha) - J(\alpha_t) \leq \frac{\eta}{2t}\|\alpha_0 - \alpha^*\|_F^2,$$

where α^* *is the optimal solution of* $\max_\alpha J(\alpha)$.

Proof Let $L = \frac{m}{G} + \frac{1}{C}$ abbreviate the Lipschitz constant of $\nabla J(\alpha)$, then we have

$$\begin{aligned}
J(\alpha) - J(\alpha_t) - \langle \nabla J(\alpha_t), \alpha - \alpha_t \rangle &= \int_{\alpha_t}^{\alpha} \nabla J(\alpha) d\alpha - \langle \nabla J(\alpha_t), \alpha - \alpha_t \rangle \\
&= \int_0^1 \langle \nabla J(\theta\alpha + (1-\theta)\alpha_t) - \nabla J(\alpha_t), \alpha - \alpha_t \rangle d\theta \\
&\geq -\int_0^1 \|\nabla J(\theta\alpha + (1-\theta)\alpha_t) - \nabla J(\alpha_t)\| \|\alpha - \alpha_t\|_F d\theta \\
&\geq -L \int_0^1 \theta \|\alpha - \alpha_t\|_F^2 d\theta \\
&\geq -\frac{\eta}{2} \|\alpha - \alpha_t\|_F^2.
\end{aligned}$$

Applying this inequality with $\alpha = \alpha_{t+1}$, we have

$$-J(\alpha_t) - \langle \nabla J(\alpha_t), \alpha_{t+1} - \alpha_t \rangle \geq -J(\alpha_{t+1}) - \frac{\eta}{2} \|\alpha_{t+1} - \alpha_t\|_F^2. \tag{27}$$

From step 5 in Algorithm 1, it is easy to verify that

$$\begin{aligned}
\alpha_{t+1} &= \arg\min_{\alpha} \|(\alpha - \alpha_t) - \nabla J(\alpha_t)/\eta\|_F^2 \\
&= \arg\min_{\alpha} -2\langle \alpha - \alpha_t, \nabla J(\alpha_t)/\eta \rangle + \|\alpha - \alpha_t\|_F^2 \\
&= \arg\min_{\alpha} -\nabla J(\alpha_t) - \langle \alpha - \alpha_t, \nabla J(\alpha_t) \rangle + \frac{\eta}{2} \|\alpha - \alpha_t\|_F^2. \qquad (28)
\end{aligned}$$

Let $f(\alpha)$ denote the right side of (28). From the first-order optimality condition over α_{t+1}, for any α we have $\langle \nabla f(\alpha_{t+1}), \alpha - \alpha_{t+1} \rangle \geq 0$, that is,

$$-\langle \nabla J(\alpha_t), \alpha - \alpha_{t+1} \rangle \geq \eta \langle \alpha_{t+1} - \alpha_t, \alpha_{t+1} - \alpha \rangle. \tag{29}$$

Adding (27) and (29) together yields that $-J(\alpha_t) - \langle \nabla J(\alpha_t), \alpha - \alpha_t \rangle \geq -J(\alpha_{t+1}) + \eta \langle \alpha_t - \alpha_{t+1}, \alpha - \alpha_t \rangle + \frac{\eta}{2} \|\alpha_t - \alpha_{t+1}\|_F^2$. Note that $-J$ is convex, $-J(\alpha) \geq -J(\alpha_t) - \langle \nabla J(\alpha_t), \alpha - \alpha_t \rangle$. Thus we have

$$J(\alpha_{t+1}) \geq J(\alpha) + \eta \langle \alpha_t - \alpha_{t+1}, \alpha - \alpha_t \rangle + \frac{\eta}{2} \|\alpha_t - \alpha_{t+1}\|_F^2.$$

Applying $\alpha = \alpha_t$, we have that

$$J(\alpha_{t+1}) \geq J(\alpha_t) + \frac{\eta}{2} \|\alpha_{t+1} - \alpha_t\|_F^2.$$

Applying $\alpha = \alpha^*$, we have that

$$J(\alpha^*) - J(\alpha_{i+1}) \leq -\eta \langle \alpha_i - \alpha_{i+1}, \alpha^* - \alpha_i \rangle - \frac{\eta}{2} \|\alpha_i - \alpha_{i+1}\|_F^2 = \frac{\eta}{2} \|\alpha^* - \alpha_i\|_F^2 - \frac{\eta}{2} \|\alpha^* - \alpha_{i+1}\|_F^2. \tag{30}$$

Taking summation over i from 0 to $t-1$, we have

$$\sum_{i=0}^{t-1} (J(\alpha^*) - J(\alpha_{i+1})) \leq \frac{\eta}{2} \|\alpha^* - \alpha_0\|_F^2.$$

From (30), we see that the sequence $\{J(\alpha_t)\}$ increase monotonically. Thus we obtain

$$t(J(\alpha^*) - J(\alpha_t)) \leq \frac{\eta}{2}\|\alpha^* - \alpha_0\|_F^2,$$

which completes the proof. ■

4.4 SimpleNPKL with Square Loss

In this subsection, we consider square alignment loss for the SimpleNPKL framework:

$$\begin{aligned} \min_{\mathbf{K},\varepsilon_{ij}} \quad & \operatorname{tr} \mathbf{LK} + \frac{C}{2} \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \varepsilon_{ij}^2 \\ \text{s.t.} \quad & \forall (i,j) \in (\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} = 1 - \varepsilon_{ij}, \\ & \mathbf{K} \succeq \mathbf{0}, \operatorname{tr} \mathbf{K}^p \leq B. \end{aligned}$$

Here we need not to enforce $\varepsilon \geq 0$. With the standard techniques of Section 4.3, we derive the following min-max problem:

$$\max_{\alpha} \min_{\mathbf{K}} \operatorname{tr}\left(\mathbf{L} - \sum_{ij} \alpha_{ij}\mathbf{T}_{ij}\right)\mathbf{K} + \sum_{ij}\alpha_{ij} - \frac{1}{2C}\sum_{ij}\alpha_{ij}^2 \ :\ \mathbf{K}\succeq \mathbf{0}, \operatorname{tr}\mathbf{K}^p \leq B.$$

Therefore, we can compute the gradient of J w.r.t. α:

$$\nabla J_{ij} = 1 - \operatorname{tr}\mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}.$$

The whole analysis of Section 4.3 still holds. The difference just lies in the way of computing gradient ∇J. We will show an application of square loss in Section 6.

4.5 SimpleNPKL with Hinge Loss

In this subsection, we consider hinge loss for the SimpleNPKL framework:

$$\begin{aligned} \min_{K,\varepsilon_{ij}} \quad & \operatorname{tr} \mathbf{LK} + C \sum_{(i,j)\in(\mathcal{S}\cup\mathcal{D})} \varepsilon_{ij} \\ \text{s.t.} \quad & \forall (i,j) \in (\mathcal{S}\cup\mathcal{D}),\ T_{ij}K_{ij} \geq 1 - \varepsilon_{ij}, \varepsilon_{ij} \geq 0 \\ & \mathbf{K} \succeq \mathbf{0}, \operatorname{tr} \mathbf{K}^p \leq B. \end{aligned}$$

Following the standard techniques of Lagrangian dual, we arrive at the min-max problem:

$$\max_{\alpha} \min_{\mathbf{K}} \operatorname{tr}\left(\mathbf{L} - \sum_{ij} \alpha_{ij}\mathbf{T}_{ij}\right)\mathbf{K} + \sum_{ij}\alpha_{ij} \ :\ \mathbf{K}\succeq \mathbf{0},\ \operatorname{tr}\mathbf{K}^p \leq B,\ 0 \leq \alpha_{ij} \leq C.$$

Therefore, we can compute the gradient of J w.r.t. α:

$$\nabla J_{ij} = 1 - \operatorname{tr}\mathbf{T}_{ij}\mathbf{K}$$

The whole analysis of Section 4.3 still holds. The difference just lies in the way of computing gradient ∇J. Note that the gradient updating $\alpha = \alpha + \eta\nabla J$ may jump out of the range $[0, C]$. We need to project α into this region at each iteration. We will also show an example of Hinge loss in Section 6.

5. Implementation Issues

In this Section, we discuss some implementation issues that are important to the success of the proposed SimpleNPKL algorithms.

5.1 Building a Sparse Graph Laplacian

Recall that the graph Laplacian $\mathbf{L}$ in (9) is often sparse, in particular, which is usually computed by finding k-nearest neighbors for the purpose of constructing the similarity matrix $\mathbf{S}$. Specifically, an entry $S(i,j) = 1$ if and only if data examples i and j are among each other's k-nearest neighbors; otherwise, it is set to 0. So, there are at most k nonzero entries on each row of $\mathbf{L}$.

A naïve implementation of finding k-nearest neighbors often takes $O(N^2 \log N)$ time. To enforce the data examples i and j are among each other's k-nearest neighbors, one can use B-matching algorithm (Jebara and Shchogolev, 2006) to find the k-nearest neighbors. However, when the data set is very large, the construction of $\mathbf{L}$ becomes non-trivial and very expensive. To address this challenge, we suggest to first construct the *cover tree* structure (Beygelzimer et al., 2006), which takes $O(N \log N)$ time. The similar idea to construct a tree structure for distance metric learning was discussed in Weinberger and Saul (2008). With the aid of this data structure, the batch query of finding k-nearest neighbors on the whole data set can be done within $O(N)$ time. Hence, the graph Laplacian $\mathbf{L}$ can be constructed efficiently for large-scale problems.

5.2 Fast Eigendecomposition by Lanczos Algorithm

Among various existing SDP approaches (Boyd and Vandenberghe, 2004), the interior-point method is often deemed as the most efficient one. However, as discussed in previous subsection, the graph Laplacian $\mathbf{L}$ is often sparse. In addition, the number of pairwise constraints is usually small due to expensive cost of human labels. Therefore, $\mathbf{L} - \sum_{(i,j)} \alpha_{ij} \mathbf{T}_{ij}$ is also sparse. Such sparse structure is not yet exploited in such general algorithms. According to Proposition 1, the time cost of each iteration in Algorithm 1 is dominated by eigen-decomposition. Moreover, from Proposition 4, the rank r of the kernel matrix $\mathbf{K}$ is upper bounded by the number of active constraints. Therefore, we can estimate the rank for sparse eigen-decomposition, which can be solved efficiently using the so-called *Implicitly Restarted Lanczos Algorithm* (IRLA) (Lehoucq et al., 1998). Its computational cost is dominated by matrix-vector multiplication. Specifically, the time cost of IRLA is linear with the number of non-zeros in $\mathbf{A}$. Assume k nearest neighbors are used to construct the graph Laplacian $\mathbf{L}$, then the number of non-zeros in $\mathbf{A}$ is at most $Nk + m$, where m is the number of nonzeros in $\mathbf{T}$, and $\mathbf{A}$ is very sparse. Moreover, the time cost of computing gradient is $O(m)$. Therefore, the time complexity per iteration of SimpleNPKL is $O(Nk + m)$.

5.3 Active Constraint Selection

As shown in Algorithm 1, the computational cost of the update procedure is highly depends on the number of pairwise constraints. However, some less informative constraints often do not contribute much to the learning of the kernel matrix $\mathbf{K}$, and fitting some noisy pairwise constraints may also lead to the poor generalization. Moreover, as discussed in Section 4.3.3, the rank of $\mathbf{K}$ is lower when there are fewer active constraints in (22). Therefore, selecting pairwise constraints for SimpleNPKL may improve both the efficiency and the generalization of the NPK learning.

To speed up the eigen-decomposition process, instead of engaging all pairwise constraints, we propose to sample a subset of T_{ij}'s for SimpleNPKL. Instead of acquiring class label information for kernel learning; here, we consider another simple active constraint selection scheme. Recall that a general principle in active learning is to request the label of the data points that are most uncertain for their predictions. Following this idea, we adopt the margin criterion to measure the uncertainty of the prediction value on a data point. In particular, given a data point $\mathbf{x}_i$, assume that we have the prediction function in the form:

$$f(\mathbf{x}_i) = \sum_j y_j K(\mathbf{x}_i, \mathbf{x}_j).$$

We can use $|y_i f(\mathbf{x}_i)|$ to measure the uncertainty of prediction, where $y_i \in \{-1, +1\}$ is the class label of data point $\mathbf{x}_i$. As a result, for a data point $\mathbf{x}_i$, we choose the constraints involving point i:

$$\begin{aligned} i^* &= \arg\min_i \left| \frac{1}{l_i} \sum_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right| \\ &= \arg\min_i \left| \frac{1}{l_i} \sum_{j, T_{ij} \neq 0} T_{ij} K(\mathbf{x}_i, \mathbf{x}_j) \right|, \end{aligned}$$

where we deem T_{ij} as an entry of yy', and $l_i = |\{j : (i, j) \in \mathcal{S} \cup \mathcal{D}\}, T_{ij} \neq 0\}|$ is used as a normalization of the margin value. Based on the above formula, we choose a subset of k data points $\mathcal{S}_k$ that are most uncertain according to the margin measure. Then, we choose all the T_{ij}'s that involve any point $i \in \mathcal{S}_k$ as pairwise constraints to form a new set of constraints. Finally, we run SimpleNPKL based on this new set of constraints.

5.4 Low Rank Approximation of K

Since the rank r of $\mathbf{K}$ often satisfies $r < n$, we may express $\mathbf{K}$ as $\mathbf{K} = \mathbf{V}\mathbf{E}\mathbf{V}'$, where the columns of $\mathbf{V}_{n\times r}$ are eigenvectors of $\mathbf{K}$. If we fix the base $\mathbf{V}$, the number of variables is reduced from n^2 to r^2. With this approximation scheme, the $\mathbf{A}$ matrix in Algorithm 1 becomes $\mathbf{A} = \mathbf{V}'(\mathbf{L} - \sum \alpha_{ij}\mathbf{T}_{ij})\mathbf{V}$. Note $\mathbf{V}'\mathbf{L}\mathbf{V}$ can be pre-computed and $\mathbf{V}' \sum \alpha_{ij}\mathbf{T}_{ij}\mathbf{V}$ can be computed efficiently by virtue of the sparseness. Therefore, SimpleNPKL can be significantly faster with this approximation.

6. Applications of SimpleNPKL

In this Section, we extend the proposed SimpleNPKL technique to other similar machine learning problems where the goal of the optimization is to find an optimal matrix such that its inner product with another matrix is maximized or minimized. In particular, we consider the data embedding problems, where the goal is to find a new data representation that preserves some similarity/distance constraints between pairs of data points. These problems typically can be implemented by constraining the alignment of the target kernel matrix to some prior affinity or distance structures. As a result, the kernel matrix $\mathbf{K} = \mathbf{V}'\mathbf{V}$ implies a data embedding with a natural interpretation, in which the column vector of $\mathbf{V}$ corresponds to the new data representation. We discuss several important data embedding methods below.

6.1 Colored Maximum Variance Unfolding

Colored MVU (Song et al., 2008) is an improvement of Maximum Variance Unfolding (MVU) (Weinberger et al., 2004), which produces a low-dimensional representation of the data by maximizing the trace of a matrix $\mathbf{K}$ subject to some positive definiteness, centering and distance-preserving constraints, that is:

$$\min_{\mathbf{K}} \quad -\mathrm{tr}\,\mathbf{K} \;\; : \;\; \mathbf{K} \succeq \mathbf{0}, \sum_{ij} \mathbf{K}_{ij} = 0,\ \mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = D_{ij},\ \forall (i,j) \in \mathcal{N}.$$

where $\mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = K_{ii} + K_{jj} - 2K_{ij}$ is the square distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

CMVU interprets MVU from a statistical perspective. It maximizes the dependence between the domain of input pattern $\mathbf{x}$ and the domain of label y, which is measured by the *Hilbert- Schmidt Independence Criterion* (Gretton et al., 2005; Song et al., 2008). Here we introduce slack variables ξ to measure the violations of distance constraints and penalize the corresponding square loss. Consequently the optimization task of colored MVU is reformulated as:

$$\min_{\mathbf{K},\xi} \quad -\mathrm{tr}\,\mathbf{HKHY} + \frac{C}{2}\sum \xi_{ij}^2, \;\; : \;\; \mathbf{K} \succeq \mathbf{0},\ \mathrm{tr}\,\mathbf{K}\mathbf{T}_{ij} = D_{ij} - \xi_{ij},\ \forall (i,j) \in \mathcal{N}$$

where $H_{ij} = \delta_{ij} - N^{-1}$ such that $\mathbf{HKH}$ centers $\mathbf{K}$, $\mathbf{Y} = \mathbf{y}\mathbf{y}'$ is the kernel matrix over labels. Apparently this belongs to an SDP problem.

Following the SimpleNPKL algorithms, we derive the minimax optimization problem by introducing dual variables for the inequality constraints:

$$\max_{\alpha} \min_{\mathbf{K}} \quad \mathrm{tr}\left(-\mathbf{HYH} - \sum_{ij} \alpha_{ij} \mathbf{T}_{ij} \right) \mathbf{K} + \sum_{ij} \alpha_{ij} D_{ij} - \frac{1}{2C} \sum_{ij} \alpha_{ij}^2 \;\; : \;\; \mathbf{K} \succeq \mathbf{0},\ \mathrm{tr}\,\mathbf{KK} \leq B. \tag{31}$$

By substituting the following results

$$\mathbf{A} = \mathbf{HYH} + \sum_{ij} \alpha_{ij} \mathbf{T}_{ij} \;\text{ and }\; \nabla J_{ij}^t = D_{ij} - \mathrm{tr}\,\mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha_{ij}^t$$

back into Algorithm 1, the problem of (31) can be solved immediately.

6.2 Minimum Volume Embedding

Minimum Volume Embedding (MVE) is another improvement of MVU (Shaw and Jebara, 2007). One limitation of MVU is that it simply maximizes the trace of $\mathbf{K}$, which may result in the solution that engages considerably more dimensions than necessity. To address this problem, Shaw and Jebara (2007) proposed to grow the top few eigenvalues of $\mathbf{K}$ while shrinking the remaining ones. In particular, let $\mathbf{K} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i'$, $\lambda_1 \geq, \ldots, \geq \lambda_n$, and $\mathbf{K}_0 = \sum_{i=1}^d \mathbf{v}_i \mathbf{v}_i' - \sum_{i=d+1}^n \mathbf{v}_i \mathbf{v}_i'$. When the intrinsic dimensionality d is available, MVE formulates the data embedding problem as follows:

$$\min_{\mathbf{K}} \quad -\mathrm{tr}\,\mathbf{K}\mathbf{K}_0 \;\; : \;\; \text{the same set of constraints of MVU.} \tag{32}$$

After obtaining the solution $\mathbf{K}^t$ at each step, MVE proceeds by substituting $\mathbf{K}_0 = \mathbf{K}^t$ back to the optimization of (32) and repeatedly solving the optimization. Hence, MVE improves MVU by decreasing the energy of the small eigen components of $\mathbf{K}$. To find the solution, every $\mathbf{K}^t$ is computed by applying a general SDP solver in Shaw and Jebara (2007).

To speed up the solution, following the similar derivation in the above CMVU, we can solve (32) by eigen-decomposition in an iterative manner. Specifically, we make the following modifications:

$$A = \mathbf{K}_0 + \sum_{ij} \alpha_{ij}\mathbf{T}_{ij} \text{ and } \nabla J^t_{ij} = D_{ij} - \operatorname{tr}\mathbf{T}_{ij}\mathbf{K} - \frac{1}{C}\alpha^t_{ij}$$

By substitute the above results back into Algorithm 1, we can solve the MVE problem efficiently.

6.3 Structure Preserving Embedding

Structure Preserving Embedding (SPE) (Shaw and Jebara, 2009) is a machine learning technique that embeds graphs in low-dimensional Euclidean space such that the embedding preserves the global topological properties of the input graph. Suppose we have a connectivity matrix $\mathbf{W}$, where $W_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected and $W_{ij} = 0$ otherwise. SPE learns a kernel matrix $\mathbf{K}$ such that the similarity $\operatorname{tr}\mathbf{KW}$ is maximized while the global topological properties of the input graph are preserved. More formally, the SPE problem is formulated into the following SDP optimization:

$$\min_{\mathbf{K}} \quad -\operatorname{tr}\mathbf{KW} + C\xi \ : \ D_{ij} > (1 - W_{ij})\max_m(W_{im}D_{im}) - \xi, \ \xi \geq 0$$

where $D_{ij} = K_{ii} + K_{jj} - 2K_{ij} = \operatorname{tr}\mathbf{KT}_{ij}$ is the squared distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Let $[n] = \{1, \ldots, n\}$ and $\mathcal{N}_i$ denote the set of indices of points which are among the nearest neighbors of $\mathbf{x}_i$. Then for each point $\mathbf{x}_i$, SPE essentially generates $(n - |\mathcal{N}_i|) \times |\mathcal{N}_i|$ constraints:

$$\operatorname{tr}\mathbf{KT}_{ij} > \operatorname{tr}\mathbf{KT}_{ik} - \xi, \ \forall i \in [n], j \in [n] - \mathcal{N}_i, k \in \mathcal{N}_i.$$

In order to speed up the SPE algorithm, we apply the SimpleNPKL technique to turn the SPE optimization into the following minimax optimization problem:

$$\max_\alpha \min_{\mathbf{K}} \quad \operatorname{tr}\left(\sum_i \sum_{k\in\mathcal{N}_i} \sum_{j\notin\mathcal{N}_i} \alpha_{ijk}(\mathbf{T}_{ik} - \mathbf{T}_{ij}) - \mathbf{W}\right)\mathbf{K} \ : \ \mathbf{K} \succeq \mathbf{0}, \operatorname{tr}\mathbf{KK} \leq B, \sum \alpha_{ijk} \in [0, C].$$

Similarly, we can derive the following results:

$$\mathbf{A} = \mathbf{W} - \sum_{ijk} \alpha_{ijk}(\mathbf{T}_{ik} - \mathbf{T}_{ij}) \text{ and } \nabla J^t_{ijk} = \operatorname{tr}\mathbf{K}(\mathbf{T}_{ik} - \mathbf{T}_{ij}).$$

Substituting them back into Algorithm 1 leads to an efficient solution for the SPE problem.

7. Experiments

In this Section, we conduct extensive experiments to examine the efficacy and efficiency of the proposed SimpleNPKL algorithms.

7.1 Experimental Setup

We examine both efficacy and efficiency of the proposed SimpleNPKL using side information to learn a kernel matrix for kernel k-means clustering. As shown in Hoi et al. (2007), the learned kernel matrix of the Non-Parametric Kernel Learning (NPKL) outperforms other kernel learning methods in the task of clustering using side information. For simplicity, we only compare our proposed SimpleNPKL algorithms with the NPKL method in Hoi et al. (2007) for kernel k-means clustering. The results of k-means clustering and constrained k-means clustering using Euclidean metric are also reported as the performance of the baseline methods. The abbreviations of different approaches are described as follows:

- ***k*-means**: k-means clustering using Euclidean metric;
- **c*k*-means**: The constrained k-means clustering algorithm using Euclidean metric and side information;
- **SimpleNPKL+LL**: The proposed SimpleNPKL with linear loss defined in (12);
- **SimpleNPKL+SHL**: The proposed SimpleNPKL with squared hinge loss defined in (21);
- **NPKL+LL**: NPKL in (10) using linear loss;
- **NPKL+HL**: NPKL in (10) using hinge loss.

To construct the graph Laplacian matrix $\mathbf{L}$ in NPKL, we adopt the cover tree data structure.[2] The sparse eigen-decomposition used in SimpleNPKL is implemented by the popular *Arpack* toolkit.[3] We also adopt the standard SDP solver, SDPT3,[4] as the baseline solution for NPKL. The pair-wise constraint is assigned for randomly generated pairs of points according to their ground truth labels. The number of constraints is controlled by the resulted amount of connected components as defined in previous studies (Xing et al., 2003; Hoi et al., 2007). Note that typically the larger the number of constraints, the smaller the number of connected components.

Several parameters are involved in both NPKL and SimpleNPKL. Their notation and settings are given as follows:

- k : The number of nearest neighbors for constructing the graph Laplacian matrix $\mathbf{L}$, we set it to 5 for small data sets in Table 1, and 50 for Adult database in Table 6;
- r : The ratio of the number of connected components compared with the data set size N. In our experiments, we set $r \approx 70\%N$ which follows the setting of Hoi et al. (2007);
- B : The parameter that controls the capacity of the learned kernel in (11). We fix $B = N$ for the adult data sets and fix $B = 1$ for the data sets in Table 1 and;
- C : The regularization parameter for the loss term in NPKL and SimpleNPKL. We fix $C = 1$ for the adult data sets and several constant values in the range $(0, 1]$ for the data sets in Table 1.

In our experiments, all clustering results were obtained by averaging the results from 20 different random repetitions. All experiments were conducted on a 32bit Windows PC with 3.4GHz CPU and 3GB RAM.

7.2 Comparisons on Benchmark Data Sets

To evaluate the clustering performance, we adopt the clustering accuracy used in Hoi et al. (2007):

$$\text{Cluster Accuracy} = \sum_{i>j} \frac{\mathbf{1}\{c_i = c_j\} = \mathbf{1}\{\hat{c}_i = \hat{c}_j\}}{0.5n(n-1)}.$$

2. The cover tree data structure is described at `http://hunch.net/~jl/projects/cover_tree/cover_tree.html`.
3. The *Arpack* toolkit can be found at `http://www.caam.rice.edu/software/ARPACK/`.
4. SDPT3 can be found at `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`.

Data Set	#Classes	#Instances	#Features
Chessboard	2	100	2
Glass	6	214	9
Heart	2	270	13
Iris	3	150	4
Protein	6	116	20
Sonar	2	208	60
Soybean	4	47	35
Spiral	2	100	3
Wine	3	178	12

Table 1: The statistics of the data sets used in our experiments.

This metric measures the percentage of data pairs that are correctly clustered together. We compare the proposed SimpleNPKL algorithms with NPKL on the nine data sets from UCI machine learning repositories,[5] as summarized in Table 1. The same data sets were also adopted in the NPKL study of Hoi et al. (2007).

The clustering accuracy and CPU time cost (the clustering time was excluded) of different NPKL methods are reported in Table 2 and 3. As can be observed from Table 2, all NPKL methods outperform the baseline k-means clustering and the constrained k-means clustering methods, which use Euclidean metric for k-means clustering. The proposed SimpleNPKL with square hinge loss produces very competitive clustering performance to the results of NPKL with hinge loss (as reported in Hoi et al., 2007). SimpleNPKL with square hinge loss and NPKL with hinge loss often perform better than the NPKL methods using linear loss.

For the CPU time cost, the time costs of SimpleNPKL and NPKL using linear loss are usually lower than those of their counterparts with (square) hinge loss. Regarding the efficiency evaluation in Table 3, our SimpleNPKL with linear loss or squared hinge loss is about 5 to 10 times faster than NPKL using the SDPT3 solver. For some cases of linear loss, SimpleNPKL can be even 100 times faster.

Recall that our key Proposition 1 provides a closed-form solution to the learned kernel matrix $\mathbf{K}$ for $p \geq 1$, in which the capacity parameter B can be omitted for SimpleNPKL+linear loss. To show the influence of the capacity parameter B for SimpleNPKL + square hinge loss, we present some results in Table 4 with a fixed $p = 2$. To clearly show the influence on convergence, we present the number of iterations instead of elapsed CPU time. We observe that SimpleNPKL + square hinge loss is not sensitive to B on the both *Iris* and *Protein* data sets. It even produces the identical accuracy on the *Iris* data set for $B \in \{2.5, 3, 3.5, 4\}$. However, it affects the number of steps it takes to converge. Similar phenomena can be observed on other data sets.

We also study the clustering performance of varying p in Table 5. We fixed $B = 1$ in this experiment. From Table 5, we can observe that SimpleNPKL+square hinge loss produces the best clustering accuracy for the *Iris* data set when $p = 4$, but the improvement is not significant comparing with $p = 2$. For the *Protein* data set, our algorithm achieves the best results when $p = 2$. In general, when $p < 2$, the clustering performance drops significantly.

5. The data sets are available at `http://archive.ics.uci.edu/ml/`.

Data Set	k-means	ck-means	NPKL		SimpleNPKL	
			LL	HL	LL	SHL
Chessboard	49.8 ±0.2	50.1±0.3	**61.1± 6.9**	56.3± 6.1	60.2± 0.0	58.8± 0.8
Glass	69.7 ±1.9	69.2±1.7	74.4± 3.7	**79.1± 4.9**	73.0± 2.5	73.5± 2.9
Heart	51.5 ±0.1	52.3±3.7	86.0± 0.3	86.2± 0.0	86.8± 0.0	**89.4± 0.1**
Iris	84.5 ±6.5	89.4±8.5	96.0± 6.1	**97.4± 0.0**	**97.4± 0.0**	**97.4± 0.0**
Protein	76.2 ±2.0	80.7±3.1	78.2± 3.2	**86.4± 3.8**	81.8± 1.8	75.9± 2.0
Sonar	50.2 ±0.1	50.8±0.2	76.8± 0.3	64.5± 6.8	70.2± 10	**78.0± 0.0**
Soybean	82.1 ±6.1	83.8±8.3	90.2± 7.7	**100.0± 0.0**	95.3± 5.1	95.4± 4.9
Spiral	50.1 ±0.6	50.6±1.3	86.5± 0.0	**94.1± 0.0**	92.2± 0.0	**94.1± 0.0**
Wine	71.2 ±1.2	76.1±2.8	78.1± 1.7	**85.5± 5.3**	83.7± 4.8	85.0± 2.6

Table 2: Clustering accuracy of SimpleNPKL, compared with the results of NPKL in (10) using a standard SDP solver, and k-means.

Data Set	NPKL		SimpleNPKL		Speedup
	LL	HL	LL	SHL	
Chessboard	1.38±0.07	5.23±0.06	**0.05±0.00**	0.13±0.00	27.6
Glass	1.85±0.04	32.36±0.37	**0.11±0.00**	2.95±0.00	16.8
Heart	2.64±0.10	63.84±0.68	**0.17±0.01**	13.15±0.08	15.5
Iris	1.36±0.03	1.65±0.04	**0.04±0.00**	3.45±0.01	34.0
Protein	1.80±0.06	8.16±0.11	**0.05±0.00**	1.32±0.00	36.0
Sonar	1.77±0.08	30.38±0.24	**0.11±0.00**	3.91±0.03	16.1
Soybean	1.51±0.05	3.25±0.04	**0.01±0.00**	0.16±0.00	151.0
Spiral	1.78±0.10	6.23±0.08	**0.05±0.00**	1.95±0.00	36.6
Wine	2.54±0.04	30.91±1.30	**0.09±0.00**	1.53±0.01	28.2

Table 3: CPU time of SimpleNPKL, compared with the results of NPKL in (10) using a standard SDP solver. (The best results are in bold and the last "Speedup" column is listed only for the linear loss case.)

7.3 Scalability Study on Adult Data Set

In this Section, we evaluate our SimpleNPKL algorithms on another larger data set to examine the efficiency and scalability. We adopt the *Adult* database, which is available at the website of LibSVM.[6] The database has a series of partitions: A1a, A2a, $\cdots$, A5a (see Table 6). Since the training time complexity of NPKL using standard SDP solvers is $O(N^{6.5})$, which cannot be applied on this database for comparison. We only report the results of both k-means and constrained k-means clustering as the baseline comparison.

Table 7 shows the clustering performance and CPU time cost (the clustering time was excluded) of SimpleNPKL on the *Adult* database. From the results, we can draw several observations. First of all, we can see that by learning better kernels from pairwise constraints, both SimpleNPKL algorithms produce better clustering performance than that of k-means clustering and constrained

6. LibSVM can be found at `http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/`.

DataSet	B	1	1.5	2	2.5	3	3.5	4
Iris	Accur.(%)	94.8±0.0	94.8±0.0	95.2±0.4	**95.7±0.0**	95.7±0.0	95.7±0.0	95.7±0.0
	#Iterations	11	13	14	**10**	10	31	26
Protein	Accur.(%)	**74.5±0.8**	73.6±1.6	74.4±0.8	74.3±0.9	74.1±1.0	73.7±1.1	73.7±1.0
	#Iterations	51	32	51	**11**	14	27	19

Table 4: Results of varying capacity parameter B with fixed $p = 2$ and $C = 1$ on *Iris* and *protein* data sets.

Data Set	p	1	1.5	2	2.5	3	3.5	4
Iris	Accur.(%)	61.6±3.5	58.6±4.0	94.8±0.0	94.8±0.0	95.1±0.4	94.8±0.0	**95.6±0.2**
	#Iterations	51	**6**	11	9	19	10	9
Protein	Accur.(%)	72.3±1.3	72.8±2.2	**74.5±0.8**	73.6±1.5	73.6±1.6	73.5±1.6	73.5±1.6
	#Iterations	32	35	51	40	**11**	**11**	21

Table 5: Results of varying p in the p-norm regularization over $\mathbf{K}$ with fixed $B = 1$ and $C = 1$ on *Iris* and *protein* data sets.

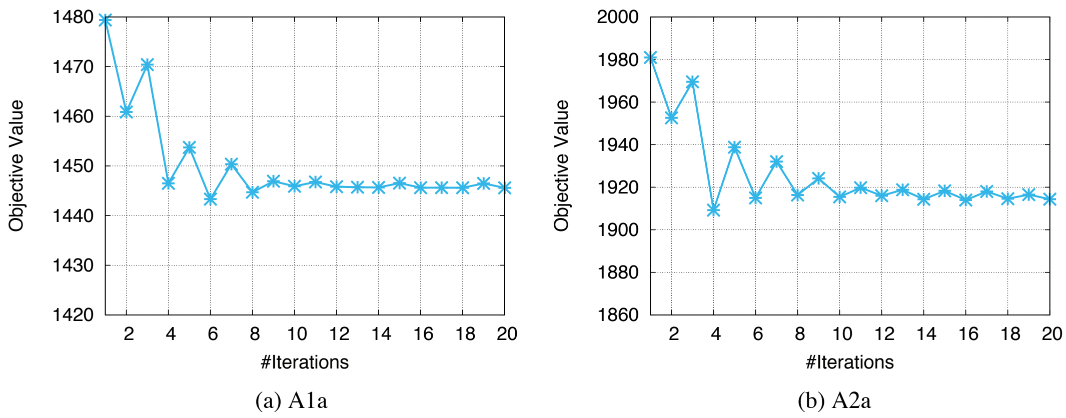


Figure 1: Convergence of SimpleNPKL using square hinge loss on *A1a* and *A2a*. The parameters are $C = 1, B = N$.

k-means clustering methods using Euclidean metric. Further, comparing the two algorithms themselves, in terms of clustering accuracy, they perform quite comparably, in which SimpleNPKL+SHL outperforms slightly. However, in terms of CPU time cost, SimpleNPKL+LL with linear loss is considerably lower than SimpleNPKL+SHL using square hinge loss.

We also plot the objective value $J(K,\alpha)$ of SimpleNPKL on two data sets *A1a* and *A2a* in Figure 1. We observe that SimpleNPKL with square hinge loss often converges quickly within 10 iterations. Similar results can be observed from the other data sets.

Data Set†	A1a	A2a	A3a	A4a	A5a
#Instances	1,605	2,265	3,185	4,781	6,414

†: #Classes=2, #Features=123

Table 6: The statistics of the *Adult* database.

Data Set	#Constraints	Accuracy(%)				CPU Time(s)	
		k-means	ck-means	SimpleNPKL		SimpleNPKL	
				LL	SHL	LL	SHL
A1a	4,104	56.4±3.5	59.0±2.3	**61.4±1.7**	60.7±2.7	**8.5**	322.9
A2a	5,443	57.3±3.6	60.2±0.1	61.1±1.3	**61.4±1.2**	**15.3**	637.2
A3a	7,773	57.8±3.5	59.2±3.0	61.1±1.7	**61.5±2.0**	**28.8**	1,160.8
A4a	12,465	58.8±1.6	59.3±3.9	**61.6±1.3**	61.4±1.5	**66.3**	2,341.3
A5a	16,161	57.7±3.1	59.8±2.2	60.8±3.1	**61.9±1.7**	**79.6**	3,692.1

Table 7: Evaluation results on *Adult* data set. (The best results are in bold.)

7.4 Comparisons on Constraint Selection

In this Section, we study the active constraint selection scheme for SimpleNPKL. Figure 2 shows the clustering performance of active constraint selection by the approach described in Section 5.3.

Several observations can be drawn from the results: 1) Comparing with the original approach using all constraints, the computation time is reduced by choosing a small amount of pairwise constraints. This is because the Lanczos algorithm can perform the sparse eigen-decomposition faster on a sparse matrix with fewer nonzero entries; 2) Though the active constraint selection costs more time than random selection, the former usually achieves better clustering (accuracy) performance than the latter with the same amount of constraints; 3) Using the proposed active constraint selection method to choose about half of the pairwise constraints for SimpleNPKL can often produce comparable or even better clustering performance than that using all constraints.

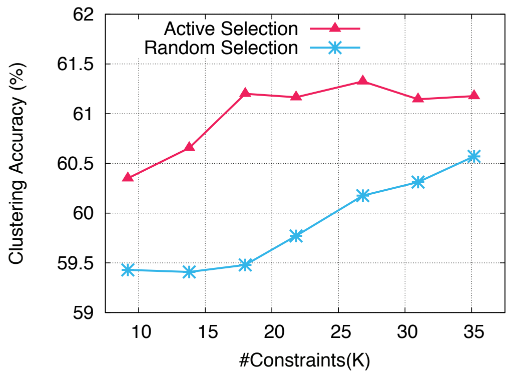
7.5 Evaluations on Data Embedding Applications

In this Section, we evaluate the performance of the proposed SimpleNPKL algorithms with applications to speed up three data embedding techniques, that is, CMVU, MVE, and SPE, respectively. Our goal is to show that SimpleNPKL is capable of producing similar empirical results to the baseline counterpart with significant efficiency gain. All the data sets are publicly available in the UCI machine learning repository. In all the experiments, we simply fix $C = 1$ for all the three methods, and set $B = m \times N, m \in \{0.1, 1, 2, 10\}$.

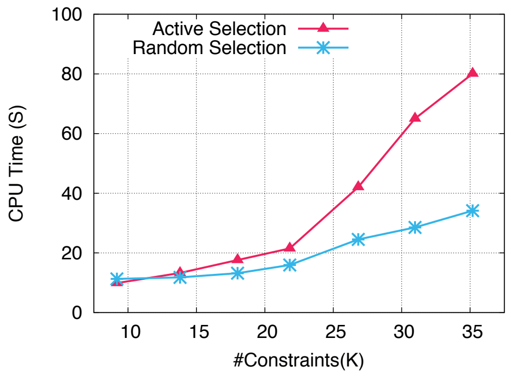
7.5.1 COLORED MAXIMUM VARIANCE UNFOLDING

The first experiment is to examine the efficiency by applying the proposed SimpleNPKL technique to solve the CMVU problem. In particular, we examine the CMVU task for learning low-dimensional embedding on three data sets which were used in Song et al. (2008). Two approaches are compared:

- **CMVU**: An approximate efficient method employed by Song et al. (2008). Suppose $\mathbf{K} = \mathbf{VAV}'$, where $\mathbf{V}$ (of size $n \times d$, $d < n$) is fixed to be the bottom d eigenvectors of the graph



(a) Clustering Accuracy



(b) CPU Time

Figure 2: Comparisons of clustering accuracy and CPU time by active constraint selection and random selection (constraint selection time is included) on A1a with parameters: $B = N, C = 1, k = 20, r = 0.6$. Using all $3.9K$ constraints directly, the accuracy is 60.8 ± 2.9 and the CPU time is 81.6 seconds.

Laplacian of the neighborhood graph via $\mathcal{N}$. Thus the number of variables is reduced from n^2 to d^2.

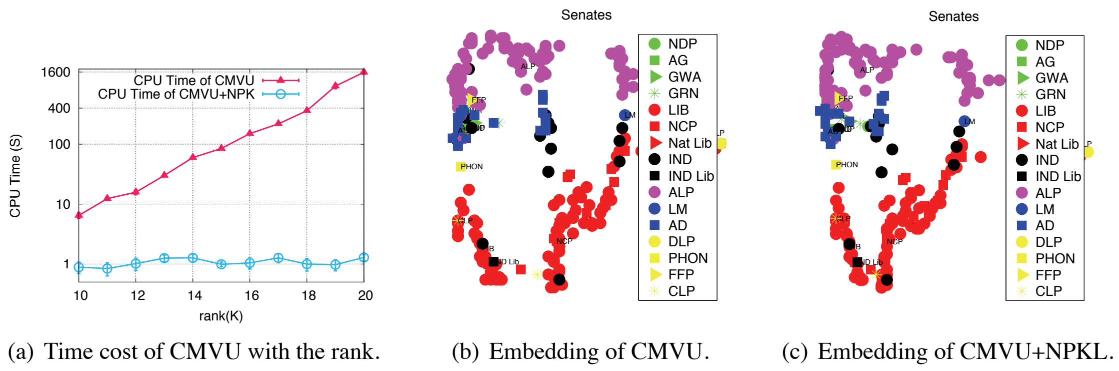
- **CMVU+NPKL**: Our SimpleNPKL method introduced in Section 6.1. Unlike the above CMVU algorithm by approximation, our method is able to obtain the global optimal solution using the SimpleNPKL scheme without approximation.

Figure 3, 4 and 5 show the experimental results of visualizing the embedding results in a 2D space and the CPU time cost of CMVU. The time costs of CMVU+NPKL were also indicated in the captions of those figures. As we can observe from the visualization results, the proposed CMVU+NPKL is able to produce comparable embedding results as those by the original CMVU in most cases. Further, by examining the time cost, we found that the time cost of CMVU increases with dimensionality d exponentially due to the intensive computational cost of solving the SDP problem. In contrast, the proposed CMVU+NPKL is able to find the global optima efficiently, which is much faster than CMVU when d is large. Although CMVU could be faster than CMVU+NPKL for very small d values, it is important to note that the optimal d value is often unknown for many applications. The proposed CMVU+NPKL approach can efficiently and directly resolve the CMVU problem without soliciting the approximation step.

7.5.2 MINIMUM VOLUME EMBEDDING AND STRUCTURE PRESERVING EMBEDDING

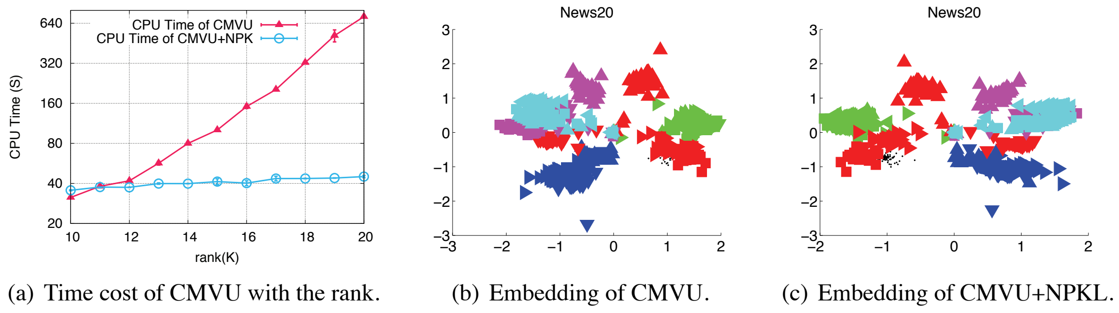
This experiment is to examine the embedding performance of the SimpleNPKL technique with applications to MVE (Shaw and Jebara, 2007) and SPE (Shaw and Jebara, 2009) tasks. In particular, five approaches are compared:

- **KPCA**: The classical Kernel Principle Component Analysis algorithm;
- **MVE**: The algorithm summarized in Table 1 in Shaw and Jebara (2007). Pay attention to the SDP solver in Step 5 and 6, which is the key for the success of MVE.
- **MVE+NPKL**: The embedding algorithm based on our SimpleNPKL algorithm. Refer to Section 6.2 for detailed discussion.



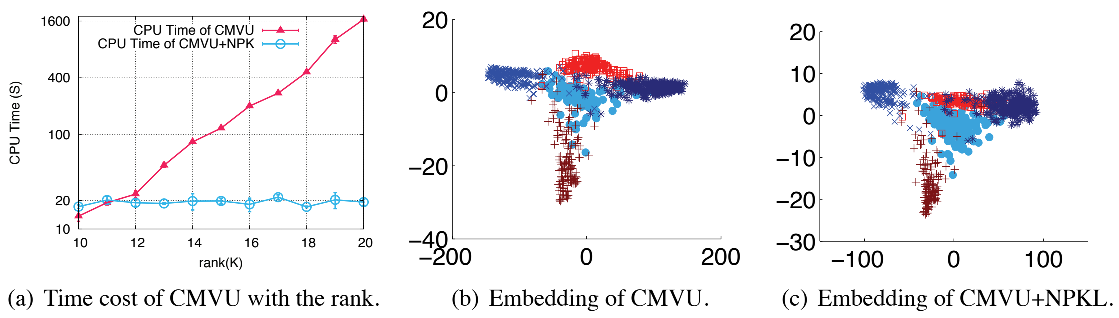
(a) Time cost of CMVU with the rank. (b) Embedding of CMVU. (c) Embedding of CMVU+NPKL.

Figure 3: Comparisons of CMVU and CMVU+NPKL on *senate* data set. **Time cost of CMVU+NPK is** 1.50 ± 0.06 **seconds.**



(a) Time cost of CMVU with the rank. (b) Embedding of CMVU. (c) Embedding of CMVU+NPKL.

Figure 4: Comparisons of CMVU and CMVU+NPK on *news20* data set. **Time cost of CMVU+NPKL is** 120.4 ± 1.7 **seconds.**



(a) Time cost of CMVU with the rank. (b) Embedding of CMVU. (c) Embedding of CMVU+NPKL.

Figure 5: Comparisons of CMVU and CMVU+NPKL on *usps* data set. **Time cost of CMVU+NPKL is** 28.95 ± 1.8 **seconds.**

- **SPE**: The algorithm summarized in Table 1 of Shaw and Jebara (2009).
- **SPE+NPKL**: The embedding algorithm based on the proposed SimpleNPKL algorithm. Refer to Section 6.3;

To examine the embedding results quantitatively, we follow the previous studies (Shaw and Jebara, 2007, 2009) to evaluate the classification performance on the embedding data by performing k-nearest neighbor classification. Similar to the settings in Shaw and Jebara (2009), we randomly choose 100 points from the largest two classes for each data set, and then divide the data examples into training/validation/test sets at the ratios of 60:20:20. The validation set is used to find the best parameter of k for k-NN classification.

Table 8 shows the comparison of the classification results by five different approaches. From the results, we can see that the two proposed algorithms, MVE+NPKL and SPE+NPKL, are generally able to achieve the competitive classification results that are comparable to the other two original algorithms using a standard SDP solver. Among all compared algorithms, MVE+NPKL tends to achieve slightly better performance than the other approaches. All these results show that the proposed algorithms are effective to produce comparable embedding performance.

Data Set	KPCA	MVE	MVE+NPKL	SPE	SPE+NPKL
Wine	90.5 ±5.6	**91.9 ±6.6**	90.9±5.8	75.2 ±0.09	87.1 ±7.9
Ionosphere	79.8±7.3	**86.3 ±7.3**	84.2±8.5	80.4 ±10.4	83.6 ±7.8
Heart	**65.6 ±8.4**	62.4 ±9.8	62.9 ±9.8	54.9 ±10.2	62.2 ±11.1
Sonar	58.2 ±12.4	59.2 ±10.2	**59.8 ±12.2**	57.4 ±11.1	59.4 ±11.4
Glass	70.7 ±9.8	73.5 ±7.8	**74.5 ±10.4**	61.7 ±9.7	69.4 ±8.7
Spiral	98.7 ±2.4	69.1 ±9.8	**98.8 ±2.4**	76.7 ±0.07	82.9 ±8.4
Australian	63.2 ±9.8	61.3 ±8.2	**63.8 ±9.3**	60.1 ± 0.10	59.5 ±10.1
Breast cancer	91.9 ±5.4	92.9 ±4.6	92.4 ±5.8	93.4 ±0.07	**94.4 ±5.5**

Table 8: k-NN classification accuracy on the 2D embedded results. (The best results are bolded.)

Next we compare the computational cost of the proposed algorithms against their original methods, respectively. Table 9 shows the summary of average CPU time cost of the compared algorithms. From the results, it is clear to see that the two proposed algorithms, MVE+NPKL and SPE+NPKL, are significantly more efficient than the other two original algorithms, respectively. By comparing MVE and MVE+NPKL, we found that MVE+NPKL achieves about 10 to 30 times speedups over the original MVE algorithm; the speedup values are even more significant for the SPE problem, where the proposed SPE+NPKL algorithm attains about 50 to 90 times speedup over the original SPE algorithm. These promising results again validate the proposed SimpleNPKL is effective for improving the efficiency and scalability of the three data embedding tasks.

To further illustrate the scalability of SPE+NPKL, we propose to solve a real-world embedding task on a large data set. In particular, we crawled a Flickr[7] data set, which consists of 3,660 Flickr user profiles and a collection of 3.7 million photos uploaded by these users. Each photo was annotated with a set of textual tags by users. Accordingly the photos for a particular Flickr user are described by tiling these tags. In total, our data set has 359,832 tags and 93,692 unique tags. Each Flickr user has a contact list, which is a collection of Flickr users who may share similar tastes / interests in their photo sharing. In our data set, every user has 19.1 contacts on average. We thus set $|\mathcal{N}|$ to 20 in both MVE and SPE. Moreover, there are 97,212 interest groups, and each Flickr user could belong to one or more interest groups. We compute *tf-idf* weight for the tags to represent a Flickr user (here the document frequency for a tag is actually the number of users annotated with

7. Flickr can be found at `http://www.flickr.com/`.

Data Set	MVE	MVE+NPKL	SpeedUp	SPE	SPE+NPKL	SpeedUp
Wine	2.92 ±0.06	**0.34 ±0.04**	8.5	47.72 ±0.29	0.51 ±0.01	93.6
Ionosphere	16.98 ±0.16	1.14 ±0.01	14.9	30.07 ±1.25	**0.60 ±0.01**	50.1
Heart	9.64 ±0.00	**0.38 ±0.02**	25.3	48.18 ±0.31	0.51 ±0.11	94.5
Sonar	7.50 ±0.13	**0.46 ±0.01**	16.3	30.40 ±1.16	0.61 ±0.02	49.8
Glass	11.08 ±0.26	**0.39 ±0.01**	28.2	29.10 ±0.12	0.53 ±0.01	54.9
Spiral	18.28 ±0.28	**0.46 ±0.00**	39.7	47.91 ±0.91	0.48 ±0.01	99.8
Australian	4.61 ±0.03	**0.30 ±0.02**	15.4	28.94 ±0.11	0.53 ±0.01	54.6
Breast cancer	16.59 ±0.10	**0.49 ±0.02**	33.9	48.72 ±0.26	0.56 ±0.01	87.0

Table 9: The evaluation of CPU time cost of different algorithms and the speedup of the SimpleNPKL method over the standard SDP solver. (The best results are bolded.)

that tag, that is, one or more photos of this user annotated with the tag). The k-nearest neighbor graph for MVE is constructed using cosine similarity between Flickr users. For SPE, we further constrain that the intra-group distance is smaller than the inter-group distance. In general, people who are friends or similar to each other tend to join the same interest group. Our goal is to apply the proposed MVE+NPKL and SPE+NPKL algorithms on these Flickr users in order to draw the 2D embedding results of the Flickr users exclusively belonging to two different interest groups: *B&W*[8] and *Catchy Colors*[9] as shown in Figure 7.

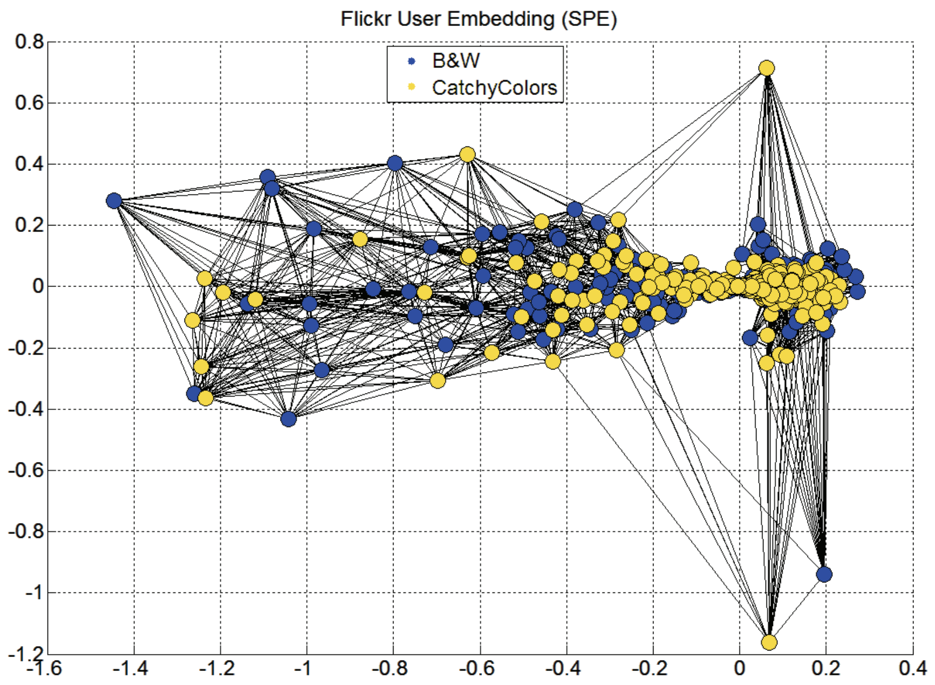
(a): Sample of B&W

(b): Sample of Catchy Colors

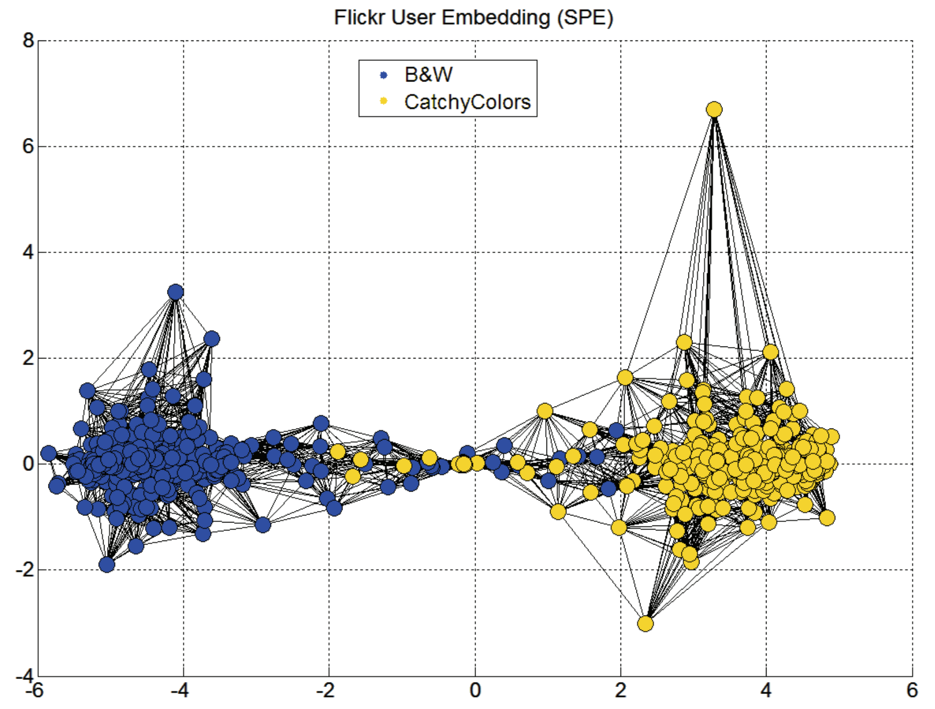
Figure 6: Sample photos from two Flickr interest groups: *B&W* and *Catchy Colors*.

Specifically, the theme of the group B&W is related to a collection of photos with black and white color only. The corresponding top 5 annotated tags for B&W are {*bw, black and white, black, white, portrait*}. In contrast, the top 5 tags for CatchyColors include {*red, blue, green, flower, yellow*}. Therefore, photos in the latter group are more colorful than the ones in B&W. An illustration of photos belonging to these two groups are depicted in Figure 6. However, the semantics of photos of these two groups are highly overlapping. Accordingly, the embedding results of MVE are highly overlapped as shown in Figure 7 (a), though it drives the spectral information into the top

8. *B&W* can be found at `http://www.flickr.com/groups/blackwhite/`.
9. *Catchy Colors* can be found at `http://www.flickr.com/groups/catchy/`.



(a) MVE+NPKL



(b) SPE+NPKL

Figure 7: The 2D embedding result of Flickr users exclusively belonging to the interest group *B&W* (blue points) and *Catchy Colors* (red points). The CPU time cost of MVE+NPKL and SPE+NPKL are 27.2 minutes and 196.4 minutes, respectively.

eigenvectors of the learned kernel matrix. On the other hand, by constraining intra-group distance less that inter-group distance, SPE can preserve the topology structure of these two groups as shown in Figure 7 (b). The 2D embedding shows the cluster structure rather clearly.

Note that the original algorithms using general SDP solvers cannot directly apply on the above large real data set. The proposed SimpleNPKL framework makes it feasible to analyze the emerging social networking problems using kernel learning methods. We hope our preliminary attempt in this paper could shed a light on a series of important applications in the future, including: 1) **Visualization:** as illustrated in Figure 7, we are able to obtain an intuitive understanding about the distribution of the entities in a social networking community. From Figure 7 (b), one can also observe the abnormal entities (e.g., the red dot on the right upper corner) and prototypes (the ones located at the centers of clusters). This may also benefit spam user detection and important user identification applications; 2) **Friend suggestion:** Given a Flickr user U_i, we can rank the other users U_j according to their similarity to U_i computed by the learned non-parametric kernel K_{ij}. With such information, a user can quickly find the other users of similar interests/tastes in photo sharing so as to facilitate the social communication between the users; 3) **Interest group recommendation:** It is interesting and beneficial to develop an intelligent scheme for recommending a Flickr user some interest groups. By applying the proposed kernel learning techniques to find similarity between Flickr users, it is possible for us to develop some recommendation scheme that suggests a Flickr user some interest groups that received the highest numbers of votes from its neighbors.

8. Conclusion

In this paper, we investigated a family of SimpleNPKL algorithms for improving the efficiency and scalability of the Non-Parametric Kernel Learning (NPKL) from large sets of pairwise constraints. We demonstrated that the proposed SimpleNPKL algorithm with linear loss for the pairwise constraints enjoys a closed-form solution, which can be simply computed by efficient sparse eigen-decomposition, such as the Lanczos algorithm. Moreover, our SimpleNPKL algorithm using other loss functions (including square hinge loss, hinge loss, and square loss) can be transformed into a saddle-point minimax optimization problem, which can be solved by an efficient iterative optimization procedure that only involves sparse eigen-decomposition computation. In contrast to the previous standard SDP solution, empirical results show that our approach achieved the same/comparable accuracy, but is significantly more efficient and scalable for large-scale data sets. We also explore some active constraint selection scheme to reduce the pairwise constraints in SimpleNPKL, which can further improve both computational efficiency and the clustering performance. Finally, we also demonstrate that the proposed family of SimpleNPKL algorithms can be applicable to other similar machine learning problems, in which we studied three example applications on data embedding problems. In the future, we will extend our technique for solving other SDP related machine learning problems.

Acknowledgments

We thank Dale Schuurmans' valuable comments and discussions. We also thank the valuable comments by the Action Editor and the anonymous reviewers. This research was in part supported by

Singapore MOE Tier-1 Grant (RG67/07), Tier-2 Grant (T208B2203), Tier-1 Grant (RG15/08), and A* SERC Grant (102 158 0034).

References

F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton. Complementarity and nondegeneracy in semidefinite programming. *Mathematical Programming*, 77:111–128, 1997.

A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceedings of Annual Conference on Learning Theory*, pages 338–352, 2005.

F. R. Bach and Z. Harchaoui. DIFFRAC: a discriminative and flexible framework for clustering. In *Advances in Neural Information Processing Systems*, pages 49–56, 2008.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of International Conference on Machine Learning*, 2004.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of International Conference on Machine Learning*, pages 97–104, 2006.

J. F. Bonnans and E. Shapiro. Optimization problems with perturbations, a guided tour. *SIAM Review*, 40:228–264, 1996.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd and L. Xiao. Least-squares covariance matrix adjustment. *SIAM Journal on Matrix Analysis and Applications*, 27(2):532–546, 2005.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 15*, pages 585–592, 2003.

J. Chen and J. Ye. Training SVM with indefinite kernels. In *International Conference on Machine Learning*, pages 136–143, 2008.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, pages 367–373, 2002.

L. Duan, I.W. Tsang, D. Xu, and S.J. Maybank. Domain transfer SVM for video concept detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

P. V. Gehler and S. Nowozin. Infinite kernel learning. In *TECHNICAL REPORT NO. TR-178,Max Planck Institute for Biological Cybernetics*, 2008.

M. Gönen and E. Alpaydin. Localized multiple kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 352–359, 2008.

A. Gretton, O. Bousquet, A. J. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Proceedings of International Conference on Algorithmic Learning Theory*, pages 63–77, 2005.

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008.

S. C. H. Hoi and R. Jin. Active kernel learning. In *Proceedings of International Conference on Machine Learning*, pages 400–407, 2008.

S. C. H. Hoi, M. R. Lyu, and E. Y. Chang. Learning the unified kernel machines for classification. In *Proceedings of ACM SIGKDD conference on Knowledge Discovery and Data Mining*, pages 187–196, 2006.

S. C. H. Hoi, R. Jin, and M. R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *Proceedings of International Conference on Machine Learning*, pages 361–368, 2007.

T. Jebara and V. Shchogolev. B-matching for spectral clustering. In *European Conference on Machine Learning*, pages 679–686, September 2006.

R. Johnson and T. Zhang. Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 54(1):275–288, 2008.

K. Krishnan and J. E. Mitchell. A unifying framework for several cutting plane methods for semidefinite programming. *Optimization Methods and Software*, 21(1):57–74, 2006.

B. Kulis, M. Sustik, and I. S. Dhillon. Learning low-rank kernel matrices. In *Proceedings of International Conference on Machine Learning*, pages 505–512, 2006.

B. Kulis, M. A. Sustik, and I. S. Dhillon. Low-rank kernel learning with bregman matrix divergences. *Journal of Machine Learning Research*, 10:341–376, 2009.

J. Kwok and I. W. Tsang. Learning with idealized kernels. In *Proceedings of International Conference on Machine Learning*, pages 400–407, 2003.

G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

R. B. Lehoucq, D. C. Sorensen, and C. Yang. ARPACK users guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods. Technical report, 1998.

D. P. Lewis, T. Jebara, and W. S. Noble. Nonstationary kernel combination. In *Proceedings of International Conference on Machine Learning*, pages 553–560, 2006.

F. Li, Y. Fu, Y.-H. Dai, C. Sminchisescu, and J. Wang. Kernel learning by unconstrained optimization. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2009.

M.S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra Applications*, 284:193–228, 1998.

R. Luss and A. d'Aspremont. Support vector machine classification with indefinite kernels. In *Advances in Neural Information Processing Systems 20*, 2008.

Q. Mao and I. W. Tsang. Parameter-free spectral kernel learning. In *Conference on Uncertainty in Artificial Intelligence*, 2010.

Q. Mao and I. W. Tsang. Multiple template learning for structured prediction. arXiv CoRR 1103.0890, Mar 2011.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103: 127–152, 2005.

Y. Nesterov and A. Nemirovskii. *Interior-point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.

G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. Technical Report MSRR-604, Carnegie Mellon University, August 1995.

F. R. Rakotomamonjy, A.and Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

B. Shaw and T. Jebara. Minimum volume embedding. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2007.

B. Shaw and T. Jebara. Structure preserving embedding. In *Proceedings of Interational Conference on Machine Learning*, page 118, 2009.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of International Conference on Machine Learning*, pages 824–831, 2005.

L. Song, A. J. Smola, K. M. Borgwardt, and A. Gretton. Colored maximum variance unfolding. In *Advances in Neural Information Processing Systems 20*, 2008.

S. Sonnenburg, G. Rätsch, and C. Schäfer. A general and efficient multiple kernel learning algorithm. In *Advances in Neural Information Processing Systems 18*, 2006a.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006b.

J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *Proceedings of Interational Conference on Machine Learning*, page 134, 2009.

A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2009.

K. Q. Weinberger and L. K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of International Conference on Machine Learning*, pages 1160–1167, 2008.

K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of International Conference on Machine Learning*, 2004.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, 2003.

Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 1825–1832, 2008.

Y. Ying, C. Campbell, and M. Girolami. Analysis of SVM with indefinite kernels. In *Advances in Neural Information Processing Systems*, 2010.

T. Zhang and R. Ando. Analysis of spectral kernel design based semi-supervised learning. In *Advances in Neural Information Processing Systems 18*, 2006.

X. Zhu, J. S. Kandola, Z. Ghahramani, and J. D. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2005.

J. Zhuang and S. C. H. Hoi. Non-parametric kernel ranking approach for social image retrieval. In *Proceedings of the 9th ACM International Conference on Image and Video Retrieval*, pages 26–33, 2010.

J. Zhuang, I. W. Tsang, and S. C. H. Hoi. SimpleNPKL: simple non-parametric kernel learning. In *Proceedings of Interational Conference on Machine Learning*, 2009.

J. Zhuang, I. W. Tsang, and S. C. H. Hoi. Two-layer multiple kernel learning. In *Proceedings of Interational Conference on Artificial Intelligence and Statistics*, 2011.

Journal of Machine Learning Research 12 (2011) 1349-1388 Submitted 4/10; Revised 1/11; Published 4/11

Faster Algorithms for Max-Product Message-Passing[*]

Julian J. McAuley[†] JULIAN.MCAULEY@NICTA.COM.AU
Tibério S. Caetano[†] TIBERIO.CAETANO@NICTA.COM.AU
Statistical Machine Learning Group
NICTA
Locked Bag 8001
Canberra ACT 2601, Australia

Editor: Tommi Jaakkola

Abstract

Maximum A Posteriori inference in graphical models is often solved via message-passing algorithms, such as the junction-tree algorithm or loopy belief-propagation. The exact solution to this problem is well-known to be exponential in the size of the maximal cliques of the triangulated model, while approximate inference is typically exponential in the size of the model's factors. In this paper, we take advantage of the fact that many models have maximal cliques that are larger than their constituent factors, and also of the fact that many factors consist only of latent variables (i.e., they do not depend on an observation). This is a common case in a wide variety of applications that deal with grid-, tree-, and ring-structured models. In such cases, we are able to decrease the exponent of complexity for message-passing by 0.5 for both exact *and* approximate inference. We demonstrate that message-passing operations in such models are equivalent to some variant of matrix multiplication in the tropical semiring, for which we offer an $O(N^{2.5})$ *expected-case* solution.

Keywords: graphical models, belief-propagation, tropical matrix multiplication

1. Introduction

It is well-known that exact inference in *tree-structured* graphical models can be accomplished efficiently by message-passing operations following a simple protocol making use of the distributive law (Aji and McEliece, 2000; Kschischang et al., 2001). It is also well-known that exact inference in *arbitrary* graphical models can be solved by the junction-tree algorithm; its efficiency is determined by the size of the maximal cliques after triangulation, a quantity related to the tree-width of the graph.

Figure 1 illustrates an attempt to apply the junction-tree algorithm to some graphical models containing cycles. If the graphs are not chordal ((a) and (b)), they need to be triangulated, or made chordal (red edges in (c) and (d)). Their clique-graphs are then guaranteed to be *junction-trees*, and the distributive law can be applied with the same protocol used for trees; see Aji and McEliece (2000) for a beautiful tutorial on exact inference in arbitrary graphs. Although the models in these

[*]. Preliminary versions of this work appeared in The 27th International Conference on Machine Learning (ICML 2010), and the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010), The NIPS 2009 Workshop on Learning with Orderings, The NIPS 2009 Workshop on Discrete Optimization in Machine Learning, and in Learning and Intelligent Optimization (LION 4).

[†]. Also at Research School of Information Sciences and Engineering, Australian National University, Canberra ACT 0200, Australia.

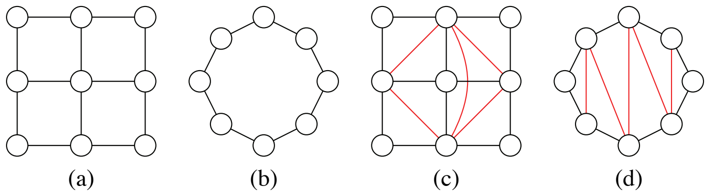


Figure 1: The models at left ((a) and (b)) can be triangulated ((c) and (d)) so that the junction-tree algorithm can be applied. Despite the fact that the new models have larger maximal cliques, the corresponding potentials are still factored over pairs of nodes only. Our algorithms exploit this fact.

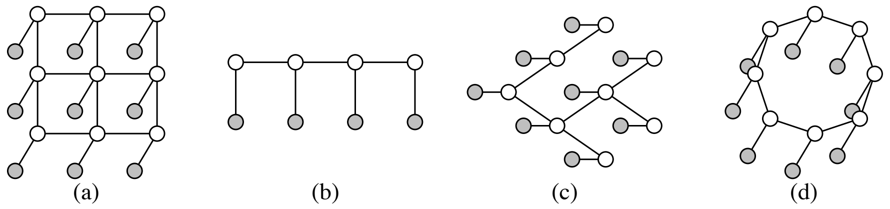


Figure 2: Some graphical models to which our results apply: *factors conditioned upon observations have fewer latent variables than purely latent factors*. White nodes correspond to latent variables, gray nodes to an observation. In other words, factors containing a gray node encode the *data likelihood*, whereas factors containing only white nodes encode *priors*. Expressed more simply, the 'node potentials' depend upon the observation, while the 'edge potentials' do not.

examples contain only pairwise factors, triangulation has increased the size of their maximal cliques, making exact inference substantially more expensive. Hence approximate solutions in the original graph (such as loopy belief-propagation, or inference in a loopy factor-graph) are often preferred over an exact solution via the junction-tree algorithm.

Even when the model's factors are the same size as its maximal cliques, neither exact nor approximate inference algorithms take advantage of the fact that many factors consist only of *latent* variables. In many models, those factors that are conditioned upon the observation contain fewer latent variables than the purely latent factors. Examples are shown in Figure 2. This encompasses a wide variety of models, including grid-structured models for optical flow and stereo disparity as well as chain and tree-structured models for text or speech.

In this paper, we exploit the fact that the maximal cliques (after triangulation) often have potentials that factor over subcliques, as illustrated in Figure 1. We will show that whenever this is the case, the expected computational complexity of message-passing between such cliques *can be improved* (both the asymptotic upper-bound and the actual runtime).

Additionally, we will show that this result can be applied in cliques *whose factors that are conditioned upon an observation* contain fewer latent variables than those factors consisting purely of latent variables; the 'purely latent' factors can be pre-processed *offline*, allowing us to achieve the same benefits as described in the previous paragraph.

We show that these properties reveal themselves in a wide variety of real applications.

A core operation encountered in the junction-tree algorithm is that of computing the inner-product of two vectors $\mathbf{v}_a$ and $\mathbf{v}_b$. In the max-product semiring (used for MAP inference), the 'inner-product' becomes

$$\max_{i \in \{1 \dots N\}} \{\mathbf{v}_a[i] \times \mathbf{v}_b[i]\}. \tag{1}$$

Our results stem from the realization that while (Equation 1) appears to be a *linear* time operation, it can be decreased to $O(\sqrt{N})$ (in the expected case) if we know the permutations that sort $\mathbf{v}_a$ and $\mathbf{v}_b$ (i.e., the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$). These permutations can be obtained efficiently when the model factorizes as described above.

Preliminary versions of this work have appeared in McAuley and Caetano (2009), McAuley and Caetano (2010a), and McAuley and Caetano (2010b).

1.1 Summary of Results

A selection of the results to be presented in the remainder of this paper can be summarized as follows:

- Our speedups apply to the operation of *passing a single message*. As a result, our method can be used regardless of the message-passing protocol.

- We are able to lower the asymptotic expected running time of max-product message-passing for *any* discrete graphical model whose cliques factorize into lower-order terms.

- The results obtained are exactly those that would be obtained by the traditional version of the algorithm, that is, no approximations are used.

- Our algorithm also applies whenever factors that are conditioned upon an observation contain fewer latent variables than those factors that are not conditioned upon an observation, as in Figure 2 (in which case certain computations can be taken offline).

- For pairwise models satisfying the above properties, we obtain an expected speed-up of *at least* $\Omega(\sqrt{N})$ (assuming N states per node; Ω denotes an *asymptotic lower-bound*). For example, in models with third-order cliques containing pairwise terms, message-passing is reduced from $\Theta(N^3)$ to $O(N^2\sqrt{N})$, as in Figure 1(d). For pairwise models whose edge potential is not conditioned upon an observation, message-passing is reduced from $\Theta(N^2)$ to $O(N\sqrt{N})$, as in Figure 2.

- For cliques composed of K-ary factors, the expected speed-up generalizes to at least $\Omega(\frac{1}{K}N^{\frac{1}{K}})$, though it is *never asymptotically slower* than the original solution.

- The expected-case improvement is derived under the assumption that the order statistics of different factors are *independent*.

- If the different factors have 'similar' order statistics, the performance will be better than the expected case.

- If the different factors have 'opposite' order statistics, the performance will be worse than the expected case, but is never asymptotically more expensive than the traditional version of the algorithm.

Our results do not apply for every semiring $(\oplus, \otimes)$, but only to those whose 'addition' operation defines an order (for example, min or max); we also assume that under this ordering, our 'multiplication' operator $\otimes$ satisfies

$$a < b \wedge c < d \;\; \Rightarrow \;\; a \otimes c < b \otimes d. \tag{2}$$

Thus our results certainly apply to the *max-sum* and *min-sum* ('tropical') semirings (as well as *max-product* and *min-product*, assuming non-negative potentials), but not for *sum-product* (for example). Consequently, our approach is useful for computing MAP-states, but cannot be used to compute marginal distributions. We also assume that the domain of each node is *discrete*.

We shall initially present our algorithm in terms of *pairwise* graphical models such as those shown in Figure 2. In such models message-passing is precisely equivalent to matrix-vector multiplication over our chosen semiring. Later we shall apply our results to models such as those in Figure 1, wherein message-passing becomes some variant of matrix multiplication. Finally we shall explore other applications besides message-passing that make use of tropical matrix multiplication as a subroutine, such all-pairs shortest-path problems.

1.2 Related Work

There has been previous work on speeding-up message-passing algorithms by exploiting different types of structure in certain graphical models. For example, Kersting et al. (2009) study the case where different cliques share the same potential function. In Felzenszwalb and Huttenlocher (2006), fast message-passing algorithms are provided for cases in which the potential of a 2-clique is only dependent on the *difference* of the latent variables (which is common in some computer vision applications); they also show how the algorithm can be made faster if the graphical model is a bipartite graph. In Kumar and Torr (2006), the authors provide faster algorithms for the case in which the potentials are *truncated*, whereas in Petersen et al. (2008) the authors offer speed-ups for models that are specifically grid-like.

The latter work is perhaps the most similar in spirit to ours, as it exploits the fact that certain factors can be *sorted* in order to reduce the search space of a certain maximization problem.

Another course of research aims at speeding-up message-passing algorithms by using 'informed' scheduling routines, which may result in faster convergence than the random schedules typically used in loopy belief-propagation and inference in factor graphs (Elidan et al., 2006). This branch of research is orthogonal to our own in the sense that our methods can be applied independently of the choice of message passing protocol.

Another closely related paper is that of Park and Darwiche (2003). This work can be seen to compliment ours in the sense that it exploits essentially the same type of factorization that we study, though it applies to *sum-product* versions of the algorithm, rather than the *max-product* version that we shall study. Kjærulff (1998) also exploits factorization within cliques of junction-trees, albeit a different type of factorization than that studied here.

In Section 4, we shall see that our algorithm is closely related to a well-studied problem known as 'tropical matrix multiplication' (Kerr, 1970). The worst-case complexity of this problem has been studied in relation to the all-pairs shortest-path problem (Alon et al., 1997; Karger et al., 1993).

Example	**description**
$A;B$	capital letters refer to sets of nodes (or similarly, cliques);
$A\cup B; A\cap B; A\setminus B$	standard set operators are used ($A\setminus B$ denotes set difference);
$\mathrm{dom}(A)$	the domain of a set; this is just the Cartesian product of the domains of each element in the set;
$\mathbf{P}$	bold capital letters refer to arrays;
$\mathbf{x}$	bold lower-case letters refer to vectors;
$\mathbf{x}[a]$	vectors are indexed using square brackets;
$\mathbf{P}[n]$	similarly, square brackets are used to index a *row* of a 2-d array,
$\mathbf{P}[\mathbf{n}]$	or a row of an $(\lvert\mathbf{n}\rvert+1)$-dimensional array;
$\mathbf{P}^X; \mathbf{v}^a$	superscripts are just labels, that is, $\mathbf{P}^X$ is an array, $\mathbf{v}^a$ is a vector;
$\mathbf{v}_a$	*constant* subscripts are also labels, that is, if a is a constant, then $\mathbf{v}_a$ is a constant vector;
$x_i; \mathbf{x}_A$	*variable* subscripts define variables; the subscript defines the domain of the variable;
$\mathbf{n}\lvert_X$	if $\mathbf{n}$ is a constant vector, then $\mathbf{n}\lvert_X$ is the *restriction* of that vector to those indices corresponding to variables in X (assuming that X is an ordered set);
$\Phi_A; \Phi_A(\mathbf{x}_A)$	a function over the variables in a set A; the argument $\mathbf{x}_A$ will be suppressed if clear, given that 'functions' are essentially arrays for our purposes;
$\Phi_{i,j}(x_i,x_j)$	a function over a pair of variables (x_i,x_j);
$\Phi_A(\mathbf{n}\lvert_B; \mathbf{x}_{A\setminus B})$	if one argument to a function is constant (here $\mathbf{n}\lvert_B$), then it becomes a function over fewer variables (in this case, only $\mathbf{x}_{A\setminus B}$ is free);

Table 1: Notation

2. Background

The notation we shall use is briefly defined in Table 1. We shall assume throughout that the *max-product* semiring is being used, though our analysis is almost identical for any suitable choice.

MAP-inference in a graphical model $\mathcal{G}$ consists of solving an optimization problem of the form

$$\hat{\mathbf{x}} = \operatorname*{argmax}_{\mathbf{x}} \prod_{C\in\mathcal{C}} \Phi_C(\mathbf{x}_C),$$

where $\mathcal{C}$ is the set of maximal cliques in $\mathcal{G}$. This problem is often solved via *message-passing* algorithms such as the junction-tree algorithm, loopy belief-propagation, or inference in a factor-graph (Aji and McEliece, 2000; Weiss, 2000; Kschischang et al., 2001).

Often, the clique-potentials $\Phi_C(\mathbf{x}_C)$ shall be decomposable into several smaller factors, that is,

$$\Phi_C(\mathbf{x}_C) = \prod_{F\subseteq C} \Phi_F(\mathbf{x}_F).$$

Some simple motivating examples are shown in Figure 3: a model for pose estimation from Sigal and Black (2006), a 'skip-chain CRF' from Galley (2006), and a model for shape-matching from Coughlan and Ferreira (2002). In each case, the triangulated model has third-order cliques, but the potentials are only pairwise. Other examples have already been shown in Figure 1; analogous cases are ubiquitous in many real applications.

It will often be more convenient to express our objective function as being conditioned upon some *observation*, $\mathbf{y}$. Thus our optimization problem becomes

$$\hat{\mathbf{x}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}} \prod_{C \in \mathcal{C}} \Phi_C(\mathbf{x}_C | \mathbf{y}) \tag{3}$$

(for simplicity when we discuss 'cliques' we are referring to sets of *latent* variables).

Further factorization may be possible if we express (Equation 3) in terms of those factors that depend upon the observation $\mathbf{y}$, and those that do not:

$$\hat{\mathbf{x}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}} \prod_{C \in \mathcal{C}} \Big\{ \underbrace{\prod_{F \subseteq C} \Phi_F(\mathbf{x}_F)}_{\text{data-independent}} \times \underbrace{\prod_{Q \subseteq C} \Phi_Q(\mathbf{x}_Q | \mathbf{y})}_{\text{data-dependent}} \Big\},$$

We shall say that those factors that are not conditioned on the observation are 'data-independent'.

Our results shall apply to message-passing equations in those cliques C where for each data-independent factor F we have $F \subset C$, *or* for each data-dependent factor Q we have $Q \subset C$, that is, when all F or all Q in C are *proper* subsets of C. In such cases we say that the clique C is *factorizable*.

The fundamental step encountered in message-passing algorithms is defined below. The message from a clique X to an intersecting clique Y (both sets of *latent* variables) is defined by

$$m_{X \to Y}(\mathbf{x}_{X \cap Y}) = \max_{\mathbf{x}_{X \setminus Y}} \left\{ \Phi_X(\mathbf{x}_X) \prod_{Z \in \Gamma(X) \setminus Y} m_{Z \to X}(\mathbf{x}_{X \cap Z}) \right\} \tag{4}$$

(where $\Gamma(X)$ is the set of neighbors of the clique X, that is, the set of cliques that intersect with X). If such messages are computed after X has received messages from all of its neighbors except Y (i.e., $\Gamma(X) \setminus Y$), then this defines precisely the update scheme used by the junction-tree algorithm. The same update scheme is used for loopy belief-propagation, though it is done iteratively in a randomized fashion.

After all messages have been passed, the MAP-state for a set of latent variables M (assumed to be a subset of a single clique X) is computed using

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X \setminus M}} \left\{ \Phi_X(\mathbf{x}_X) \prod_{Z \in \Gamma(X)} m_{Z \to X}(\mathbf{x}_{X \cap Z}) \right\}. \tag{5}$$

For cliques that are *factorizable* (according to our previous definition), both (Equation 4) and (Equation 5) take the form

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X \setminus M}} \left\{ \prod_{F \subseteq X} \Phi_F(\mathbf{x}_F) \prod_{Q \subseteq X} \Phi_Q(\mathbf{x}_Q | \mathbf{y}) \right\}. \tag{6}$$

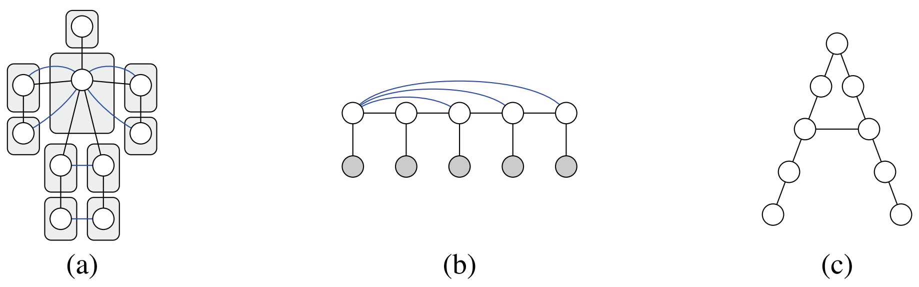


Figure 3: (a) A model for pose reconstruction from Sigal and Black (2006); (b) A ‘skip-chain CRF’ from Galley (2006); (c) A model for deformable matching from Coughlan and Ferreira (2002). Although the (triangulated) models have cliques of size three, their potentials factorize into pairwise terms.

Note that we always have $Z \cap X \subset X$ for messages $Z \to X$, meaning that the presence of the messages has no effect on the ‘factorizability’ of (Equation 6).

Algorithm 1 gives the traditional solution to this problem, which does not exploit the factorization of $\Phi_X(\mathbf{x}_X)$. This algorithm runs in $\Theta(N^{|X|})$, where N is the number of states per node, and $|X|$ is the size of the clique X (for a given $\mathbf{x}_X$, we treat computing $\prod_{F \subset X} \Phi_F(\mathbf{x}_F)$ as a constant time operation, as our optimizations shall not modify this cost).

In the following sections, we shall consider the two types of factorizability separately: first, in Section 3, we shall consider cliques X whose messages take the form

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X \setminus M}} \left\{ \Phi_X(\mathbf{x}_X) \prod_{Q \subset X} \Phi_Q(\mathbf{x}_Q | \mathbf{y}) \right\}.$$

We say that such cliques are *conditionally factorizable* (since all conditional terms factorize); examples are shown in Figure 2. Next, in Section 4, we consider cliques whose messages take the form

$$m_M(\mathbf{x}_M) = \max_{\mathbf{x}_{X \setminus M}} \prod_{F \subset X} \Phi_F(\mathbf{x}_F).$$

We say that such cliques are *latently factorizable* (since terms containing only latent variables factorize); examples are shown in Figure 1.

3. Optimizing Algorithm 1: Conditionally Factorizable Models

In order to specify a more efficient version of Algorithm 1, we begin by considering the simplest nontrivial *conditionally factorizable* model: a pairwise model in which each latent variable depends upon the observation, that is,

$$\hat{\mathbf{x}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{x}} \underbrace{\prod_{i \in \mathcal{N}} \Phi_i(x_i | y)}_{\text{node potential}} \times \underbrace{\prod_{(i,j) \in \mathcal{E}} \Phi_{i,j}(x_i, x_j)}_{\text{edge potential}}. \tag{7}$$

This is the type of model depicted in Figure 2 and encompasses a large class of grid- and tree-structured models. Using our previous definitions, we say that the node potentials are ‘data-dependent’, whereas the edge potentials are ‘data-independent’.

Algorithm 1 Brute-force computation of max-marginals

```
Input: a clique $X$ whose max-marginal $m_M(\mathbf{x}_M)$ (where $M \subset X$) we wish to compute; assume that
    each node in $X$ has domain $\{1 \ldots N\}$
1: for $\mathbf{m} \in \text{dom}(M)$ {i.e., $\{1 \ldots N\}^{|M|}$} do
2:   $max := -\infty$
3:   for $\mathbf{z} \in \text{dom}(X \setminus M)$ do
4:     if $\prod_{F \subset X} \Phi_F(\mathbf{m}|_F; \mathbf{z}|_F) > max$ then
5:       $max := \prod_{F \subset X} \Phi_F(\mathbf{m}|_F; \mathbf{z}|_F)$
6:     end if
7:   end for {this loop takes $\Theta(N^{|X \setminus M|})$}
8:   $m_M(\mathbf{m}) := max$
9: end for {this loop takes $\Theta(N^{|X|})$}
10: Return: $m_M$
```

Message-passing in models of the type shown in (Equation 7) takes the form

$$m_{A \to B}(x_i) = \Phi_i(x_i|y) \times \max_{x_j} \Phi_j(x_j|y) \times \Phi_{i,j}(x_i, x_j) \tag{8}$$

(where $A = \{i, j\}$ and $B = \{i, k\}$). Note once again that in (Equation 8) we are not concerned solely with exact inference via the junction-tree algorithm. In many models, such as grids and rings, (Equation 7) shall be solved *approximately* by means of either loopy belief-propagation, or inference in a factor-graph, which consists of solving (Equation 8) according to protocols other than the optimal junction-tree protocol.

It is useful to consider $\Phi_{i,j}$ in (Equation 8) as an $N \times N$ *matrix*, and Φ_j as an N-dimensional *vector*, so that solving (Equation 8) is precisely equivalent to matrix-vector multiplication in the max-product semiring. For a particular value $x_i = q$, (Equation 8) becomes

$$m_{A \to B}(q) = \Phi_i(q|y) \times \max_{x_j} \underbrace{\Phi_j(x_j|y)}_{\mathbf{v}_a} \times \underbrace{\Phi_{i,j}(q, x_j)}_{\mathbf{v}_b}, \tag{9}$$

which is precisely the 'max-product inner-product' operation that we claimed was critical in Section 1.

As we have previously suggested, it will be possible to solve (Equation 9) efficiently if we know the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$, that is, if we know the permutations that sort Φ_j and every row of $\Phi_{i,j}$ in (Equation 8). Sorting Φ_j takes $\Theta(N \log N)$, whereas sorting every row of $\Phi_{i,j}$ takes $\Theta(N^2 \log N)$ ($\Theta(N \log N)$ for each of N rows). The critical point to be made is that $\Phi_{i,j}(x_i, x_j)$ *does not depend on the observation*, meaning that its order statistics can be obtained *offline* in several applications.

The following elementary lemma is the key observation required in order to solve (Equation 1), and therefore (Equation 9) efficiently:

Lemma 1 *For any index q, the solution to $p = \text{argmax}_{i \in \{1 \ldots N\}} \{\mathbf{v}_a[i] \times \mathbf{v}_b[i]\}$ must have $\mathbf{v}_a[p] \geq \mathbf{v}_a[q]$ or $\mathbf{v}_b[p] \geq \mathbf{v}_b[q]$. Therefore, having computed $\mathbf{v}_a[q] \times \mathbf{v}_b[q]$, we can find 'p' by computing only those products $\mathbf{v}_a[i] \times \mathbf{v}_b[i]$ where either $\mathbf{v}_a[i] > \mathbf{v}_a[q]$ or $\mathbf{v}_b[i] > \mathbf{v}_b[q]$.*

Algorithm 2 Find i such that $\mathbf{v}_a[i] \times \mathbf{v}_b[i]$ is maximized

```
Input: two vectors $\mathbf{v}_a$ and $\mathbf{v}_b$, and permutation functions $p_a$ and $p_b$ that sort them in decreasing
    order (so that $\mathbf{v}_a[p_a[1]]$ is the largest element in $\mathbf{v}_a$)
 1: Initialize: $start := 1$, $end_a := p_a^{-1}[p_b[1]]$, $end_b := p_b^{-1}[p_a[1]]$ {if $end_b = k$, then the largest
    element in $\mathbf{v}_a$ has the same index as the $k^{\text{th}}$ largest element in $\mathbf{v}_b$}
 2: $best := p_a[1], max := \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
 3: if $\mathbf{v}_a[p_b[1]] \times \mathbf{v}_b[p_b[1]] > max$ then
 4:     $best := p_b[1], max := \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
 5: end if
 6: while $start < end_a$ {in practice, we could also stop if $start < end_b$, but the version given here is
    the one used for analysis in Appendix A} do
 7:     $start := start + 1$
 8:     if $\mathbf{v}_a[p_a[start]] \times \mathbf{v}_b[p_a[start]] > max$ then
 9:         $best := p_a[start]$
10:         $max := \mathbf{v}_a[best] \times \mathbf{v}_b[best]$
11:     end if
12:     if $p_b^{-1}[p_a[start]] < end_b$ then
13:         $end_b := p_b^{-1}[p_a[start]]$
14:     end if
15:     {repeat lines 8–14, interchanging $a$ and $b$}
16: end while {this loop takes expected time $O(\sqrt{N})$}
17: Return: $best$
```

This observation is used to construct Algorithm 2. Here we iterate through the indices starting from the largest values of $\mathbf{v}_a$ and $\mathbf{v}_b$, stopping once both indices are 'behind' the maximum value found so far (which we then know is the maximum). This algorithm is demonstrated pictorially in Figure 4. Note that Lemma 1 only depends upon the *relative* values of elements in $\mathbf{v}_a$ and $\mathbf{v}_b$, meaning that the number of computations that must be performed is purely a function of their *order statistics* (i.e., it does not depend on the actual values of $\mathbf{v}_a$ or $\mathbf{v}_b$).

If Algorithm 2 can solve (Equation 9) in $O(f(N))$, then we can solve (Equation 8) in $O(Nf(N))$. Determining precisely the running time of Algorithm 2 is not trivial, and will be explored in depth in Appendix A. At this stage we shall state an upper-bound on the true complexity in the following theorem:

Theorem 2 *The* expected *running time of Algorithm 2 is $O(\sqrt{N})$, yielding a speed-up of at least $\Omega(\sqrt{N})$ in cliques containing pairwise factors. This expectation is derived under the assumption that $\mathbf{v}_a$ and $\mathbf{v}_b$ have independent order statistics.*

Algorithm 3 uses Algorithm 2 to solve (Equation 8), where we assume that the order statistics of the rows of $\Phi_{i,j}$ have been obtained offline.

While the offline cost of sorting is not problematic in situations where the model is to be repeatedly reused on several observations, it can be avoided in two situations. Firstly, many models have a 'homogeneous' prior, that is, the same prior is shared amongst every edge (or clique) of the model. In such cases, only a single 'copy' of the prior needs to be sorted, meaning that in any model

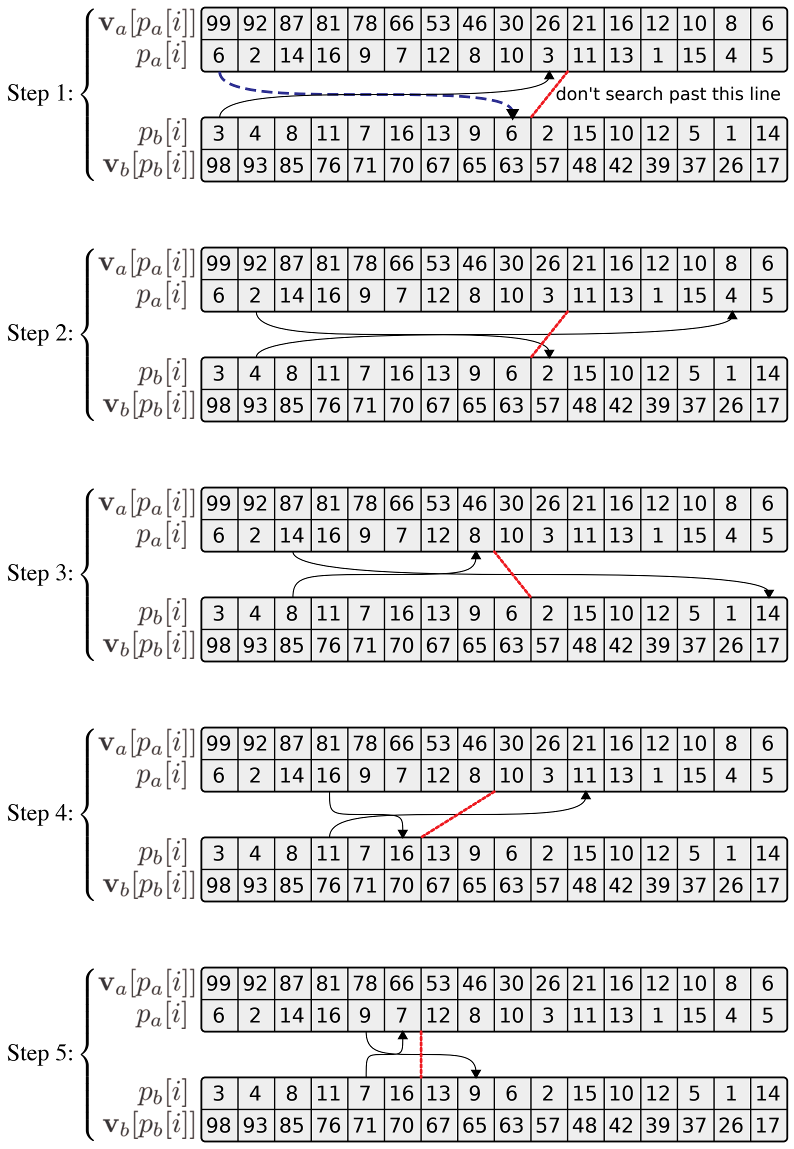


Figure 4: Algorithm 2, explained pictorially. The arrows begin at $p_a[start]$ and $p_b[start]$; the red dashed line connects end_a and end_b, behind which we need not search; a dashed arrow is used when a new maximum is found. Note that in the event that $\mathbf{v}_a$ and $\mathbf{v}_b$ contain repeated elements, they can be sorted arbitrarily.

containing $\Omega(\log N)$ edges, speed improvements can be gained over the naïve implementation. Secondly, where an iterative algorithm (such as loopy belief-propagation) is to be used, the sorting step need only take place prior to the *first* iteration; if $\Omega(\log N)$ iterations of belief-propagation are to be performed (or in a homogeneous model, if the number of edges multiplied by the number of

Algorithm 3 Solve (Equation 8) using Algorithm 2

```
Input: a potential $\Phi_{i,j}(a,b) \times \Phi_i(a|y_i) \times \Phi_j(b|y_j)$ whose max-marginal $m_i(x_i)$ we wish to compute,
    and a set of permutation functions $\mathbf{P}$ such that $\mathbf{P}[i]$ sorts the $i^{th}$ row of $\Phi_{i,j}$ (in decreasing order).
1: compute the permutation function $p_a$ by sorting $\Psi_j$ {takes $\Theta(N \log N)$}
2: for $q \in \{1 \ldots N\}$ do
3:     $(\mathbf{v}_a, \mathbf{v}_b) := (\Psi_j, \Phi_{i,j}(q, x_j|y_i, y_j))$
4:     $best := Algorithm2\,(\mathbf{v}_a, \mathbf{v}_b, p_a, \mathbf{P}[q])$ {$O(\sqrt{N})$}
5:     $m_{A \to B}(q) := \Phi_i(q) \times \Phi_j(best) \times \Phi_{i,j}(q, best|y_i, y_j)$
6: end for {this loop takes expected time $O(N\sqrt{N})$}
7: Return: $m_{A \to B}$
```

iterations is $\Omega(\log N)$), we shall again gain speed improvements even when the sorting step is done online.

In fact, the second of these conditions obviates the need for 'conditional factorizability' (or 'data-independence') altogether. In other words, in *any* pairwise model in which $\Omega(\log N)$ iterations of belief-propagation are to be performed, *the pairwise terms need to be sorted only during the first iteration*. Thus these improvements apply to those models in Figure 1, so long as the number of iterations of belief-propagation is $\Omega(\log N)$.

4. Latently Factorizable Models

Just as we considered the simplest *conditionally factorizable* model in Section 3, we now consider the simplest nontrivial *latently factorizable* model: a clique of size three containing pairwise factors. In such a case, our aim is to compute

$$m_{i,j}(x_i, x_j) = \max_{x_k} \Phi_{i,j,k}(x_i, x_j, x_k), \tag{10}$$

which we have assumed takes the form

$$m_{i,j}(x_i, x_j) = \max_{x_k} \Phi_{i,j}(x_i, x_j) \times \Phi_{i,k}(x_i, x_k) \times \Phi_{j,k}(x_j, x_k).$$

For a particular value of $(x_i, x_j) = (a, b)$, we must solve

$$m_{i,j}(a,b) = \Phi_{i,j}(a,b) \times \max_{x_k} \underbrace{\Phi_{i,k}(a, x_k)}_{\mathbf{v}_a} \times \underbrace{\Phi_{j,k}(b, x_k)}_{\mathbf{v}_b}, \tag{11}$$

which again is in precisely the form shown in (Equation 1).

Just as (Equation 8) resembled matrix-vector multiplication, there is a close resemblance between (Equation 11) and the problem of matrix-matrix multiplication in the max-product semiring (often referred to as 'tropical matrix multiplication', 'funny matrix multiplication', or simply 'max-product matrix multiplication'). While traditional matrix multiplication is well-known to have a subcubic worst-case solution (see Strassen, 1969), the version in (Equation 11) has no known subcubic solution (the fastest known solution is $O(N^3/\log N)$, but there is no known solution that runs in $O(N^{3-\epsilon})$ (Chan, 2007); Kerr (1970) shows that no subcubic solution exists under certain models of computation). The worst-case complexity of solving (Equation 11) can also be shown to be

Algorithm 4 Use Algorithm 2 to compute the max-marginal of a 3-clique containing pairwise factors

```
Input: a potential $\Phi_{i,j,k}(a,b,c) = \Phi_{i,j}(a,b) \times \Phi_{i,k}(a,c) \times \Phi_{j,k}(b,c)$ whose max-marginal $m_{i,j}(x_i,x_j)$ we wish to compute
 1: for $n \in \{1 \dots N\}$ do
 2:     compute $\mathbf{P}^i[n]$ by sorting $\Phi_{i,k}(n,x_k)$ {takes $\Theta(N \log N)$}
 3:     compute $\mathbf{P}^j[n]$ by sorting $\Phi_{j,k}(n,x_k)$ {$\mathbf{P}^i$ and $\mathbf{P}^j$ are $N \times N$ arrays, each row of which is a permutation; $\Phi_{i,k}(n,x_k)$ and $\Phi_{j,k}(n,x_k)$ are functions over $x_k$, since $n$ is constant in this expression}
 4: end for {this loop takes $\Theta(N^2 \log N)$}
 5: for $(a,b) \in \{1 \dots N\}^2$ do
 6:     $(\mathbf{v}_a, \mathbf{v}_b) := \big(\Phi_{i,k}(a,x_k), \Phi_{j,k}(b,x_k)\big)$
 7:     $(p_a, p_b) := \big(\mathbf{P}^i[a], \mathbf{P}^j[b]\big)$
 8:     $best := Algorithm2\,(\mathbf{v}_a, \mathbf{v}_b, p_a, p_b)$ {takes $O(\sqrt{N})$}
 9:     $m_{i,j}(a,b) := \Phi_{i,j}(a,b) \times \Phi_{i,k}(a, best) \times \Phi_{j,k}(b, best)$
10: end for {this loop takes $O(N^2\sqrt{N})$}
    {the total running time is $O(N^2 \log N + N^2\sqrt{N})$, which is dominated by $O(N^2\sqrt{N})$}
11: Return: $m_{i,j}$
```

equivalent to the all-pairs shortest-path problem, which is studied in Alon et al. (1997). Although we shall not improve the worst-case complexity, Algorithm 2 leads to far better *expected-case* performance than existing solutions.

In principle Strassen's algorithm could be used to perform *sum-product* inference in the setting we discuss here, and indeed there has been some work on performing sum-product inference in graphical models that factorize (Park and Darwiche, 2003). Interestingly, there is also a sub-quadratic solution to sum-product matrix-vector multiplication that requires preprocessing (Williams, 2007), that is, the sum-product version of the setting we discussed in Section 3.

A prescription of how Algorithm 2 can be used to solve (Equation 10) is given in Algorithm 4. As we mentioned in Section 3, the expected-case running time of Algorithm 2 is $O(\sqrt{N})$, meaning that the time taken to solve Algorithm 4 is $O(N^2\sqrt{N})$.

5. Extensions

So far we have only considered the case of *pairwise* graphical models, though as mentioned our results can in principle be applied to any conditionally or latently factorizable models, no matter the size of the factors. Essentially our results about matrices become results about tensors. We first treat latently factorizable models, after which the same ideas can be applied to conditionally factorizable models.

5.1 An Extension to Higher-Order Cliques with Three Factors

The simplest extension that we can make to Algorithms 2, 3, and 4 is to note that they can be applied even when there are several overlapping terms in the factors. For instance, Algorithm 4 can

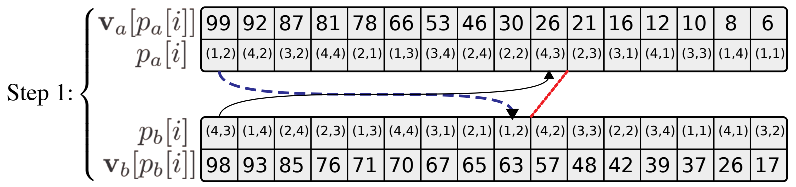


Figure 5: The reasoning applied in Algorithm 2 applies even when the elements of p_a and p_b are multidimensional indices.

be adapted to solve

$$m_{i,j}(x_i,x_j) = \max_{x_k,x_m} \Phi_{i,j}(x_i,x_j) \times \Phi_{i,k,m}(x_i,x_k,x_m) \times \Phi_{j,k,m}(x_j,x_k,x_m), \qquad (12)$$

and similar variants containing three factors. Here both x_k and x_m are shared by $\Phi_{i,k,m}$ and $\Phi_{j,k,m}$. We can follow precisely the reasoning of the previous section, except that when we sort $\Phi_{i,k,m}$ (similarly $\Phi_{j,k,m}$) for a fixed value of x_i, we are now sorting an *array* rather than a *vector* (Algorithm 4, lines 2 and 3); in this case, the permutation functions p_a and p_b in Algorithm 2 simply return *pairs* of indices. This is illustrated in Figure 5. Effectively, in this example we are sorting the variable $x_{k,m}$ whose domain is $\text{dom}(x_k) \times \text{dom}(x_m)$, which has state space of size N^2.

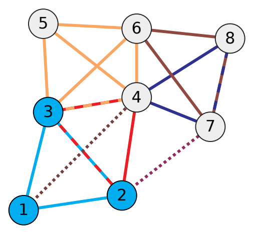
As the number of shared terms increases, so does the improvement to the running time. While (Equation 12) would take $\Theta(N^4)$ to solve using Algorithm 1, it takes only $O(N^3)$ to solve using Algorithm 4 (more precisely, if Algorithm 2 takes $O(f(N))$, then (Equation 12) takes $O(N^2 f(N^2))$, which we have mentioned is $O(N^2\sqrt{N^2}) = O(N^3)$). In general, if we have S shared terms, then the running time is $O(N^2\sqrt{N^S})$, yielding a speed-up of $\Omega(\sqrt{N^S})$ over the naïve solution of Algorithm 1.

5.2 An Extension to Higher-Order Cliques with Decompositions Into Three Groups

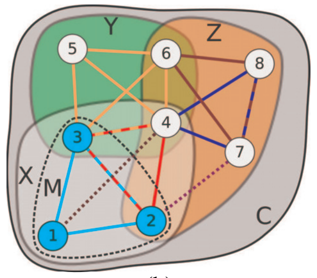
By similar reasoning, we can apply our algorithm to cases where there are more than three factors, in which the factors can be separated into three *groups*. For example, consider the clique in Figure 6(a), which we shall call G (the entire graph is a clique, but for clarity we only draw an edge when the corresponding nodes belong to a common factor). Each of the factors in this graph have been labeled using either differently colored edges (for factors of size larger than two) or dotted edges (for factors of size two), and the max-marginal we wish to compute has been labeled using colored nodes. We assume that it is possible to split this graph into three groups such that every factor is contained within a single group, along with the max-marginal we wish to compute (Figure 6, (b)). If such a decomposition is not possible, we will have to resort to further extensions to be described in Section 5.3.

Ideally, we would like these groups to have size $\simeq |G|/3$, though in the worst case they will have size no larger than $|G|-1$. We call these groups X, Y, Z, where X is the group containing the max-marginal M that we wish to compute. In order to simplify the analysis of this algorithm, we shall express the running time in terms of the size of the largest group, $S = \max(|X|,|Y|,|Z|)$, and the largest difference, $S_\setminus = \max(|Y \setminus X|, |Z \setminus X|)$. The max-marginal can be computed using Algorithm 5.

The running times shown in Algorithm 5 are loose upper-bounds, given for the sake of expressing the running time in simple terms. More precise running times are given in Table 2; any of the

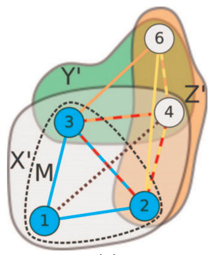


(a)

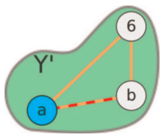


(b)

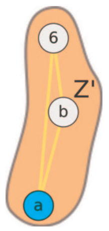
(a) We begin with a set of factors (indicated using colored lines), which are assumed to belong to some clique in our model; we wish to compute the max-marginal with respect to one of these factors (indicated using colored nodes); (b) The factors are split into three groups, such that every factor is entirely contained within one of them (Algorithm 5, line 1).



(c)

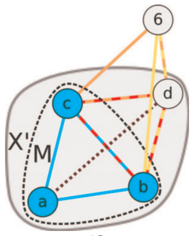


(d)

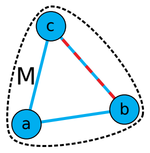


(e)

(c) Any nodes contained in only one of the groups are marginalized (Algorithm 5, lines 2, 3, and 4); the problem is now very similar to that described in Algorithm 4, except that *nodes* have been replaced by *groups*; note that this essentially introduces maximal factors in Y' and Z'; (d) For every value $(a,b) \in \mathrm{dom}(x_3,x_4)$, $\Psi^Y(a,b,x_6)$ is sorted (Algorithm 5, lines 5–7); (e) For every value $(a,b) \in \mathrm{dom}(x_2,x_4)$, $\Psi^Z(a,b,x_6)$ is sorted (Algorithm 5, lines 8–10).



(f)



(g)

(f) For every $\mathbf{n} \in \mathrm{dom}(X')$, we choose the best value of x_6 by Algorithm 2 (Algorithm 5, lines 11–16); (g) The result is marginalized with respect to M (Algorithm 5, line 17).

Figure 6: Algorithm 5, explained pictorially. In this case, the most computationally intensive step is the marginalization of Z (in step (c)), which takes $\Theta(N^5)$. However, the algorithm can actually be applied *recursively* to the group Z, resulting in an overall running time of $O(N^4\sqrt{N})$, for a max-marginal that would have taken $\Theta(N^8)$ to compute using the naïve solution of Algorithm 1.

Algorithm 5 Compute the max-marginal of G with respect to M, where G is split into three groups

```
Input: potentials $\Phi_G(\mathbf{x}) = \Phi_X(\mathbf{x}_X) \times \Phi_Y(\mathbf{x}_Y) \times \Phi_Z(\mathbf{x}_Z)$; each of the factors should be contained in
    exactly one of these terms, and we assume that $M \subseteq X$ (see Figure 6)
 1: Define: $X' := ((Y \cup Z) \cap X) \cup M$; $Y' := (X \cup Z) \cap Y$; $Z' := (X \cup Y) \cap Z$ {$X'$ contains the variables
    in $X$ that are shared by at least one other group; alternately, the variables in $X \setminus X'$ appear only
    in $X$ (sim. for $Y'$ and $Z'$)}
 2: compute $\Psi^X(\mathbf{x}_{X'}) := \max_{X \setminus X'} \Phi_X(\mathbf{x}_X)$ {we are marginalizing over those variables in $X$ that
    do not appear in any of the other groups (or in $M$); this takes $\Theta(N^S)$ if done by brute-force
    (Algorithm 1), but may also be done by a recursive call to Algorithm 5}
 3: compute $\Psi^Y(\mathbf{x}_{Y'}) := \max_{Y \setminus Y'} \Phi_Y(\mathbf{x}_Y)$
 4: compute $\Psi^Z(\mathbf{x}_{Z'}) := \max_{Z \setminus Z'} \Phi_Z(\mathbf{x}_Z)$
 5: for $\mathbf{n} \in \mathrm{dom}(X \cap Y)$ do
 6:     compute $\mathbf{P}^Y[\mathbf{n}]$ by sorting $\Psi^Y(\mathbf{n}; \mathbf{x}_{Y' \setminus X})$ {takes $\Theta(S_\setminus N^{S_\setminus} \log N)$; $\Psi^Y(\mathbf{n}; \mathbf{x}_{Y' \setminus X})$ is free over
        $\mathbf{x}_{Y' \setminus X}$, and is treated as an array by 'flattening' it; $\mathbf{P}^Y[\mathbf{n}]$ contains the $|Y' \setminus X| = |(Y \cap Z) \setminus X|$-
        dimensional indices that sort it}
 7: end for {this loop takes $\Theta(S_\setminus N^S \log N)$}
 8: for $\mathbf{n} \in \mathrm{dom}(X \cap Z)$ do
 9:     compute $\mathbf{P}^Z[\mathbf{n}]$ by sorting $\Psi^Z(\mathbf{n}; \mathbf{x}_{Z' \setminus X})$
10: end for {this loop takes $\Theta(S_\setminus N^S \log N)$}
11: for $\mathbf{n} \in \mathrm{dom}(X')$ do
12:     $(\mathbf{v}_a, \mathbf{v}_b) := \left(\Psi^Y(\mathbf{n}|_{Y'}; \mathbf{x}_{Y' \setminus X'}), \Psi^Z(\mathbf{n}|_{Z'}; \mathbf{x}_{Z' \setminus X'})\right)$ {$\mathbf{n}|_{Y'}$ is the 'restriction' of the vector $\mathbf{n}$ to those
        indices in $Y'$ (meaning that $\mathbf{n}|_{Y'} \in \mathrm{dom}(X' \cap Y')$); hence $\Psi^Y(\mathbf{n}|_{Y'}; \mathbf{x}_{Y' \setminus X'})$ is free in $\mathbf{x}_{Y' \setminus X'}$, while
        $\mathbf{n}|_{Y'}$ is fixed}
13:     $(p_a, p_b) := \left(\mathbf{P}^Y[\mathbf{n}|_{Y'}], \mathbf{P}^Z[\mathbf{n}|_{Z'}]\right)$
14:     $best := Algorithm2(\mathbf{v}_a, \mathbf{v}_b, p_a, p_b)$ {takes $O(\sqrt{S_\setminus})$}
15:     $m_X(\mathbf{n}) := \Psi^X(\mathbf{n}) \times \Psi^Y(best; \mathbf{n}|_{Y'}) \times \Psi^Z(best; \mathbf{n}|_{Z'})$
16: end for
17: $m_M(\mathbf{x}_M) := Algorithm1(m_X, M)$ {i.e., we are using Algorithm 1 to marginalize $m_X(\mathbf{x}_X)$ with
    respect to $M$; this takes $\Theta(N^S)$}
```

terms shown in Table 2 may be dominant. Some example graphs, and their resulting running times are shown in Figure 7.

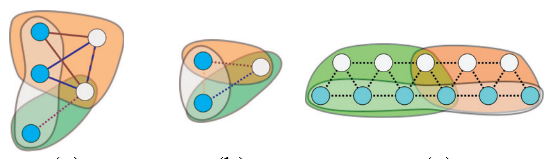
5.2.1 APPLYING ALGORITHM 5 RECURSIVELY

The marginalization steps of Algorithm 5 (lines 2, 3, and 4) may further decompose into smaller groups, in which case Algorithm 5 can be applied recursively. For instance, the graph in Figure 7(a) represents the marginalization step that is to be performed in Figure 6(c) (Algorithm 5, line 4). Since this marginalization step is the asymptotically dominant step in the algorithm, applying Algorithm 5 recursively lowers the asymptotic complexity.

Another straightforward example of applying recursion in Algorithm 5 is shown in Figure 8, in which a ring-structured model is marginalized with respect to two of its nodes. Doing so takes $O(MN^2\sqrt{N})$; in contrast, solving the same problem using the junction-tree algorithm (by triangulating the graph) would take $\Theta(MN^3)$. Loopy belief-propagation takes $\Theta(MN^2)$ per iteration, meaning

Description	lines	time
Marginalization of Φ_X, without recursion	2	$\Theta(N^{\lvert X \rvert})$
Marginalization of Φ_Y	3	$\Theta(N^{\lvert Y \rvert})$
Marginalization of Φ_Z	4	$\Theta(N^{\lvert Z \rvert})$
Sorting Φ_Y	5–7	$\Theta(\lvert Y' \setminus X \rvert N^{\lvert Y' \rvert} \log N)$
Sorting Φ_Z	8–10	$\Theta(\lvert Z' \setminus X \rvert N^{\lvert Z' \rvert} \log N)$
Running Algorithm 2 on the sorted values	11–16	$O(N^{\lvert X' \rvert} \sqrt{N^{\lvert (Y' \cap Z') \setminus X' \rvert}})$

Table 2: Detailed running time analysis of Algorithm 5; any of these terms may be asymptotically dominant



Graph:	(a)	(b)	(c)	(d)	{A complete graph K_M, with pairwise terms} (e)
Algorithm 1:	$\Theta(N^5)$	$\Theta(N^3)$	$\Theta(N^{11})$	$\Theta(N^6)$	$\Theta(N^M)$
Algorithm 5:	$O(N^3\sqrt{N})$	$O(N^2\sqrt{N})$	$O(N^6\sqrt{N})$	$O(N^5)$	$O(N^{5M/6})$
Speed-up:	$\Omega(N\sqrt{N})$	$\Omega(\sqrt{N})$	$\Omega(N^4\sqrt{N})$	$\Omega(N)$	$\Omega(N^{M/6})$

Figure 7: Some example graphs whose max-marginals are to be computed with respect to the colored nodes, using the three regions shown. Factors are indicated using differently colored edges, while dotted edges always indicate pairwise factors. (a) is the region Z from Figure 6 (recursion is applied *again* to achieve this result); (b) is the graph used to motivate Algorithm 4; (c) shows a query in a graph with regular structure; (d) shows a complete graph with six nodes; (e) generalizes this to a clique with M nodes.

that our algorithm will be faster if the number of iterations is $\Omega(\sqrt{N})$. Naturally, Algorithm 4 could be applied directly to the triangulated graph, which would again take $O(MN^2\sqrt{N})$.

5.3 A General Extension to Higher-Order Cliques

Naturally, there are cases for which a decomposition into three terms is not possible, such as

$$m_{i,j,k}(x_i,x_j,x_k) = \max_{x_m} \Phi_{i,j,k}(x_i,x_j,x_k) \times \Phi_{i,j,m}(x_i,x_j,x_m) \times \\ \Phi_{i,k,m}(x_i,x_k,x_m) \times \Phi_{j,k,m}(x_j,x_k,x_m) \quad (13)$$

(i.e., a clique of size four with all possible third-order factors). However, if the model contains factors of size K, it must always be possible to split it into $K+1$ groups (e.g., four in the case of Equation 13).

Our optimizations can easily be applied in these cases simply by adapting Algorithm 2 to solve problems of the form

$$\max_{i \in \{1 \dots N\}} \{\mathbf{v}_1[i] \times \mathbf{v}_2[i] \times \cdots \times \mathbf{v}_K[i]\}. \quad (14)$$

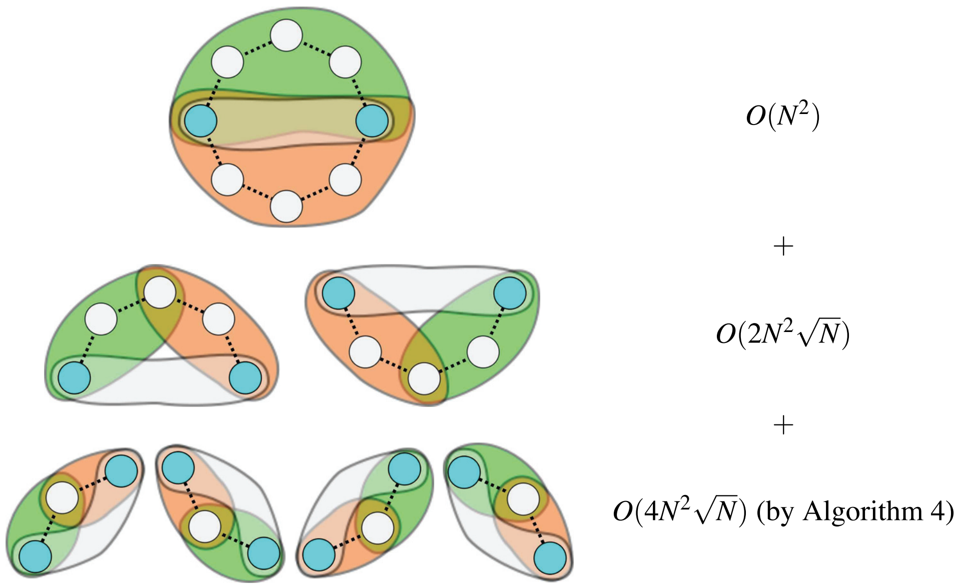


Figure 8: In the above example, lines 2–4 of Algorithm 5 are applied recursively, achieving a total running time of $O(MN^2\sqrt{N})$ for a loop with M nodes (our algorithm achieves the same running time in the triangulated graph).

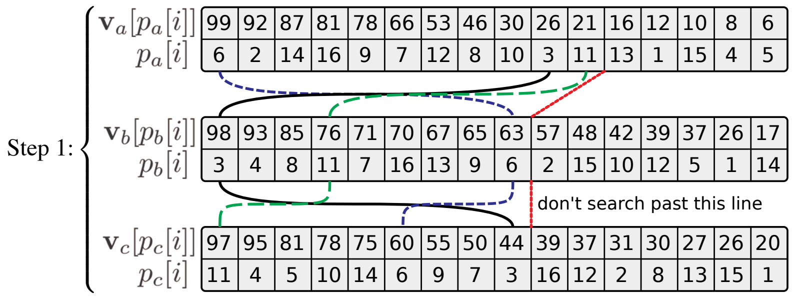

Step 1:
$\mathbf{v}_a[p_a[i]]$ 99 92 87 81 78 66 53 46 30 26 21 16 12 10 8 6
$p_a[i]$ 6 2 14 16 9 7 12 8 10 3 11 13 1 15 4 5
$\mathbf{v}_b[p_b[i]]$ 98 93 85 76 71 70 67 65 63 57 48 42 39 37 26 17
$p_b[i]$ 3 4 8 11 7 16 13 9 6 2 15 10 12 5 1 14
don't search past this line
$\mathbf{v}_c[p_c[i]]$ 97 95 81 78 75 60 55 50 44 39 37 31 30 27 26 20
$p_c[i]$ 11 4 5 10 14 6 9 7 3 16 12 2 8 13 15 1

Figure 9: Algorithm 2 can easily be extended to cases including more than two sequences.

Pseudocode for this extension is presented in Algorithm 6. Note carefully the use of the variable *read*: we are storing which indices have been read to avoid re-reading them; this guarantees that our Algorithm is never asymptotically worse than the naïve solution. Figure 9 demonstrates how such an algorithm behaves in practice. Again, we shall discuss the running time of this extension in Appendix A. For the moment, we state the following theorem:

Theorem 3 *Algorithm 6 generalizes Algorithm 2 to K lists with an expected running time of* $O(KN^{\frac{K-1}{K}})$*, yielding a speed-up of at least* $\Omega(\frac{1}{K}N^{\frac{1}{K}})$ *in cliques containing K-ary factors. It is never worse than the naïve solution, meaning that it takes* $O(\min(N, KN^{\frac{K-1}{K}}))$*.*

Using Algorithm 6, we can similarly extend Algorithm 5 to allow for any number of *groups* (pseudocode is not shown; all statements about the groups Y and Z simply become statements

Algorithm 6 Find i such that $\prod_{k=1}^{K} \mathbf{v}_k[i]$ is maximized

```
Input: $K$ vectors $\mathbf{v}_1 \dots \mathbf{v}_K$; permutation functions $p_1 \dots p_K$ that sort them in decreasing order; a
    vector $read$ indicating which indices have been read, and a unique value $T \notin read$ {$read$ is
    essentially a boolean array indicating which indices have been read; since *creating* this array is
    an $O(N)$ operation, we create it externally, and reuse it $O(N)$ times; setting $read[i] = T$ indicates
    that a particular index has been read; we use a different value of $T$ for each call to this function
    so that $read$ can be reused without having to be reinitialized}
 1: Initialize: $start := 1$,
    $max := \max_{p \in \{p_1 \dots p_K\}} \prod_{k=1}^{K} \mathbf{v}_k[p[1]]$,
    $best := \mathrm{argmax}_{p \in \{p_1 \dots p_K\}} \prod_{k=1}^{K} \mathbf{v}_k[p[1]]$
 2: for $k \in \{1 \dots K\}$ do
 3:     $end_k := \max_{q \in \{p_1 \dots p_K\}} p_k^{-1}[q[1]]$
 4:     $read[p_k[1]] = T$
 5: end for
 6: while $start < \max\{end_1 \dots end_K\}$ do
 7:     $start := start + 1$
 8:     for $k \in \{1 \dots K\}$ do
 9:         if $read[p_k[start]] := T$ then
10:             continue
11:         end if
12:         $read[p_k[start]] := T$
13:         $m := \prod_{x=1}^{K} \mathbf{v}_x[p_k[start]]$
14:         if $m > max$ then
15:             $best := p_k[start]$
16:             $max := m$
17:         end if
18:         $e_k := \max_{q \in \{p_1 \dots p_K\}} p_k^{-1}[q[start]]$
19:         $end_k := \min(e_k, end_k)$
20:     end for
21: end while {see Appendix A for running times}
22: Return: $best$
```

about K groups $\{G_1 \dots G_K\}$, and calls to Algorithm 2 become calls to Algorithm 6). The one remaining case that has not been considered is when the sequences $\mathbf{v}_1 \cdots \mathbf{v}_K$ are functions of different (but overlapping) variables; naïvely, we can create a new variable whose domain is the product space of all of the overlapping terms, and still achieve the performance improvement guaranteed by Theorem 3; in some cases, better results can again be obtained by applying recursion, as in Figure 7.

As a final comment we note that we have not provided an algorithm for choosing *how* to split the variables of a model into $(K+1)$-groups. We note even if we split the groups in a naïve way, we are guaranteed to get *at least* the performance improvement guaranteed by Theorem 3, though more ‘intelligent’ splits may further improve the performance.

Furthermore, in all of the applications we have studied, K is sufficiently small that it is inexpensive to consider all possible splits by brute-force.

5.4 Extensions for Conditionally Factorizable Models

Just as in Section 5.2, we can extend Algorithm 3 to factors of any size, so long as the purely latent cliques contain more latent variables than those cliques that depend upon the observation. The analysis for this type of model is almost exactly the same as that presented in Section 5.2, except that any terms consisting of purely latent variables are processed offline.

As we mentioned in 5.2, if a model contains (non-maximal) factors of size K, we will gain a speed-up of $\Omega(\frac{1}{K}N^{\frac{1}{K}})$. If in addition there is a factor (either maximal or non-maximal) consisting of purely latent variables, we can still obtain a speed-up of $\Omega(\frac{1}{K+1}N^{\frac{1}{K+1}})$, since this factor merely contributes an additional term to (Equation 14). Thus when our 'data-dependent' terms contain only a single latent variable (i.e., $K = 1$), we gain a speed-up of $\Omega(\sqrt{N})$, as in Algorithm 3.

6. Performance Improvements in Existing Applications

Our results are immediately compatible with several applications that rely on inference in graphical models. As we have mentioned, our results apply to *any model whose cliques decompose into lower-order terms*.

Often, potentials are defined only on *nodes* and *edges* of a model. A D^{th}-order Markov model has a tree-width of D, despite often containing only pairwise relationships. Similarly 'skip-chain CRFs' (Sutton and McCallum, 2006; Galley, 2006), and junction-trees used in SLAM applications (Paskin, 2003) often contain only pairwise terms, and may have low tree-width under reasonable conditions. These are examples of *latently factorizable* models. In each case, if the tree-width is D, Algorithm 5 takes $O(MN^D\sqrt{N})$ (for a model with M nodes and N states per node), yielding a speed-up of $\Omega(\sqrt{N})$.

Models for shape-matching and pose reconstruction often exhibit similar properties (Tresadern et al., 2009; Donner et al., 2007; Sigal and Black, 2006). In each case, third-order cliques factorize into second-order terms; hence we can apply Algorithm 4 to achieve a speed-up of $\Omega(\sqrt{N})$.

Another similar model for shape-matching is that of Felzenszwalb (2005); this model again contains third-order cliques, though it includes a 'geometric' term constraining all three variables. Here, the third-order term is *independent of the input data*, meaning that each of its rows can be sorted *offline*, as described in Section 3. This is an example of a *conditionally factorizable* model. In this case, those factors that depend upon the observation are pairwise, meaning that we achieve a speed-up of $\Omega(N^{\frac{1}{3}})$. Further applications of this type shall be explored in Section 7.4.

In Coughlan and Ferreira (2002), deformable shape-matching is solved approximately using loopy belief-propagation. Their model has only second-order cliques, meaning that inference takes $\Theta(MN^2)$ *per iteration*. Although we cannot improve upon this result, we note that we can typically do *exact* inference in a single iteration in $O(MN^2\sqrt{N})$; thus our model has the same running time as $O(\sqrt{N})$ iterations of the original version. This result applies to all second-order models containing a single loop (Weiss, 2000).

In McAuley et al. (2008), a model is presented for graph-matching using loopy belief-propagation; the maximal cliques for D-dimensional matching have size $(D+1)$, meaning that inference takes $\Theta(MN^{D+1})$ *per iteration* (it is shown to converge to the correct solution); we improve this to $O(MN^D\sqrt{N})$.

Interval graphs can be used to model resource allocation problems (Fulkerson and Gross, 1965); each node encodes a request, and overlapping requests form edges. Maximal cliques grow with the

Reference	description	running time	our method
McAuley et al. (2008)	D-d graph-matching	$\Theta(MN^{D+1})$ (iter.)	$O(MN^D\sqrt{N})$ (iter.)
Sutton and McCallum (2006)	Width-D skip-chain	$O(MN^D)$	$O(MN^{D-1}\sqrt{N})$
Galley (2006)	Width-3 skip-chain	$\Theta(MN^3)$	$O(MN^2\sqrt{N})$
Tresadern et al. (2009)	Deformable matching	$\Theta(MN^3)$	$O(MN^2\sqrt{N})$
Coughlan and Ferreira (2002)	Deformable matching	$\Theta(MN^2)$ (iter.)	$O(MN^2\sqrt{N})$
Sigal and Black (2006)	Pose reconstruction	$\Theta(MN^3)$	$O(MN^2\sqrt{N})$
Felzenszwalb (2005)	Deformable matching	$\Theta(MN^3)$	$\Theta(MN^{\frac{8}{3}})$ (online)
Fulkerson and Gross (1965)	Width-D interval graph	$O(MN^{D+1})$	$O(MN^D\sqrt{N})$

Table 3: Some existing work to which our results can be immediately applied (M is the number of nodes in the model, N is the number of states per node. 'iter.' denotes that the algorithm is iterative).

number of overlapping requests, though the constraints are only pairwise, meaning that we again achieve an $\Omega(\sqrt{N})$ improvement.

Finally, in Section 7.4 we shall explore a variety of applications in which we have pairwise models of the form shown in (Equation 7). In all of these cases, we see an (expected) reduction of a $\Theta(MN^2)$ message-passing algorithm to $O(MN\sqrt{N})$.

Table 3 summarizes these results. Reported running times reflect the *expected case*. Note that we are assuming that *max-product belief-propagation is being used in a discrete model*; some of the referenced articles may use different variants of the algorithm (e.g., Gaussian models, or approximate inference schemes). We believe that our improvements may revive the exact, discrete version as a tractable option in these cases.

7. Experiments

We present experimental results for two types of models: latently factorizable models, whose cliques factorize into smaller terms, as discussed in Section 4, and conditionally factorizable models, whose factors *that depend upon the observation* contain fewer latent variables than their maximal cliques, as discussed in Section 3.

We begin with an asymptotic analysis of the running time of our algorithm on the 'inner product' operations of (Equation 1) and (Equation 14), in order to assess Theorems 2 and 3 experimentally.

7.1 Comparison Between Asymptotic Performance and Upper-Bounds

For our first experiment, we compare the performance of Algorithms 2 and 6 to the naïve solution of Algorithm 1. These are core subroutines of each of the other algorithms, meaning that determining their performance shall give us an accurate indication of the improvements we expect to obtain in real graphical models.

For each experiment, we generate N i.i.d. samples from $[0,1)$ to obtain the lists $v_1 \dots v_K$. N is the domain size; this may refer to a single node, or a *group* of nodes as in Algorithm 6; thus large values of N may appear even for binary-valued models. K is the number of lists in (Equation 14); we can observe this number of lists only if we are working in cliques of size $K+1$, and then only if the factors are of size K (e.g., we will only see $K=5$ if we have cliques of size 6 with factors

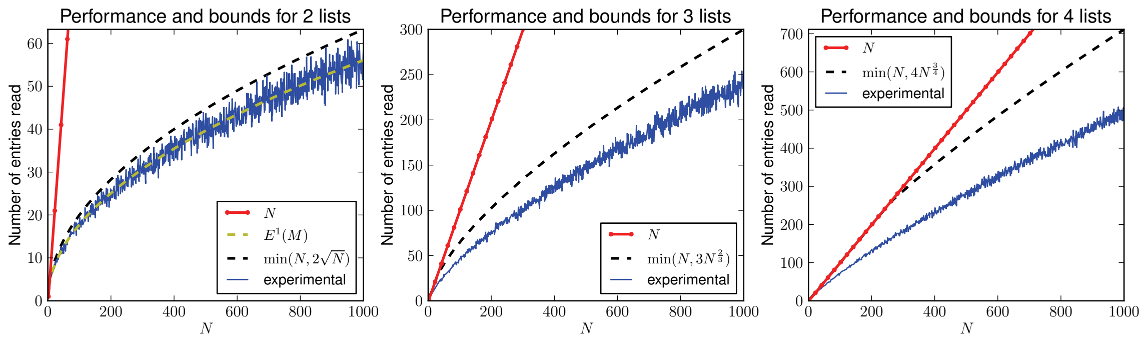


Figure 10: Performance of our algorithm and bounds. For $K = 2$, the exact expectation is shown, which appears to precisely match the average performance (over 100 trials). The dotted lines show the bound of (Equation 23). While the bound is close to the true performance for $K = 2$, it becomes increasingly loose for larger K.

of size 5); therefore smaller values of K are probably more realistic in practice (indeed, all of the applications in Section 6 have $K = 2$).

The performance of our algorithm is shown in Figure 10, for $K = 2$ to 4 (i.e., for 2 to 4 lists). When $K = 2$, we execute Algorithm 2, while Algorithm 6 is executed for $K \geq 3$. The performance reported is simply the number of elements read from the lists (which is at most $K \times$ *start*). This is compared to N itself, which is the number of elements read by the naïve algorithm. The upper-bounds we obtained in (Equation 23) are also reported, while the true expected performance (i.e., Equation 19) is reported for $K = 2$. Note that the variable *read* was introduced into Algorithm 6 in order to guarantee that it can never be asymptotically slower than the naïve algorithm. If this variable is ignored, the performance of our algorithm deteriorates to the point that it closely approaches the upper-bounds shown in Figure 10. Unfortunately, this optimization proved overly complicated to include in our analysis, meaning that our upper-bounds remain highly conservative for large K.

7.2 Performance Improvement for Dependent Variables

The expected-case running time of our algorithm was derived under the assumption that each list has independent order statistics, as was the case for our previous experiment. We suggested that we will obtain worse performance in the case of negatively correlated variables, and better performance in the case of positively correlated variables; we shall assess these claims in this experiment.

Figure 11 shows how the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$ can affect the performance of our algorithm. Essentially, the running time of Algorithm 2 is determined by the level of 'diagonalness' of the permutation matrices in Figure 11; highly diagonal matrices result in better performance than the expected case, while highly off-diagonal matrices result in worse performance. The expected case was simply obtained under the assumption that every permutation is equally likely.

We report the performance for two lists (i.e., for Algorithm 2), where each $(\mathbf{v}_a[i], \mathbf{v}_b[i])$ is an independent sample from a 2-dimensional Gaussian with covariance matrix

$$\Sigma = \begin{bmatrix} 1 & c \\ c & 1 \end{bmatrix},$$

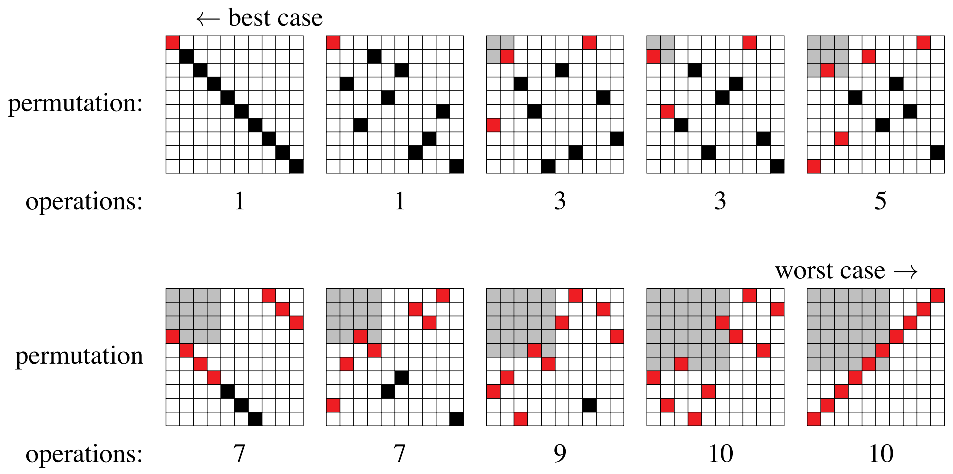


Figure 11: Different permutation matrices and their resulting cost (in terms of entries read/multiplications performed). Each permutation matrix transforms the *sorted* values of one list into the sorted values of the other, that is, it transforms $\mathbf{v}_a$ as sorted by p_a into $\mathbf{v}_b$ as sorted by p_b. The red (lighter) squares show the entries that must be read before the algorithm terminates (each corresponding to one multiplication). See Figure 23 for further explanation.

meaning that the two lists are correlated with correlation coefficient c (here we are working in the max-sum semiring). This dependence between the values of the two lists leads to a dependence in their order statistics, so that in the case of Gaussian random variables, the correlation coefficient precisely captures the 'diagonalness' of the matrices in Figure 11. Performance is shown in Figure 12 for different values of c ($c = 0$, is not shown, as this is the case observed in the previous experiment).

7.3 Message-Passing in Latently Factorizable Models

In this section we present experiments in models whose cliques factorize into smaller terms, as discussed in Section 4.

7.3.1 2-DIMENSIONAL GRAPH-MATCHING

Naturally, Algorithm 5 has additional overhead compared to the naïve solution, meaning that it will not be beneficial for small N. In this experiment, we aim to assess the extent to which our approach is faster in real applications. We reproduce the model from McAuley et al. (2008), which performs 2-dimensional graph-matching, using a loopy graph with cliques of size three, containing only second-order potentials (as described in Section 6); the $\Theta(NM^3)$ performance of McAuley et al. (2008) is reportedly state-of-the-art. We also show the performance on a graphical model with *random* potentials, in order to assess how the results of the previous experiments are reflected in terms of actual running time.

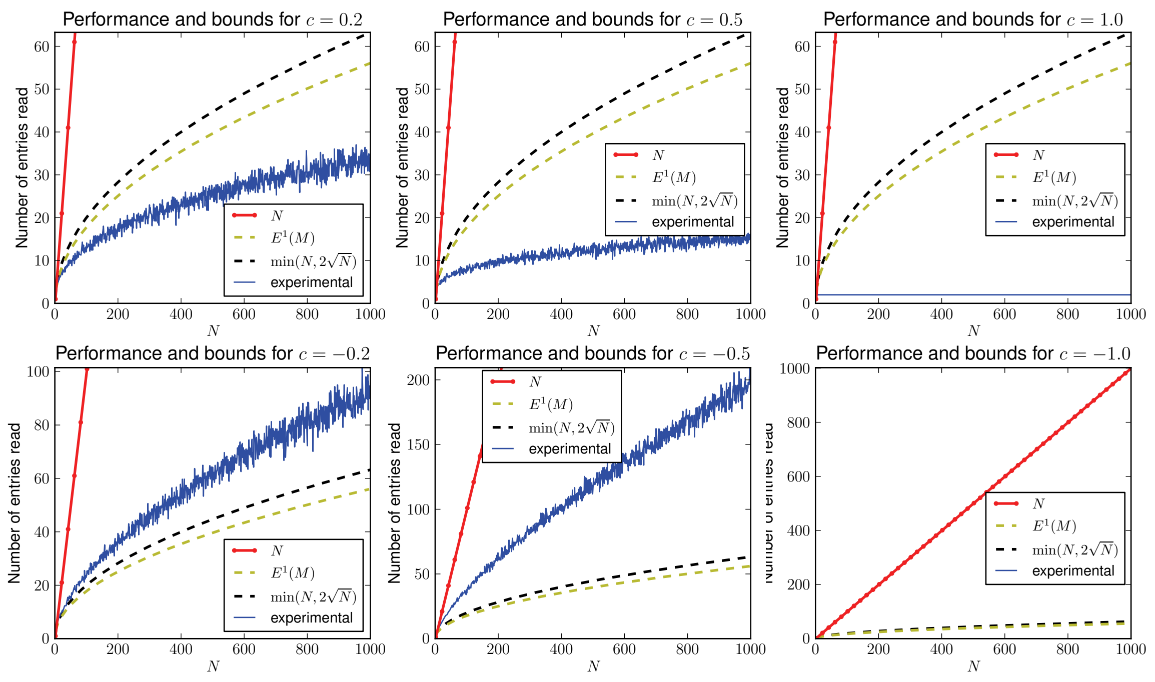


Figure 12: Performance of our algorithm for different correlation coefficients. The top three plots show positive correlation, the bottom three show negative correlation. Correlation coefficients of $c = 1.0$ and $c = -1.0$ capture precisely the best and worst-case performance of our algorithm, resulting in $O(1)$ and $\Theta(N)$ performance, respectively (when $c = -1.0$ the linear curve obscures the experimental curve).

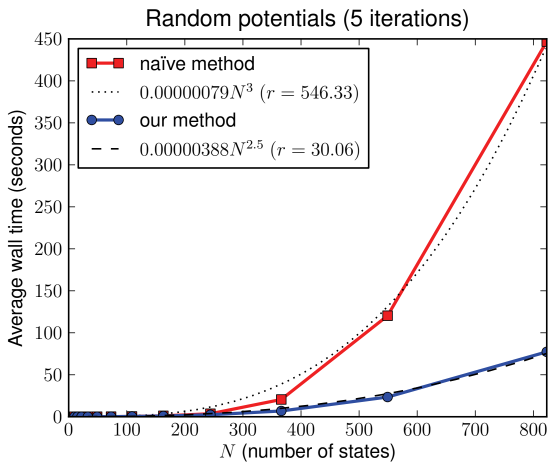
We perform matching between a *template* graph with M nodes, and a *target* graph with N nodes, which requires a graphical model with M nodes and N states per node (see McAuley et al. 2008 for details). We fix $M = 10$ and vary N.

Figure 13 (left) shows the performance on random potentials, that is, the performance we hope to obtain if our model assumptions are satisfied. Figure 13 (right) shows the performance for graph-matching, which closely matches the expected-case behavior. Fitted curves are shown together with the actual running time of our algorithm, confirming its $O(MN^2\sqrt{N})$ performance. The coefficients of the fitted curves demonstrate that our algorithm is useful even for modest values of N.

We also report results for graph-matching using graphs from the MPEG-7 data set (Bai et al., 2009), which consists of 1,400 silhouette images (Figure 14). Again we fix $M = 10$ (i.e., 10 points are extracted in each template graph) and vary N (the number of points in the target graph). This experiment confirms that even when matching real-world graphs, the assumption of independent order statistics appears to be reasonable.

7.3.2 HIGHER-ORDER MARKOV MODELS

In this experiment, we construct a simple Markov model for text denoising. Random noise is applied to a text segment, which we try to correct using a prior extracted from a text corpus. For instance



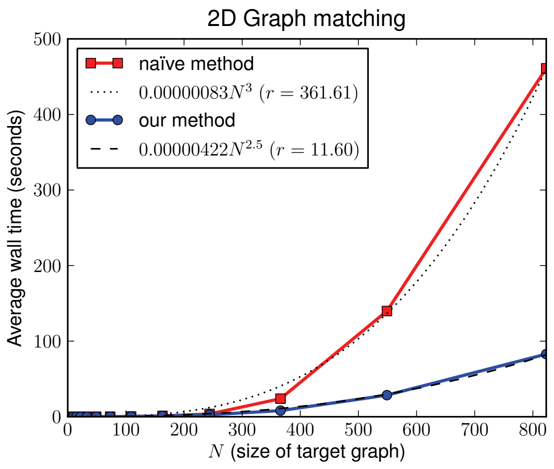


Figure 13: The running time of our method on randomly generated potentials, and on a graph-matching experiment (both graphs have the same topology). Fitted curves are also obtained by performing least-squares regression; the residual error *r* indicates the 'goodness' of the fitted curve.

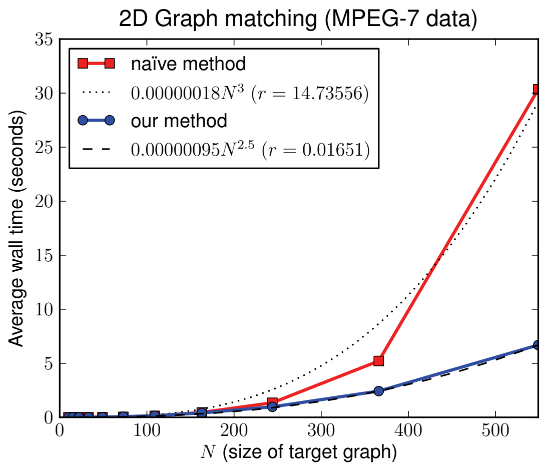


Figure 14: The running time of method our on graphs from the MPEG-7 data set.

`wondrous sight of th4 ivory Pequod` is corrected to `wondrous sight of the ivory Pequod`.

In such a model, we would like to exploit higher-order relationships between characters, though the amount of data required to construct an accurate prior grows exponentially with the size of the maximal cliques. Instead, our prior consists entirely of pairwise relationships between characters (or 'bigrams'); higher-order relationships are encoded by including bigrams of non-adjacent characters.

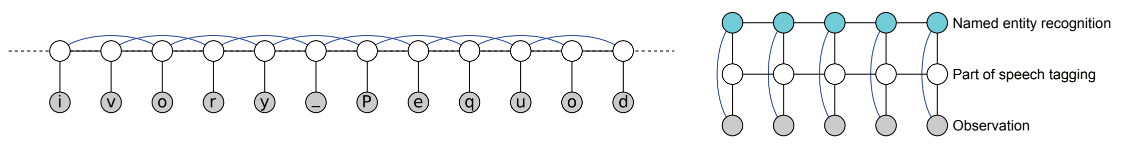


Figure 15: Left: Our model for denoising. Its computational complexity is similar to that of a skip-chain CRF, and models for named-entity recognition (right).

Specifically, our model takes the form

$$\Phi_X(\mathbf{x}_X) = \prod_{i=1}^{|X|-1} \Phi_{i,i+1}(x_i, x_{i+1}) \times \prod_{i=1}^{|X|-2} \Phi_{i,i+2}(x_i, x_{i+2})$$

where

$$\Phi_{i,j}(x_i, x_j) = \psi_{i,j}(x_i, x_j) p(x_i|o_i) p(x_j|o_j).$$

Here ψ is our *prior* (extracted from text statistics), and p is our 'noise model' (given the observation $\mathbf{o}$). The computational complexity of inference in this model is similar to that of the skip-chain CRF shown in Figure 3(b), as well as models for part-of-speech tagging and named-entity recognition, as in Figure 15. Text denoising is useful for the purpose of demonstrating our algorithm, as there are several different corpora available in different languages, allowing us to explore the effect that the domain size (i.e., the size of the language's alphabet) has on running time.

We extracted pairwise statistics based on 10,000 characters of text, and used this to correct a series of 25 character sequences, with 1% random noise introduced to the text. The domain was simply the set of characters observed in each corpus. The Japanese data set was not included, as the $\Theta(MN^2)$ memory requirements of the algorithm made it infeasible with $N \simeq 2000$; this is addressed in Section 7.4.1.

The running time of our method, compared to the naïve solution, is shown in Figure 16. One might expect that texts from different languages would exhibit different dependence structures in their order statistics, and therefore deviate from the expected case in some instances. However, the running times appear to follow the fitted curve closely, that is, we are achieving approximately the expected-case performance in all cases.

Since the prior $\psi_{i,i+1}(x_i, x_{i+1})$ is *data-independent*, we shall further discuss this type of model in reference to Algorithm 3 in Section 7.4.

7.4 Experiments with Conditionally Factorizable Models

In each of the following experiments we perform belief-propagation in models of the form given in (Equation 7). Thus each model is completely specified by defining the node potentials $\Phi_i(x_i|y_i)$, the edge potentials $\Phi_{i,j}(x_i, x_j)$, and the topology $(\mathcal{N}, \mathcal{E})$ of the graph.

Furthermore we assume that the edge potentials are *homogeneous*, that is, that the potential for each edge is the same, or rather that they have the same order statistics (for example, they may differ by a multiplicative constant). This means that sorting can be done *online* without affecting the asymptotic complexity. When subject to heterogeneous potentials we need merely sort them *offline*; the online cost shall be similar to what we report here.

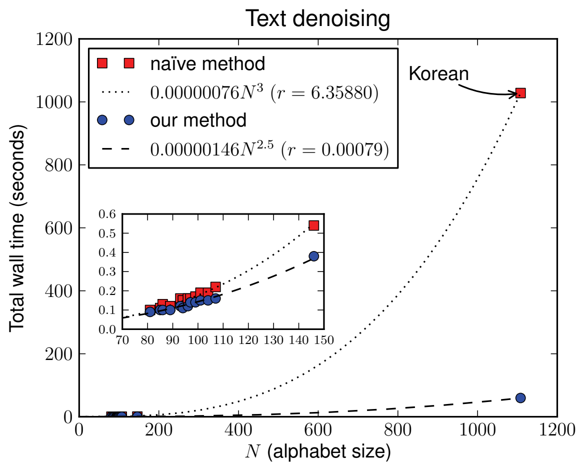


Figure 16: The running time of our method compared to the naïve solution. A fitted curve is also shown, whose coefficient estimates the computational overhead of our model.

7.4.1 Chain-Structured Models

In this section, we consider *chain-structured* graphs. Here we have nodes $\mathcal{N} = \{1 \dots Q\}$, and edges $\mathcal{E} = \{(1,2),(2,3)\dots(Q-1,Q)\}$. The max-product algorithm is known to compute the maximum-likelihood solution exactly for tree-structured models.

Figure 17 (left) shows the performance of our method on a model with *random* potentials, that is, $\Phi_i(x_i|y_i) = U[0,1)$, $\Phi_{i,i+1}(x_i,x_{i+1}) = U[0,1)$, where $U[0,1)$ is the uniform distribution. Fitted curves are superimposed onto the running time, confirming that the performance of the standard solution grows quadratically with the number of states, while ours grows at a rate of $N\sqrt{N}$. The residual error r shows how closely the fitted curve approximates the running time; in the case of random potentials, both curves have almost the same constant.

Figure 17 (right) shows the performance of our method on the text denoising experiment. This experiment is essentially identical to that shown in Section 7.3.2, except that the model is a chain (i.e., there is no $\Phi_{i,i+2}$), and we exploit the notion of data-independence (i.e., the fact that $\Phi_{i,i+1}$ does not depend on the observation). Since the same $\Phi_{i,i+1}$ is used for every adjacent pair of nodes, there is no need to perform the ‘sorting’ step offline—only a single copy of $\Phi_{i,i+1}$ needs to be sorted, and this is included in the total running time shown in Figure 17.

7.4.2 Grid-Structured Models

Similarly, we can apply our method to *grid-structured* models. Here we resort to loopy belief-propagation to approximate the MAP solution, though indeed the same analysis applies in the case of factor-graphs (Kschischang et al., 2001). We construct a 50×50 grid model and perform loopy belief-propagation using a random message-passing schedule for five iterations. In these experiments our nodes are $\mathcal{N} = \{1 \dots 50\}^2$, and our edges connect the 4-neighbors, that is, the node (i, j) is connected to both $(i+1, j)$ and $(i, j+1)$ (similar to the grid shown in Figure 2(a)).

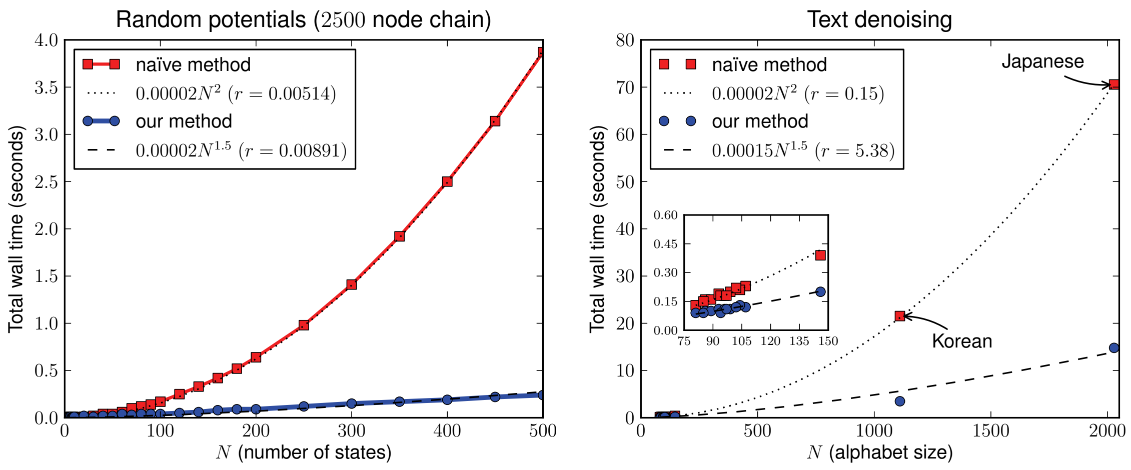


Figure 17: Running time of inference in chain-structured models: random potentials (left), and text denoising (right). Fitted curves confirm that the exponent of 1.5 given theoretically is maintained in practice (r denotes the sum of residuals, that is, the ‘goodness’ of the fitted curve).

Figure 18 (left) shows the performance of our method on a grid with random potentials (similar to the experiment in Section 7.4.1). Figure 18 (right) shows the performance of our method on an optical flow task (Lucas and Kanade, 1981). Here the states encode *flow vectors*: for a node with N states, the flow vector is assumed to take integer coordinates in the square $[-\sqrt{N}/2, \sqrt{N}/2)^2$ (so that there are N possible flow vectors). For the unary potential we have

$$\Phi_{(i,j)}(x|y) = \left\|Im_1[i,j] - Im_2[(i,j)+f(x)]\right\|,$$

where $Im_1[a,b]$ and $Im_2[a,b]$ return the gray-level of the pixel at (a,b) in the first and second images (respectively), and $f(x)$ returns the flow vector encoded by x. The pairwise potentials simply encode the Euclidean distance between two flow vectors. Note that a variety of low-level computer vision tasks (including optical flow) are studied in Felzenszwalb and Huttenlocher (2006), where the highly structured nature of the potentials in question often allows for efficient solutions.

Our fitted curves in Figure 18 show $O(N\sqrt{N})$ performance for both random data and for optical flow. Clearly the fitted curve for optical flow deviates somewhat from that obtained for random data; naturally the potentials are highly structured in this case, as exploited by Felzenszwalb and Huttenlocher (2006); it appears that some aspect of this structure is slightly harmful to our algorithm, though a more thorough analysis of this type of potential remains as future work. More ‘harmful’ structures are explored in the following section.

7.4.3 FAILURE CASES

In our previous experiments on graph-matching, text denoising, and optical flow we observed running times similar to those for random potentials, indicating that there is no prevalent dependence structure between the order statistics of the messages and the potentials.

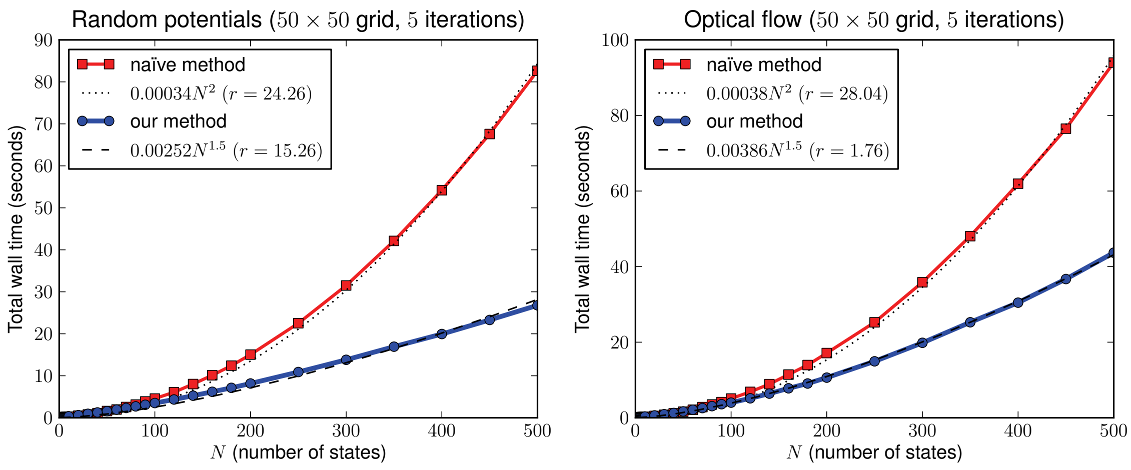


Figure 18: Running time of inference in grid-structured models: random potentials (left), and optical flow (right).

In certain applications the order statistics of these terms are highly dependent in a way that is detrimental to our algorithm. This behavior is observed for certain types of concave potentials (or convex potentials in a min-sum formulation). For instance, in a stereo disparity experiment, the unary potentials encode the fact that the output should be 'close to' a certain value; the pairwise potentials encode the fact that neighboring nodes should take similar values (Scharstein and Szeliski, 2001; Sun et al., 2003).

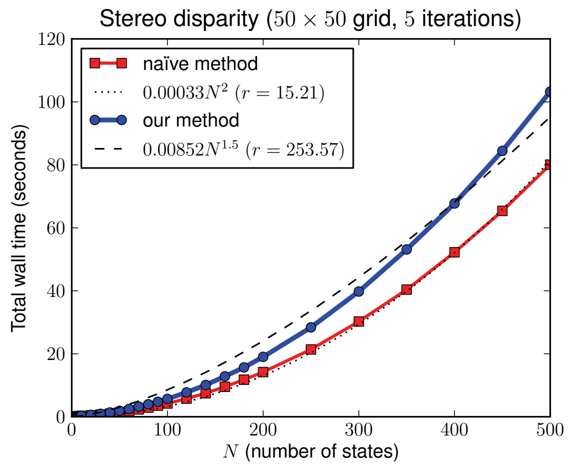
In these applications, the permutation matrices that transform the sorted values of $\mathbf{v}_a$ to the sorted values of $\mathbf{v}_b$ are block-off-diagonal (see the sixth permutation in Figure 11). In such cases, our algorithm only decreases the number of multiplication operations by a multiplicative constant, and may in fact be slower due to its computational overhead. This is precisely the behavior shown in Figure 19 (left), in the case of stereo disparity.

It should be noted that there exist algorithms specifically designed for this class of potential functions (Kolmogorov and Shioura, 2007; Felzenszwalb and Huttenlocher, 2006), which are preferable in such instances.

We similarly perform an experiment on image denoising, where the unary potentials are again convex functions of the input (see Geman and Geman, 1984; Lan et al., 2006). Instead of using a pairwise potential that merely encodes smoothness, we extract the pairwise statistics from image data (similar to our experiment on text denoising); thus the potentials are no longer concave. We see in Figure 19 (right) that even if a small number of entries exhibit some 'randomness' in their order statistics, we begin to gain a modest speed improvement over the naïve solution (though indeed, the improvements are negligible compared to those shown in previous experiments).

7.5 Other Applications of Tropical Matrix Multiplication

As we have mentioned, our improvements to message-passing in graphical models arise from a fast solution to matrix multiplication in the max-product semiring. In this section we discuss other



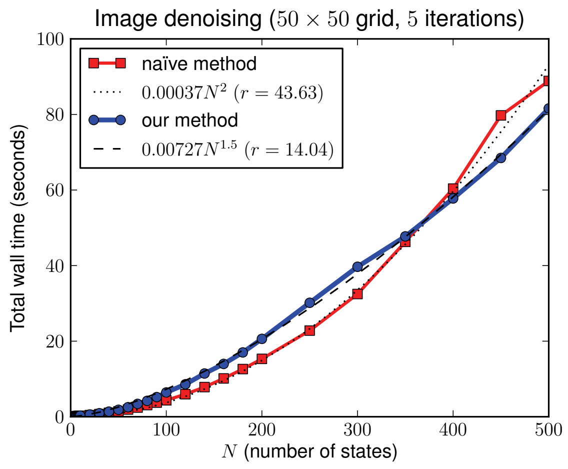


Figure 19: Two experiments whose potentials and messages have highly dependent order statistics: stereo disparity (left), and image denoising (right).

problems which include max-product (or 'tropical') matrix multiplication as a subroutine. Williams and Williams (2010) discusses the relationship between this type of matrix multiplication problem and various other problems.

7.5.1 MAX-PRODUCT LINEAR PROGRAMMING

In Sontag et al. (2008), a method is given for exact MAP-inference in graphical models using LP-relaxations. Where exact solutions cannot be obtained by considering only pairwise factors, 'clusters' of pairwise terms are introduced in order to refine the solution. Message-passing in these clusters turns out to take exactly the form that we consider, as third-order (or larger) clusters are formed from pairwise terms. Although a number of applications are presented in Sontag et al. (2008), we focus on protein design, as this is the application in which we typically observe the largest domain sizes. Other applications with larger domains may yield further benefits.

Without going into detail, we simply copy the two equations from Sontag et al. (2008) to which our algorithm applies. The first of these is concerned with passing messages between clusters, while the second is concerned with choosing new clusters to add. Below are the two equations, reproduced verbatim from Sontag et al. (2008):

$$\lambda_{c\to e}(x_e) \leftarrow -\frac{2}{3}\Big(\lambda_{e\to e}(x_e) + \sum_{c'\neq c, e\in c'} \lambda_{c'\to e}(x_e)\Big) + \frac{1}{3}\max_{x_{c\setminus e}}\Big[\sum_{e'\in c\setminus e}\Big(\lambda_{e'\to e'}(x_{e'}) + \sum_{c'\neq c, e'\in c'} \lambda_{c'\to e'}(x_{e'})\Big)\Big] \tag{15}$$

(see Sontag et al., 2008, Figure 1, bottom), which consists of marginalizing a cluster (c) that decomposes into edges (e), and

$$d(c) = \sum_{e\in c}\max_{x_e} b_e(x_e) - \max_{x_c}\left[\sum_{e\in c} b_e(x_e)\right], \tag{16}$$

(see Sontag et al., 2008, (Equation 4)), which consists of finding the MAP-state in a ring-structured model.

As the code from Sontag et al. (2008) was publicly available, we simply replaced the appropriate functions with our own (in order to provide a fair comparison, we also replaced their implementation of the naïve algorithm, as ours proved to be faster than the highly generic matrix library used in their code).

In order to improve the running time of our algorithm, we made the following two modifications to Algorithm 2:

- We used an *adaptive sorting algorithm* (i.e., a sorting algorithm that runs faster on nearly-sorted data). While Quicksort was used during the first iteration of message-passing, subsequent iterations used insertion sort, as the optimal ordering did not change significantly between iterations.

- We added an additional stopping criterion to the algorithm. Namely, we terminate the algorithm if $\mathbf{v}_a[p_a[start]] \times \mathbf{v}_b[p_b[start]] < max$. In other words, we check how large the maximum *could be* given the best possible permutation of the next elements (i.e., if they have the same index); if this value could not result in a new maximum, the algorithm terminates. This check costs us an additional multiplication, but it means that the algorithm will terminate faster in cases where a large maximum is found early on.

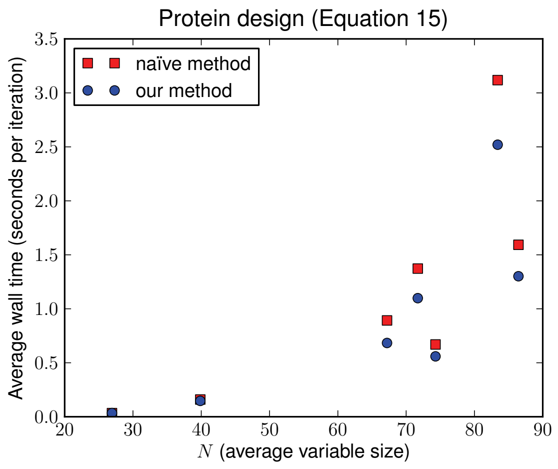
Results for these two problems are shown in Figure 20. Although our algorithm consistently improves upon the running time of Sontag et al. (2008), the domain size of the variables in question is not typically large enough to see a marked improvement. Interestingly, neither method follows the expected running time closely in this experiment. This is partly due to the fact that there is significant variation in the variable size (note that N only shows the *average* variable size), but it may also suggest that there is a complicated structure in the potentials which violates our assumption of independent order statistics.

7.5.2 ALL-PAIRS SHORTEST-PATH

The ‘all-pairs shortest-path’ problem consists of finding the shortest path between every pair of nodes in a graph. Although the most commonly used solution is probably the well-known Floyd-Warshall algorithm (Floyd, 1962), the state-of-the-art *expected-case* solution to this problem is that of Karger et al. (1993), whose expected-case running time is $O(N^2 \log N)$ when applied to graphs with distances sampled from the uniform distribution.

Unfortunately, the solution of Karger et al. (1993) requires a Fibonacci heap or similar data structure in order to achieve the reported running time (i.e., a heap with $O(1)$ insertion and decrease-key operations); such data structures are known to be inefficient in practice (Fredman and Tarjan, 1987). When their algorithm is implemented using a standard priority queue, it has running time $O(N^2 \log^2 N)$.

In Aho et al. (1983), a transformation is shown between the all-pairs shortest-path problem and min-sum matrix multiplication. Using our algorithm, this gives us an expected-case $O(N^2\sqrt{N})$ solution to the all-pairs shortest-path problem, assuming that the subproblems created by this transformation have i.i.d. order statistics; this assumption is notably different than the assumption of uniformity made in Karger et al. (1993).



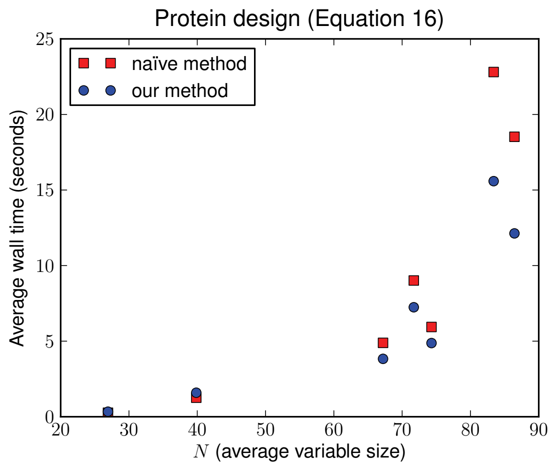


Figure 20: The running time of our method on protein design problems from Sontag et al. (2008). In this figure, N reflects the *average* domain size amongst all variables involved in the problem; fitted curves are not shown due to the highly variable nature of the domain sizes included in each problem instance.

In Figure 21, we show the performance of our method on i.i.d. uniform graphs, compared to the Floyd-Warshall algorithm, and that of Karger et al. (1993). On graph sizes of practical interest, our algorithm is found to give the fastest performance, in spite of its more expensive asymptotic cost. Our solution is comparable to that of Karger et al. (1993) for the largest graph size shown; larger graph sizes could not be shown due to memory constraints. Note that while these algorithms are fast in practice, each has $\Theta(N^3)$ *worst-case* performance; more 'exotic' solutions that improve upon the worst-case bound are discussed in Alon et al. (1997) and Chan (2007), among others, though none are truly subcubic (i.e., $O(N^{3-\varepsilon})$).

It should also be noted that the transformations given in Aho et al. (1983) apply in both directions, that is, solutions to the all-pairs shortest-path problem can be used to solve min-sum matrix multiplication. Thus any subcubic solution to the all-pairs shortest-path problem can be applied to the inference problems in graphical models presented in Section 4. However, the transformation of Aho et al. (1983) introduces a very high computational overhead (namely, solving min-sum matrix multiplication for an $N \times N$ matrix requires solving all-pairs shortest-path in a graph with $3N$ nodes), and moreover it violates the assumptions on the graph distribution required for fast inference given in Karger et al. (1993). In practice, we were unable to produce an implementation of min-sum matrix multiplication based on this transformation that was faster than the naïve solution.

Interestingly, a great deal of attention has been focused on expected-case solutions to all-pairs shortest-path, while to our knowledge ours is the first work to approach the expected-case analysis of min-sum matrix multiplication. Given the strong relationship between the two problems, it remains a promising open problem to assess whether the analysis from these solutions to all-pairs shortest-path can be applied to produce max-product matrix multiplication algorithms with similar asymptotic running times.

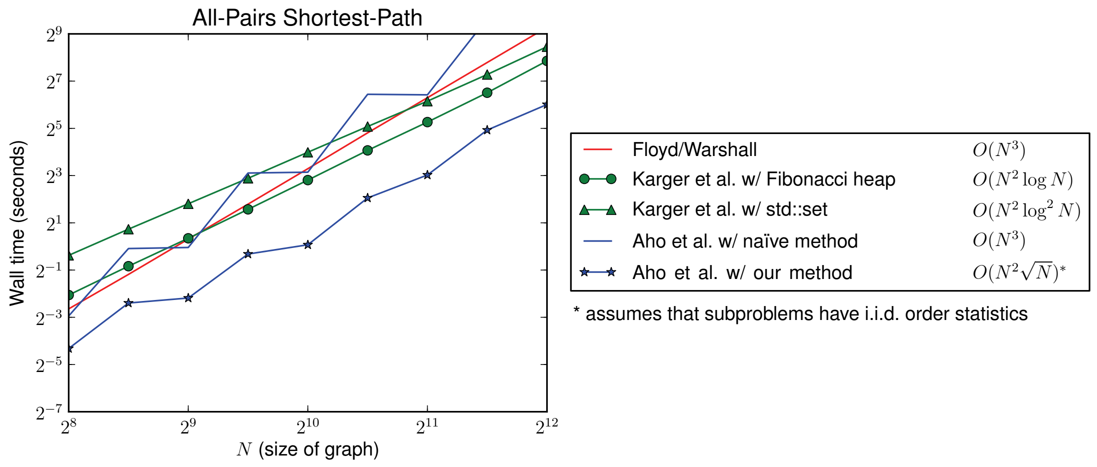


Figure 21: Our algorithm applied to the 'all-pairs shortest-path' problem. The expected-case running times of each algorithm are shown at right.

7.5.3 L^∞ DISTANCES

The problem of computing an inner product in the max-sum semiring is closely related to computing the L^∞ distance between two vectors

$$||\mathbf{v}_a - \mathbf{v}_b||_\infty = \max_{i \in \{1 \dots N\}} \left|\mathbf{v}_a[i] - \mathbf{v}_b[i]\right|. \tag{17}$$

Naïvely, we would like to solve (Equation 17) by applying Algorithm 2 to $\mathbf{v}_a$ and $-\mathbf{v}_b$ with the multiplication operator replaced by $a \times b = |a+b|$, however this violates the condition of (Equation 2), since the optimal solution may arise either when both $\mathbf{v}_a[i]$ and $-\mathbf{v}_b[i]$ are large, or when both $\mathbf{v}_a[i]$ and $-\mathbf{v}_b[i]$ are small (in fact, this operation violates the semiring axiom of associativity).

We address this issue by running Algorithm 2 *twice*, first considering the *largest* values of $\mathbf{v}_a$ and $-\mathbf{v}_b$, before re-running the algorithm starting from the *smallest* values. This ensures that the maximum solution is found in either case.

Pseudocode for this solution is given in Algorithm 7, which adapts Algorithm 4 to the problem of computing an L^∞ distance matrix. Similarly, we can adapt Algorithm 3 to solve L^∞ nearest-neighbor problems, where an array of M points in $\mathbb{R}^N$ is processed offline, allowing us compute the distance of a query point to all M other points $O(M\sqrt{N})$.

Figure 22 shows the running time of our algorithm for computing an L^∞ distance matrix (where $M = N$), and the online cost of performing a nearest-neighbor query. Again the expected speedup over the naïve solution is $\Omega(\sqrt{N})$ for both problems, though naturally our algorithm requires larger values of N than does Algorithm 4 in order to be beneficial, since Algorithm 2 must be executed *twice* in order to solve (Equation 17).

A similar trick can be applied to compute message in the max-product semiring even for potentials that contain negative values, though this may require up to four executions of Algorithm 2, so it is unlikely to be practical.

Algorithm 7 Use Algorithm 2 to compute an L^∞ distance matrix

```
Input: an $M \times N$ array $\mathbf{A}$ containing $M$ points in $\mathbb{R}^N$
 1: initialize an $M \times M$ distance matrix $\mathbf{D} := \mathbf{0}$
 2: for $x \in \{1 \dots M\}$ do
 3:   compute $\overrightarrow{\mathbf{P}}[x]$ by sorting $\mathbf{A}[x]$ {takes $\Theta N \log N$}
 4:   compute $\overleftarrow{\mathbf{P}}[x]$ by sorting $-\mathbf{A}[x]$ {i.e., $\overrightarrow{\mathbf{P}}[x]$ in reverse order}
 5: end for {this loop takes $\Theta(MN \log N)$}
 6: for $x \in \{1 \dots M\}$ do
 7:   for $y \in \{x+1 \dots M\}$ do
 8:     $best_1 := Algorithm2\left(\mathbf{A}[x], -\mathbf{A}[y], \overrightarrow{\mathbf{P}}[x], \overleftarrow{\mathbf{P}}[y]\right)$
        {takes $O(\sqrt{N})$; Algorithm 2 uses the operator $a \times b = |a+b|$}
 9:     $best_2 := Algorithm2\left(\mathbf{A}[y], -\mathbf{A}[x], \overrightarrow{\mathbf{P}}[y], \overleftarrow{\mathbf{P}}[x]\right)$
10:     $\mathbf{D}[x,y] := \max\left(\left|\mathbf{A}[x, best_1] - \mathbf{A}[y, best_1]\right|, \left|\mathbf{A}[x, best_2] - \mathbf{A}[y, best_2]\right|\right)$
11:     $\mathbf{D}[y,x] := \mathbf{D}[x,y]$
12:   end for
13: end for {this loop takes expected time $O(M^2\sqrt{N})$}
```

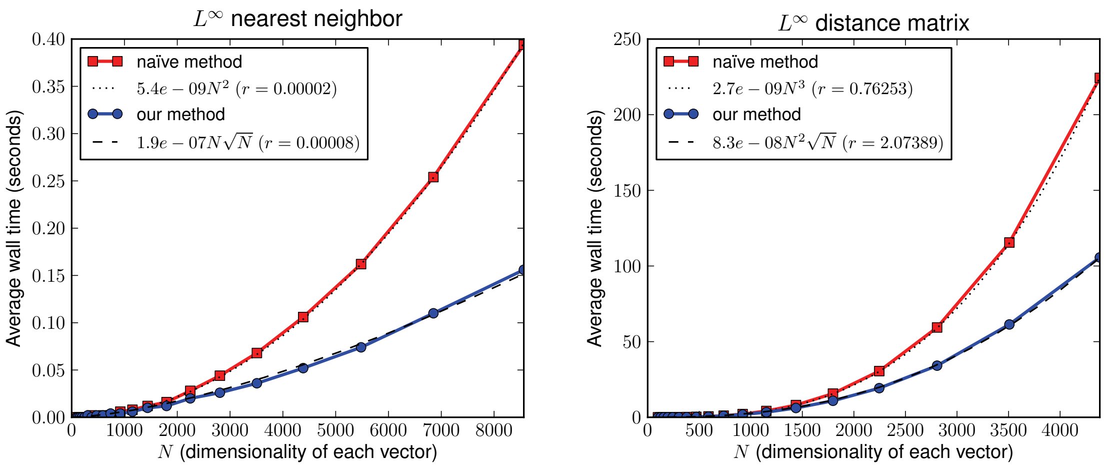


Figure 22: The running time of our method compared to the naïve solution. A fitted curve is also shown, whose coefficient estimates the computational overhead of our model.

8. Discussion and Future Work

We have briefly discussed the application of our algorithm to the all-pairs shortest-path problem, and also mentioned that a variety of other problems are related to max-product matrix multiplication via a series of subcubic transformations (Williams and Williams, 2010). To our knowledge, of all these problems only all-pairs shortest-paths has received significant attention in terms of expected-case analysis. The analysis in question centers around two types of model: the *uniform* model, where edge weights are sampled from a uniform distribution, and the *endpoint-independent model*, which essentially makes an assumption on the independence of outgoing edge weights from each vertex (Moffat and Takaoka, 1987), which seems very similar to our assumption of independent order statistics. It remains to be seen whether this analysis can lead to better solutions to the problems discussed here, and indeed if the analysis applied to uniform models can be applied in our setting to uniform *matrices*.

It is interesting to consider the fact that our algorithm's running time is purely a function of the input data's *order statistics*, and in fact does not depend on the *data itself*. While it is pleasing that our assumption of independent order statistics appears to be a weak one, and is satisfied in a wide variety of applications, it ignores the fact that stronger assumptions may be reasonable in many cases. In factors with a high dynamic range, or when different factors have different scales, it may be possible to identify the maximum value very quickly, as we attempted to do in Section 7.5.1. Deriving faster algorithms that make stronger assumptions about the input data remains a promising avenue for future work.

Our algorithm may also lead to faster solutions for *approximately* passing a single message. While the stopping criterion of our algorithm *guarantees* that the maximum value is found, it is possible to terminate the algorithm earlier and state that the maximum has *probably* been found. A direction for future work would be to adapt our algorithm to determine the probability that the maximum has been found after a certain number of steps; we could then allow the user to specify an error probability, or a desired running time, and our algorithm could be adapted accordingly.

9. Conclusion

We have presented a series of approaches that allow us to improve the performance of exact and approximate max-product message-passing for models with factors smaller than their maximal cliques, and more generally, for models whose factors *that depend upon the observation* contain fewer latent variables than their maximal cliques. We are *always* able to improve the expected computational complexity in any model that exhibits this type of factorization, no matter the size or number of factors.

Acknowledgments

We would like to thank Pedro Felzenszwalb, Johnicholas Hines, and David Sontag for comments on initial versions of this paper, and James Petterson and Roslyn Lau for helpful discussions. NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative, and the Australian Research Council's *ICT Centre of Excellence* program.

Appendix A. Asymptotic Performance of Algorithm 2 and Extensions

In this section we shall determine the expected-case running times of Algorithm 2 and Algorithm 6. Algorithm 2 traverses $\mathbf{v}_a$ and $\mathbf{v}_b$ until it reaches the smallest value of m for which there is some $j \leq m$ for which $m \geq p_b^{-1}[p_a[j]]$. If M is a random variable representing this smallest value of m, then we wish to find $E(M)$. While $E(M)$ is the number of 'steps' the algorithms take, each step takes $\Theta(K)$ when we have K lists. Thus the expected running time is $\Theta(KE(M))$.

To aid understanding our algorithm, we show the elements being read for specific examples of $\mathbf{v}_a$ and $\mathbf{v}_b$ in Figure 23. This figure reveals that the actual *values* in $\mathbf{v}_a$ and $\mathbf{v}_b$ are unimportant, and

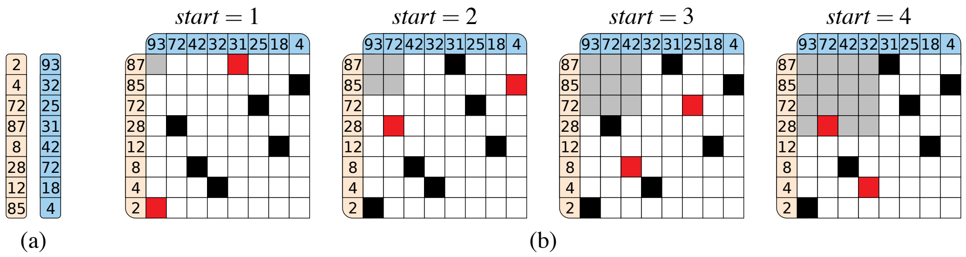


Figure 23: (a) The lists $\mathbf{v}_a$ and $\mathbf{v}_b$ before sorting; (b) Black squares show corresponding elements in the sorted lists ($\mathbf{v}_a[p_a[i]]$ and $\mathbf{v}_b[p_b[i]]$); red squares indicate the elements read during each step of the algorithm ($\mathbf{v}_a[p_a[start]]$ and $\mathbf{v}_b[p_b[start]]$). We can imagine expanding a gray box of size $start \times start$ until it contains an entry; note that the maximum is found during the first step.

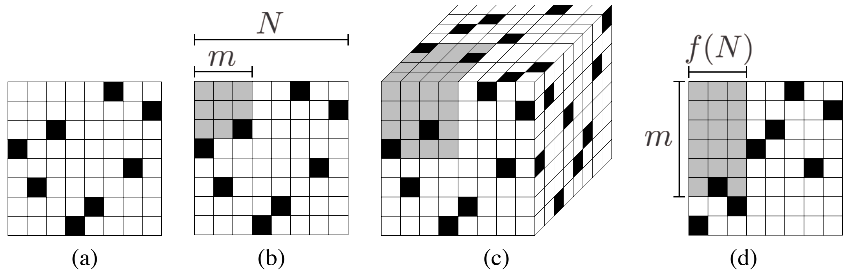


Figure 24: (a) As noted in Figure 23, a permutation can be represented as an array, where there is exactly one non-zero entry in each row and column; (b) We want to find the smallest value of m such that the gray box includes a non-zero entry; (c) A *pair* of permutations can be thought of as a cube, where every two-dimensional plane contains exactly one non-zero entry; we are now searching for the smallest gray cube that includes a non-zero entry; the faces show the projections of the points onto the exterior of the cube (the third face is determined by the first two); (d) For the sake of establishing an upper-bound, we consider a shaded region of width $f(N)$ and height m.

it is only the order statistics of the two lists that determine the performance of our algorithm. By representing a permutation of the digits 1 to N as shown in Figure 24 ((a), (b), and (d)), we observe that m is simply the width of the smallest square (expanding from the top left) that includes an element of the permutation (i.e., it includes i and $p[i]$).

Simple analysis reveals that the probability of choosing a permutation that does not contain a value inside a square of size m is

$$P(M > m) = \frac{(N-m)!(N-m)!}{(N-2m)!N!}. \tag{18}$$

This is precisely $1 - F(m)$, where $F(m)$ is the cumulative density function of M. It is immediately clear that $1 \leq M \leq \lfloor N/2 \rfloor$, which defines the best and worst-case performance of Algorithm 2.

Using the identity $E(X) = \sum_{x=1}^{\infty} P(X \geq x)$, we can write down a formula for the expected value of M:

$$E(M) = \sum_{m=0}^{\lfloor N/2 \rfloor} \frac{(N-m)!(N-m)!}{(N-2m)!N!}. \tag{19}$$

The case where we are sampling from multiple permutations simultaneously (i.e., Algorithm 6) is analogous. We consider $K-1$ permutations embedded in a K-dimensional hypercube, and we wish to find the width of the smallest shaded hypercube that includes exactly one element of the permutations (i.e., $i, p_1[i], \dots, p_{K-1}[i]$). This is represented in Figure 24(c) for $K=3$. Note carefully that K is the number of *lists* in (Equation 14); if we have K lists, we require $K-1$ permutations to define a correspondence between them.

Unfortunately, the probability that there is no non-zero entry in a cube of size m^K is not trivial to compute. It is possible to write down an expression that generalizes (Equation 18), such as

$$P^K(M > m) = \frac{1}{N!^{K-1}} \times \sum_{\sigma_1 \in S_N} \cdots \sum_{\sigma_{K-1} \in S_N} \bigwedge_{i=1}^{m} \left(\max_{k \in \{1 \dots K-1\}} \sigma_k(i) > m \right) \tag{20}$$

(in which we simply enumerate over all possible permutations and 'count' which of them do not fall within a hypercube of size m^K), and therefore state that

$$E^K(M) = \sum_{m=0}^{\infty} P^K(M > m). \tag{21}$$

However, it is very hard to draw any conclusions from (Equation 20), and in fact it is intractable even to evaluate it for large values of N and K. Hence we shall instead focus our attention on finding an upper-bound on (Equation 21). Finding more computationally convenient expressions for (Equation 20) and (Equation 21) remains as future work.

A.1 An Upper-Bound on $E^K(M)$

Although (Equation 19) and (Equation 21) precisely define the running times of Algorithm 2 and Algorithm 6, it is not easy to ascertain the speed improvements they achieve, as the values to which the summations converge for large N are not obvious. Here, we shall try to obtain an upper-bound on their performance, which we assessed experimentally in Section 7. In doing so we shall prove Theorems 2 and 3.

Proof [Proof of Theorem 2] (see Algorithm 2) Consider the shaded region in Figure 24(d). This region has a width of $f(N)$, and its height m is chosen such that it contains precisely one non-zero entry. Let $\dot{M}$ be a random variable representing the height of the gray region needed in order to include a non-zero entry. We note that

$$E(\dot{M}) \in O(f(N)) \ \Rightarrow \ E(M) \in O(f(N));$$

our aim is to find the smallest $f(N)$ such that $E(\dot{M}) \in O(f(N))$. The probability that none of the first m samples appear in the shaded region is

$$P(\dot{M} > m) = \prod_{i=0}^{m} \left(1 - \frac{f(N)}{N-i} \right).$$

Next we observe that if the entries in our $N \times N$ grid do not define a permutation, but we instead choose a *random* entry in each row, then the probability (now for $\ddot{M}$) becomes

$$P(\ddot{M} > m) = \left(1 - \frac{f(N)}{N}\right)^m \tag{22}$$

(for simplicity we allow m to take arbitrarily large values). We certainly have that $P(\ddot{M} > m) \geq P(\dot{M} > m)$, meaning that $E(\ddot{M})$ is an upper-bound on $E(\dot{M})$, and therefore on $E(M)$. Thus we compute the expected value

$$E(\ddot{M}) = \sum_{m=0}^{\infty} \left(1 - \frac{f(N)}{N}\right)^m.$$

This is just a geometric progression, which sums to $N/f(N)$. Thus we need to find $f(N)$ such that

$$f(N) \in O\left(\frac{N}{f(N)}\right).$$

Clearly $f(N) \in O(\sqrt{N})$ will do. Thus we conclude that

$$E(M) \in O(\sqrt{N}).$$

■

Proof [Proof of Theorem 3] (see Algorithm 6) We would like to apply the same reasoning in the case of multiple permutations in order to compute a bound on $E^K(M)$. That is, we would like to consider $K-1$ *random* samples of the digits from 1 to N, rather than $K-1$ permutations, as random samples are easier to work with in practice.

To do so, we begin with some simple corollaries regarding our previous results. We have shown that in a permutation of length N, we expect to see a value less than or equal to f after N/f steps. There are now $f-1$ other values that are less than or equal to f amongst the remaining $N - N/f$ values; we note that

$$\frac{f-1}{N - \frac{N}{f}} = \frac{f}{N}.$$

Hence we expect to see the *next* value less than or equal to f in the next N/f steps also. A consequence of this fact is that we not only expect to see the *first* value less than or equal to f earlier in a permutation than in a random sample, but that when we sample m elements, we expect *more* of them to be less than or equal to f in a permutation than in a random sample.

Furthermore, when considering the *maximum* of $K-1$ permutations, we expect the first m elements to contain more values less than or equal to f than the maximum of $K-1$ random samples. (Equation 20) is concerned with precisely this problem. Therefore, when working in a K-dimensional hypercube, we can consider $K-1$ random samples rather than $K-1$ permutations in order to obtain an upper-bound on (Equation 21).

Thus we define $\ddot{M}$ as in (Equation 22), and conclude that

$$P(\ddot{M} > m) = \left(1 - \frac{f(N,K)^{K-1}}{N^{K-1}}\right)^m.$$

Thus the expected value of $\ddot{M}$ is again a geometric progression, which this time sums to $(N/f(N,K))^{K-1}$. Thus we need to find $f(N,K)$ such that

$$f(N,K) \in O\left(\left(\frac{N}{f(N,K)}\right)^{K-1}\right).$$

Clearly

$$f(N,K) \in O\left(N^{\frac{K-1}{K}}\right)$$

will do. As mentioned, each step takes $\Theta(K)$, so the final running time is $O(KN^{\frac{K-1}{K}})$. ■

To summarize, for problems decomposable into $K+1$ groups, we will need to find the index that chooses the maximal product amongst K lists; we have shown an upper-bound on the expected number of steps this takes, namely

$$E^K(M) \in O\left(N^{\frac{K-1}{K}}\right). \tag{23}$$

References

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997.

Xiang Bai, Xingwei Yang, Longin Jan Latecki, Wenyu Liu, and Zhuowen Tu. Learning context-sensitive shape similarity by graph transduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):861–874, 2009.

Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *Annual ACM Symposium on Theory of Computing*, pages 590–598, 2007.

James M. Coughlan and Sabino J. Ferreira. Finding deformable shapes using loopy belief propagation. In *ECCV*, 2002.

René Donner, Georg Langs, and Horst Bischof. Sparse MRF appearance models for fast anatomical structure localisation. In *BMVC*, 2007.

Gal Elidan, Ian Mcgraw, and Daphne Koller. Residual belief propagation: informed scheduling for asynchronous message passing. In *UAI*, 2006.

Pedro F. Felzenszwalb. Representation and detection of deformable shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):208–220, 2005.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

Delbert R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, (15):835–855, 1965.

Michel Galley. A skip-chain conditional random field for ranking meeting utterances by importance. In *EMNLP*, 2006.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distribution and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

David R. Karger, Daphne Koller, and Steven J. Phillips. Finding the hidden path: time bounds for all-pairs shortest paths. *SIAM Journal of Computing*, 22(6):1199–1217, 1993.

Leslie R. Kerr. The effect of algebraic structure on the computational complexity of matrix multiplication. *PhD Thesis*, 1970.

Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan. Counting belief propagation. In *UAI*, 2009.

Uffe Kjærulff. Inference in bayesian networks using nested junction trees. In *Proceedings of the NATO Advanced Study Institute on Learning in Graphical Models*, 1998.

Vladimir Kolmogorov and Akiyoshi Shioura. New algorithms for the dual of the convex cost network flow problem with application to computer vision. Technical report, University College London, 2007.

Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

M. Pawan Kumar and Philip Torr. Fast memory-efficient generalized belief propagation. In *ECCV*, 2006.

Xiang-Yang Lan, Stefan Roth, Daniel P. Huttenlocher, and Michael J. Black. Efficient belief propagation with learned higher-order markov random fields. In *ECCV*, 2006.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

Julian J. McAuley and Tibério S. Caetano. Exact inference in graphical models: is there more to it? *CoRR*, abs/0910.3301, 2009.

Julian J. McAuley and Tibério S. Caetano. Exploiting within-clique factorizations in junction-tree algorithms. *AISTATS*, 2010a.

Julian J. McAuley and Tibério S. Caetano. Exploiting data-independence for fast belief-propagation. *ICML*, 2010b.

Julian J. McAuley, Tibério S. Caetano, and Marconi S. Barbosa. Graph rigidity, cyclic belief propagation and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):2047–2054, 2008.

Alistair Moffat and Tadao Takaoka. An all pairs shortest path algorithm with expected time $O(n^2 \log n)$. *SIAM Journal of Computing*, 16(6):1023–1031, 1987.

James D. Park and Adnan Darwiche. A differential semantics for jointree algorithms. In *NIPS*, 2003.

Mark A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *IJCAI*, 2003.

Kersten Petersen, Janis Fehr, and Hans Burkhardt. Fast generalized belief propagation for MAP estimation on 2D and 3D grid-like markov random fields. In *DAGM*, 2008.

Daniel Scharstein and Richard S. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1–3):7–42, 2001.

Leonid Sigal and Michael J. Black. Predicting 3D people from 2D pictures. In *AMDO*, 2006.

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening LP relaxations for MAP using message passing. In *UAI*, 2008.

Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 14(3):354–356, 1969.

Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

Charles Sutton and Andrew McCallum. *Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

Philip A. Tresadern, Harish Bhaskar, Steve A. Adeshina, Chris J. Taylor, and Tim F. Cootes. Combining local and global shape models for deformable object matching. In *BMVC*, 2009.

Yair Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.

Ryan Williams. Matrix-vector multiplication in sub-quadratic time (some preprocessing required). In *SODA*, pages 1–11, 2007.

Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. In *FOCS*, 2010.

Journal of Machine Learning Research 12 (2011) 1389-1423 Submitted 8/09; Revised 9/10; Published 4/11

Clustering Algorithms for Chains

Antti Ukkonen AUKKONEN@YAHOO-INC.COM
Yahoo! Research
Av. Diagonal 177
08018 Barcelona, Spain

Editor: Marina Meila

Abstract

We consider the problem of clustering a set of chains to k clusters. A chain is a totally ordered subset of a finite set of items. Chains are an intuitive way to express preferences over a set of alternatives, as well as a useful representation of ratings in situations where the item-specific scores are either difficult to obtain, too noisy due to measurement error, or simply not as relevant as the order that they induce over the items. First we adapt the classical k-means for chains by proposing a suitable distance function and a centroid structure. We also present two different approaches for mapping chains to a vector space. The first one is related to the planted partition model, while the second one has an intuitive geometrical interpretation. Finally we discuss a randomization test for assessing the significance of a clustering. To this end we present an MCMC algorithm for sampling random sets of chains that share certain properties with the original data. The methods are studied in a series of experiments using real and artificial data. Results indicate that the methods produce interesting clusterings, and for certain types of inputs improve upon previous work on clustering algorithms for orders.

Keywords: Lloyd's algorithm, orders, preference statements, planted partition model, randomization testing

1. Introduction

Clustering (see, e.g., Alpaydin, 2004; Hand et al., 2001) is a traditional problem in data analysis. Given a set of objects, the task is to divide the objects to homogeneous groups based on some criteria, typically a distance function between the objects. Cluster analysis has applications in numerous fields, and a myriad of different algorithms for various clustering problems have been developed over the past decades. The reader is referred to the surveys by Xu and Wunsch (2005) and Berkhin (2006) for a more general discussion about clustering algorithms and their applications.

This work is about *clustering a set of orders*, a problem previously studied by Murphy and Martin (2003), Busse et al. (2007), and Kamishima and Akaho (2009). Rankings of items occur naturally in various applications, such as preference surveys, decision analysis, certain voting systems, and even bioinformatics. As an example, consider the Single transferable vote system (Tideman, 1995), where a vote is an ordered subset of the candidates. By clustering such votes, the set of voters can be divided to a number of groups based on their political views. Or, in gene expression analysis it is sometimes of interest to analyze the order of genes induced by the expression levels instead of the actual numeric values (Ben-Dor et al., 2002). In this case a clustering groups genes according to their activity for example under various environmental conditions.

We focus on a particular subclass of (partial) orders, called *chains*. Informally, chains are totally ordered subsets of a set of items, meaning that for all items that belong to a chain we know the order, and for items not belonging to the chain the order is unknown. For example, consider a preference survey about movies where the respondents are requested to rank movies they have seen from best to worst. In this scenario chains are a natural representation for the preference statements, as it is very unlikely that everyone would list the same movies. In a clustering of the responses people with similar preferences should be placed in the same cluster, while people who strongly disagree should be placed in different clusters.

This example also illustrates a very useful property of chains as preference statements: independence of the "scale" used by the respondents when assigning scores to the alternatives. For example, suppose that person A gives movie X three stars, and movie Y five stars. Person B gives movies X and Y one and three stars, respectively. While these ratings are very different, both A and B prefer movie Y to movie X. If we represent a response as a vector of ratings, there is a risk of obtaining clusters that are based on the general level of ratings instead the actual preferences. That is, one cluster might contain respondents who tend to give low ratings, while another cluster contains respondents who give high ratings. Clearly this is not a desirable outcome if the purpose is to study the preferences of the respondents. Statements in the form of chains let us directly focus on the relationships between the alternatives. Moreover, the use of chains can also facilitate preference elicitation, as people may find it easier to rank a small set of items instead of assigning scores to individual items.

Fundamentally the problem of clustering orders does not differ much from the problem of clustering any set of objects for which a distance function can be defined. There are some issues, however. First, *defining a good distance function for chains is not straightforward*. One option is to use existing distance functions for permutations, such as Kendall's tau or Spearman's rho. The usual approach to accommodate these for chains, as taken for example by Kamishima and Akaho (2009), is to only consider the common items of two chains. However, if the chains have no overlap, which can in practice happen quite often, their distance has to be defined in some arbitrary way. The second issue is the *computational complexity* of some of the operations that are commonly used by clustering algorithms. For instance, running Lloyd's algorithm (often called k-means) requires the computation of the mean of a set of objects. While this is very easy for numerical inputs and common distance functions, in case of orders one has to solve the rank aggregation problem that is computationally nontrivial; for some choices of the distance function rank aggregation is NP-hard (Dwork et al., 2001). We tackle the aforementioned issues on one hand by formulating the clustering problem in a way that no computationally hard subproblems are involved (Section 2), and on the other hand by by mapping the chains to a vector space (Section 3). By taking the latter approach the problem of clustering chains is reduced to that of clustering vectors in $\mathbb{R}^n$.

In general clustering algorithms will always produce a clustering. However, it is not obvious whether this clustering is reflecting any real phenomena present in the input. Chances are that the output is simply a consequence of random noise. Therefore, in addition to algorithms for finding a clustering, we also propose a method for assessing the validity of the clusterings we find. Our approach falls in the framework of *randomization testing* (Good, 2000), where the statistical significance of a data analysis result is evaluated by running the same analysis on a number of random data sets. If clusterings of a number of random data sets are indistinguishable from a clustering of real data (according to a relevant test statistic), the validity of the clustering found in real data can be questioned. To make use of this approach we propose a method for generating random sets of chains that share some properties with our original input (Section 4).

1.1 Related Work

Previous research on cluster analysis in general is too numerous to be covered here in full. Instead, we refer the readers to recent surveys by Xu and Wunsch (2005) and Berkhin (2006). For the problem of clustering orders, surprisingly little work has been done. The problem discussed in this paper is also studied by Kamishima and Akaho (2009), and earlier by Kamishima and Fujiki (2003). Murphy and Martin (2003) propose a mixture model for clustering orders. However, they only consider inputs that consist of total orders, that is, every chain in the input must order all items in M. This restriction is not made by Busse et al. (2007) who study a setting similar to ours. An important aspect of their approach is to represent a chain using the set of total orders that are compatible with the chain. This idea can also be found in the work by Critchlow (1985), and is a crucial component of a part of our work in Section 3. Recently Clémençon and Jakubowicz (2010) propose a distance function for permutations based on earth mover's distance between doubly stochastic matrices. While this framework seems quite interesting, extending it for chains seems nontrivial. The use of randomization testing (Good, 2000) in the context of data mining was first proposed by Gionis et al. (2007). Theoretical aspects of the sampling approach are discussed by Besag and Clifford (1989) and Besag and Clifford (1991).

1.2 Organization and Contributions of This Paper

The contributions of this paper are the following:

- In Section 2 we adapt Lloyd's algorithm (Lloyd, 1982) for chains. The main problem is the lack of a good distance function for chains, as well as the computational complexity of rank aggregation. At the core of our approach is to consider the probabilities of pairs of items to precede one another in the cluster.

- In Section 3 we present two methods for mapping chains to high-dimensional vector spaces. The first method aims to preserve the distance between two chains that are assumed to originate from the same component in a simple generative model. The second method represents each chain as the mean of the set of linear extensions of the chain. Our main contribution here is Theorem 5 stating that this can be achieved with a very simple mapping. In particular, it is not necessary to enumerate the set of linear extensions of a chain.

- In Section 4 we present an MCMC algorithm for uniformly sampling sets of chains that share a number of characteristics with a given set of chains. The random sets of chains are used for significance testing.

- In Section 5 we conduct a number of experiments to compare the proposed method with existing algorithms for clustering chains. Turns out that the algorithms are in some sense orthogonal. For smaller data sets the algorithms by Kamishima and Akaho (2009) give in most cases a better result. However, as the input size increases, the method proposed in this paper outperforms other algorithms.

Many of the results presented have appeared previously as a part of the author's doctoral dissertation (Ukkonen, 2008). Theorem 5 in Section 3.2 was presented earlier by Ukkonen (2007) but

Algorithm 1 Lloyd's algorithm

```
1: $k$-means($D$, $k$) {Input: $D$, set of points; $k$, number of clusters. Output: The clustering $\mathcal{C} = \{D_1,\ldots,D_k\}$.}
2: $\{D_1,\ldots,D_k\} \leftarrow$ PickInitialClusters( $D, k$ );
3: $e \leftarrow \sum_{i=1}^{k} \sum_{x \in D_i} d(\pi, \text{Centroid}(D_i))$;
4: repeat
5:     $e_0 \leftarrow e$;
6:     $\mathcal{C}_0 \leftarrow \{D_1,\ldots,D_k\}$;
7:     for $i \leftarrow 1,\ldots,k$ do
8:         $D_i \leftarrow \{x \in D \mid i = \arg\min_j d(x, \text{Centroid}(D_j))\}$;
9:     end for
10:    $e \leftarrow \sum_{i=1}^{k} \sum_{x \in D_i} d(x, \text{Centroid}(D_i))$;
11: until $e = e_0$ ;
12: return $\mathcal{C}_0$;
```

its proof was omitted. Also contents of Section 4 have appeared in less detail in previous work by Ukkonen and Mannila (2007).

2. Adapting Lloyd's Algorithm for Chains

Lloyd's algorithm, also known as k-means, is one of the most common clustering algorithms. In this section we address questions related to the use of Lloyd's algorithm with chains. We start with the basic definitions used throughout this paper.

2.1 Basic Definitions

Let M be a set of m items. A *chain* π is a subset of M together with a total order τ_π on the items, meaning that for every $u, v \in \pi \subseteq M$ we have either $(u,v) \in \tau_\pi$ or $(v,u) \in \tau_\pi$. We use a slightly simplified notation, and say that the pair (u,v) belongs to π, denoted $(u,v) \in \pi$, whenever $(u,v) \in \tau_\pi$. Whenever (u,v) belongs to π, we say that u precedes v according to π. For items in $M \setminus \pi$, the chain π does not specify the order in any way. The chain π is therefore a *partial order*. When π is defined over the entire set M of items, we say it is a *total order*. Let D be a multiset of n chains. A clustering of D is a disjoint partition of D to k subsets, denoted $D_1,\ldots,D_k$, so that every $\pi \in D$ belongs to one and only one D_i.

Lloyd's algorithm (Duda and Hart, 1973; Lloyd, 1982; Ball and Hall, 1967) finds a clustering of $D_1,\ldots,D_k$ so that its *reconstruction error*, defined as

$$\sum_{i=1}^{k} \sum_{x \in D_i} d(x, \text{Centroid}(D_i)), \tag{1}$$

is at a local minimum. Here d is a distance function, D_i is a cluster, and Centroid(D_i) refers to a "center point" of D_i. With numerical data one typically uses the mean as the centroid and squared Euclidean distance as d. The algorithm is given in Algorithm 1. On every iteration Lloyd's algorithm updates the clustering by assigning each point $x \in D$ to the cluster with the closest centroid. The PickInitialClusters function on line 2 of Algorithm 1 can be implemented for example by selecting k total orders at random, and assigning each chain to the the closest one. More sophisticated techniques, such as the one suggested by Arthur and Vassilvitskii (2007) can also be considered. The algorithm terminates when the clustering error no longer decreases. Note that the resulting clustering is not necessarily a global optima of Equation 1, but the algorithm can end up at a local minimum.

2.2 Problems with Chains

Clustering models are usually based on the concept of distance. In the case of hierarchical clustering we must be able to compute distances between two objects in the input, while with Lloyd's algorithm we have to compute distances to a centroid. Usually the centroid belongs to the same family of objects as the ones in D that we are clustering. However, it can also be something else, and in particular for the problem of clustering chains, *the centroid does not have to be a chain* or even a total order. This is very useful, because defining a good distance function for chains is not straightforward. For example, given the chains $(1,4,5)$ and $(2,3,6)$, it is not easy to say anything about their similarity, as they share no common items. We return to this question later in Section 3.1, but before this we will describe an approach where the distance between two chains is not required.

Another issue arises from the centroid computation. If we use a total order for representing the centroid we have to solve the rank aggregation problem: given all chains belonging to the cluster C_i, we have to compute a total order that is in some sense the "average" of the chains in C_i. This is not trivial, but can be solved by several different approaches. Some of them have theoretical performance guarantees, such as the algorithms by Ailon et al. (2005) and Coppersmith et al. (2006), and some are heuristics that happen to give reasonable results in practice (Kamishima and Akaho, 2006). The hardness of rank aggregation also depends on the distance function. For the Kendall's tau the problem is always NP-hard (Dwork et al., 2001), but for Spearman's rho it can be solved in polynomial time if all chains in the input happen to be total orders. In the general case the problem is NP-hard also for Spearman's rho (Dwork et al., 2001). Our approach is to replace the centroid with a structure that can be computed more efficiently.

2.3 Distances and Centroids

Next we discuss the choice of a centroid and a distance function so that Algorithm 1 can be used directly with an input consisting of chains. Suppose first that the centroid of a cluster is the total order τ. Observe that τ can be represented by a matrix X_τ, where $X_\tau(u,v) = 1$ if and only if we have $(u,v) \in \tau$, otherwise $X_\tau(u,v) = 0$. We can view X_τ as an *order relation*. This relation is completely deterministic, since each pair (u,v) either belongs, or does not belong to τ. Moreover, if (u,v) does not belong to τ, the pair (v,u) has to belong to τ.

A simple generalization of this is to allow the centroid to contain fractional contributions for the pairs. That is, the pair (u,v) may belong to the centroid with a weight that is a value between 0 and 1. We restrict the set of possible weights so that they satisfy the *probability constraint*, defined as $X(u,v) + X(v,u) = 1$ for all $u,v \in M$. In this case the centroid corresponds to a *probabilistic order relation*. Below we show that for a suitable distance function this approach leads to a natural generalization of the case where the centroids are represented by total orders together with Kendall's tau as the distance function. However, this relaxation lets us avoid the rank aggregation problem discussed above.

Consider the following general definition of a centroid. Given a set D of objects and the class Q of centroids for D, we want to find a $X^* \in Q$, so that

$$X^* = \arg\min_{X \in Q} \sum_{\pi \in D} d(\pi, X),$$

where $d(\pi, X)$ is a distance between π and X. Intuitively X^* must thus reside at the "center" of the set D. We let Q be set of probabilistic order relations on M, that is, the set of $|M| \times |M|$ matrices satisfying the probability constraint. Given a matrix $X \in Q$ and a chain π, we define the distance $d(\pi, X)$ as

$$d(\pi, X) = \sum_{(u,v) \in \pi} X(v,u)^2. \tag{2}$$

This choice of $d(\pi, X)$ leads to a simple way of computing the optimal centroid, as is shown below. Note that this distance function is equivalent with Kendall's tau if X is a deterministic order relation. To find the centroid of a given set D of chains, we must find a matrix $X \in Q$ such that the cost

$$c(X, D) = \sum_{\pi \in D} \sum_{(u,v) \in \pi} X(v,u)^2$$

is minimized. By writing the sum in terms of pairs of items instead of chains, we obtain

$$c(X, D) = \sum_{u \in M} \sum_{v \in M} C_D(u,v) X(v,u)^2,$$

where $C_D(u,v)$ denotes the number of chains in D where u appears before v. Let U denote the set of all unordered pairs of items from M. Using U the above can be written as

$$c(X, D) = \sum_{\{u,v\} \in U} \left(C_D(u,v) X(v,u)^2 + C_D(v,u) X(u,v)^2 \right).$$

As X must satisfy the probability constraint, this becomes

$$c(X, D) = \sum_{\{u,v\} \in U} \underbrace{\left(C_D(u,v)(1 - X(u,v))^2 + C_D(v,u) X(u,v)^2 \right)}_{c(X,\{u,v\})}. \tag{3}$$

To minimize Equation 3 it is enough to independently minimize the individual parts of the sum corresponding to the pairs in U, denoted $c(X, \{u,v\})$. Setting the first derivative of this with respect to $X(u,v)$ equal to zero gives

$$X^*(u,v) = \frac{C_D(u,v)}{C_D(u,v) + C_D(v,u)}. \tag{4}$$

That is, the optimal centroid is represented by a matrix X^* where $X^*(u,v)$ can be seen as a simple estimate of the probability of item $u \in M$ to precede item $v \in M$ in the input D. This is a natural way of expressing the the ordering information present in a set of chains without having to construct an explicit total order.

It is also worth noting that long chains will be consistently further away from the centroid than short chains, because we do not normalize Equation 2 with the length of the chain. This is not a problem, however, since we are only using the distance to assign a chain to one of the k centroids.

Distances of two chains of possibly different lengths are not compared. We also emphasize that even if longer chains in some sense contribute more to the centroid, as they contain a larger number of pairs, the contribution to an individual element of the matrix X is independent of chain length.

We propose thus to use Lloyd's algorithm as shown in Algorithm 1 with the distance function in Equation 2 and the centroid as defined by Equation 4. The algorithm converges to a local optimum, as the reconstruction error decreases on every step. When assigning chains to updated centroids the error can only decrease (or stay the same) because the chains are assigned to clusters that minimize the error (line 8 of Alg. 1). When we recompute the centroids given the new assignment of chains to clusters, the error is non-increasing as well, because the centroid X^* (Equation 4) by definition minimizes the error for every cluster.

3. Mappings to Vector Spaces

In this section we describe an alternative approach to clustering chains. Instead of operating directly on the chains, we first map them to a vector space. This makes it possible to compute the clustering using *any* algorithm that clusters vectors. Note that this will lead to a clustering that does not minimize the same objective function as the algorithm described in the previous section. However, the two approaches are complementary: we can first use the vector space representation to compute an initial clustering of the chains, and then refine this with Lloyd's algorithm using the centroid and distance function of the previous section. Note that these mappings can also be used to visualize sets of chains (Ukkonen, 2007; Kidwell et al., 2008).

3.1 Graph Representation

The mapping that we describe in this section is based on the adjacency matrices of two graphs where the chains of the input D appear as vertices. These graphs can be seen as special cases of the so called planted partition model (Condon and Karp, 2001; Shamir and Tsur, 2002).

3.1.1 MOTIVATION

We return to the question of computing the distance between two chains. Both Spearman's rho and Kendall's tau can be modified for chains so that they only consider the common items. If the chains π_1 and π_2 have no items in common, we have to use a fixed distance between π_1 and π_2. This is done for example by Kamishima and Fujiki (2003), where the distance between two chains is given by $1-\rho$, where $\rho \in [-1,1]$ is Spearman's rho. For two fully correlated chains the distance becomes 0, and for chains with strong negative correlation the distance is 2. If the chains have no common items we have $\rho = 0$ and the distance is 1. We could use the same approach also with the Kendall distance by defining the distance between the chains π_1 and π_2 as the (normalized) Kendall distance between the permutations that are induced by the common items in π_1 and π_2. If there are no common items we set the distance to 0.5. Now consider the following example. Let $\pi_1 = (1,2,3,4,5)$, $\pi_2 = (6,7,8,9,10)$, and $\pi_3 = (4,8,2,5,3)$. By definition we have $d_K(\pi_1,\pi_2) = 0.5$, and a simple calculation gives $d_K(\pi_1,\pi_3) = 0.5$ as well. Without any additional information this is a valid approach.

However, suppose that the input D has been generated by the following model: We are given k partial orders Π_j, $j = 1,\ldots,k$, on M. A chain π is generated by first selecting one of the Π_js at random, then choosing one linear extension τ of Π_j at random, and finally picking a random subset of l items and creating the chain by projecting τ on this subset. (This model is later used in the experiments in Section 5).

Continuing the example, let π_1, π_2, and π_3 be defined as above, assume for simplicity that the Π_js of the generative model are total orders, and that π_1 and π_2 have been generated by the same component, the total order $(1,2,3,4,5,6,7,8,9,10)$, and that π_3 is generated by another component, the total order $(6,7,9,10,4,8,2,5,3,1)$. Under this assumption it no longer appears meaningful to have $d_K(\pi_1,\pi_2) = d_K(\pi_1,\pi_3)$, as the clustering algorithm should separate chains generated by different components from each other. We would like to have $d_K(\pi_1,\pi_2) < d_K(\pi_1,\pi_3)$. Of course we can a priori not know the underlying components, but when computing a clustering we are assuming that they exist.

3.1.2 Agreement and Disagreement Graphs

Next we propose a method for mapping the chains to $\mathbb{R}^n$ so that the distances between the vectors that correspond to π_1, π_2 and π_3 satisfy the inequality of the example above. In general we want chains that are generated by the same component to have a shorter distance to each other than to chains that originate from other components. To this end, we define the distance between two chains in D as the distance between their neighborhoods in appropriately constructed graphs. If the neighborhoods are similar, that is, there are many chains in D that are (in a sense to be formalized shortly) "close to" both π_1 and π_2, we consider also π_1 and π_2 similar to each other. Note that this definition of distance between two chains is dependent on the input D. In other words, the distance between π_1 and π_2 can change if other chains in D are modified.

We say that chains π_1 and π_2 *agree* if for some items u and v we have $(u,v) \in \pi_1$ and $(u,v) \in \pi_2$. Likewise, the chains π_1 and π_2 *disagree* if for some u and v we have $(u,v) \in \pi_1$ and $(v,u) \in \pi_2$. Note that π_1 and π_2 can simultaneously both agree and disagree. We define the agreement and disagreement graphs:

Definition 1 *Let $G_a(D)$ and $G_d(D)$ be undirected graphs with chains in D as vertices. The graph $G_a(D)$ is the* agreement graph*, where two vertices are connected by an edge if their respective chains* agree and do not disagree. *The graph $G_d(D)$ is the* disagreement graph*, where two vertices are connected by an edge if their respective chains* disagree and do not agree.

The distance between chains π_1 and π_2 will be a function of the sets of neighboring vertices of π_1 and π_2 in $G_a(D)$ and $G_d(D)$. Before giving the precise definition we discuss some theory related to the graph $G_a(D)$. This will shed some light on the hardness of finding a clustering if the input D is very sparse.

3.1.3 The Planted Partition Model

Consider the following stochastic model for creating a random graph of n vertices. First partition the set of vertices to k disjoint subsets denoted $V_1,\ldots,V_k$. Then, independently generate edges between the vertices as follows: add an edge between two vertices that belong to the same subset with probability p, and add an edge between two vertices that belong to different subsets with probability $q < p$. This model, called the planted partition model, was first discussed by Condon and Karp (2001) and subsequently by Shamir and Tsur (2002). They also proposed algorithms for recovering the underlying clustering as long as the gap $\Delta = p - q$ is not too small.

Assuming a simple process that generates the input D we can view the agreement graph $G_a(D)$ as an instance of the planted partition model with values of p and q that depend on the characteristics of the input D. More specifically, let D be generated by k total orders on the set of items M, so that each chain $\pi \in D$ is the projection of one of the total orders on some l-sized subset of M. In theory we can compute a clustering of D by applying one of the existing algorithms for the planted partition model on the graph $G_a(D)$. However, this approach may fail in practice. We argue that for realistic inputs D the graph $G_a(D)$ is unlikely to satisfy the condition on the gap Δ required by the algorithms given by Condon and Karp (2001) and Shamir and Tsur (2002). Also, these algorithms are rather complex to implement.

We start by considering the probability of observing an edge between two vertices in the graph $G_a(D)$ when D is generated using the model outlined above. This happens when two independent events are realized. First, the chains corresponding to the vertices must have at least 2 common items, the probability of which we denote by $\Pr(|\pi_1 \cap \pi_2| \geq 2)$. Observe that this is the disjoint union of events where there are exactly i common items, $i \in [2,l]$. Therefore, we have $\Pr(|\pi_1 \cap \pi_2| \geq 2) = \sum_{i=2}^{l} \Pr(|\pi_1 \cap \pi_2| = i)$. Second, the common items must be ordered in the same way in both of the chains. Denote the probability of this by $\Pr(\pi_1 \perp_i \pi_2)$ for the case of i common items. The probability of observing an edge between π_1 and π_2 is thus given by the sum

$$\sum_{i=2}^{l} \Pr(|\pi_1 \cap \pi_2| = i) \Pr(\pi_1 \perp_i \pi_2). \tag{5}$$

Next we use this to derive the probabilities p and q of observing an edge between two chains that belong either to the same, or two different components, respectively. Clearly we have $\Pr(|\pi_1 \cap \pi_2| = i) = \binom{l}{i}\binom{m-l}{l-i}\binom{m}{l}^{-1}$ in both cases, as the number of common items is independent of their ordering. The only part that matters is thus $\Pr(\pi_1 \perp_i \pi_2)$. When π_1 and π_2 belong to the *same component*, this probability is equal to 1, because π_1 and π_2 are always guaranteed to order every subset of items in the same way. Hence Equation 5 gives

$$p = \binom{m}{l}^{-1} \sum_{i=2}^{l} \binom{l}{i}\binom{m-l}{l-i}. \tag{6}$$

When π_1 and π_2 belong to *different components*, we must make sure that the component that emits π_2 orders the common items in the same way as π_1. (To simplify matters we allow the second component to be identical to the one that has generated π_1. This will not significantly affect the subsequent analysis.) The number of permutations on m items where the order of i items is fixed is $m!/i!$. Since the component of π_2 is sampled uniformly at random from all possible permutations, we have $\Pr(\pi_1 \perp_i \pi_2) = \frac{m!}{i!m!} = 1/i!$. This together with Equation 5 yields

$$q = \binom{m}{l}^{-1} \sum_{i=2}^{l} \frac{\binom{l}{i}\binom{m-l}{l-i}}{i!}. \tag{7}$$

The algorithm of Condon and Karp (2001) requires a gap Δ of order $\Omega(n^{-\frac{1}{2}+\varepsilon})$ given an input of size n to find the correct partitioning (for $k = 2$). The improved algorithm by Shamir and Tsur (2002) is shown to produce a correct output with Δ of order $\Omega(kn^{-\frac{1}{2}} \log n)$. Another way of seeing these results is that as Δ decreases more and more data is needed (n must increase) for the algorithms to give good results. Next we study how the gap Δ behaves in $G_a(D)$ as a function of $m = |M|$ and the length l of the chains. (Assuming that all chains are of equal length.) Since we have

$$\Delta = p - q = \frac{\sum_{i=2}^{l} \binom{l}{i}\binom{m-l}{l-i}\left(1 - \frac{1}{i!}\right)}{\binom{m}{l}},$$

where $(1-\frac{1}{i!})$ is significantly less than 1 only for very small i (say, $i \leq 3$), it is reasonable to bound Δ by using an upper bound for p. We obtain the following theorem:

Theorem 2 *Let p and q be defined as in Equations 6 and 7, respectively, and let $\Delta = p - q$. For $l < m/2$, we have*

$$\Delta < p = O\left(\frac{l^2}{m}\right).$$

Proof See Appendix A.1. ■

The bound expresses how the density of the graph $G_a(D)$ depends on the number of items m and the length of the chains l. The gap Δ becomes smaller as m increases and l decreases. This, combined with the existing results concerning Δ, means that for short chains over a large M the input D has to be very large for the algorithms of Condon and Karp (2001) and Shamir and Tsur (2002) to produce good results. For example with $l = 5$ and $m = 200$, Theorem 2 gives an upper bound of $1/8$ for Δ. But for example the algorithm of Shamir and Tsur (2002) requires Δ to be lower bounded by $kn^{-\frac{1}{2}}\log(n)$ (up to a constant factor). To reach $1/8$ with $k = 2$, n must in this case be of order 10^5, which can be tricky for applications such as preference surveys. Therefore, we conclude that for these algorithms to be of practical use a relatively large number of chains is needed if the data consists of short chains over a large number of different items. Also, even though Theorem 2 is related to the graph $G_a(D)$, it gives some theoretical justification to the intuition that increasing the length of the chains should make the clusters easier to separate.

3.1.4 USING $G_a(D)$ AND $G_d(D)$

In the agreement graph, under ideal circumstances the chain π is mostly connected to chains generated by the same component as π. Also, it is easy to see that in the disagreement graph the chain π is (again under ideal circumstances) not connected to *any* of the chains generated by the same component, and only to chains generated by the other components. This latter fact makes it possible to find the correct clustering by finding a k-coloring of $G_d(D)$. Unfortunately this has little practical value as in real data sets we expect to observe noise that will distort both $G_a(D)$ and $G_d(D)$.

Above we argued that representations of two chains emitted by the same component should be more alike than representations of two chains emitted by different components. Consider the case where $k = 2$ and both clusters are of size $n/2$. Let $f_\pi \in \mathbb{R}^n$ be the row of the adjacency matrix of $G_a(D)$ that corresponds to chain π. Let chain π_1 be generated by the same component as π, and let π_2 be generated by a different component. Also, define the similarity s between f_π and $f_{\pi'}$ as the number of elements where both f_π and $f_{\pi'}$ have the value 1. Consider the expected value of this similarity under the planted partition model. We have:

$$\begin{aligned} E[s(f_\pi, f_{\pi_1})] &= \frac{n}{2}p^2 + \frac{n}{2}q^2 = \frac{n}{2}(p^2+q^2), \\ E[s(f_\pi, f_{\pi_2})] &= \frac{n}{2}pq + \frac{n}{2}qp = nqp. \end{aligned}$$

It is easy to see that $E[s(f_\pi, f_{\pi_1})] > E[f_\pi, f_{\pi_2}]$ if we let $p = cq$, with $c > 1$. (This is true if p and q are defined as in Equations 6 and 7.) Therefore, at least under these simple assumptions the expected distance between two chains from the same component is always less than the expected distance between two chains from different components. In practice we can combine the adjacency matrices of $G_a(D)$ and $G_d(D)$ to create the final mapping:

Definition 3 *Let $G_{ad} = G_a(D) - G_d(D)$, where $G_a(D)$ and $G_d(D)$ denote the adjacency matrices of the agreement and disagreement graphs. The representation of the chain π in $\mathbb{R}^n$ is the row of G_{ad} that corresponds to π.*

While the analysis above only concerns $G_a(D)$, we chose to combine both graphs in the final representation. This can be motivated by the following example. As above, let f_π denote the row of the adjacency matrix of $G_a(D)$ that corresponds to the chain π, and let g_π denote the same for $G_d(D)$. Suppose that the chain π_1 agrees with the chain π, meaning that $f_{\pi_1}(\pi) = 1$ and $g_{\pi_1}(\pi) = 0$, and let the chain π_2 disagree with π, meaning that $f_{\pi_2}(\pi) = 0$ and $g_{\pi_2}(\pi) = 1$. Also, assume that the chain π_3 neither agrees nor disagrees with π, meaning that $f_{\pi_3}(\pi) = g_{\pi_3}(\pi) = 0$. Intuitively, in this example the distance between π_1 and π_2 should be larger than the distance between π_1 and π_3. With $G_{ad}(D)$ this property is satisfied, as now in the final representations, defined as $h_{\pi_i} = f_{\pi_i} - g_{\pi_i}$, we have $h_{\pi_1}(\pi) = 1$, $h_{\pi_2}(\pi) = -1$, and $h_{\pi_3}(\pi) = 0$. Using only $G_a(D)$ fails to make this distinction, because $f_{\pi_2}(\pi) = f_{\pi_3}(\pi)$.

Using the agreement and disagreement graphs has the obvious drawback that the adjacency matrices of $G_a(D)$ and $G_d(D)$ are both of size $n \times n$, and computing one entry takes time proportional to l^2. Even though $G_a(D)$ and $G_d(D)$ have the theoretically nice property of being generated by the planted partition model, using them in practice can be prohibited by these scalability issues. However, there is some experimental evidence that the entire G_{ad} graph is not necessarily needed (Ukkonen, 2008).

3.2 Hypersphere Representation

Next we devise a method for mapping chains to an m-dimensional (as opposed to n-dimensional) vector space. The mapping can be computed in time $O(nm)$. This method has a slightly different motivation than the one discussed above. Let f be the mapping from the set of all chains to $\mathbb{R}^m$ and let d be a distance function in $\mathbb{R}^m$. Furthermore, let π be a chain and denote by π^R the reverse of π, that is, the chain that orders the same items as π, but in exactly the opposite way. The mapping f and distance d should satisfy

$$d(f(\pi), f(\pi^R)) = \max_{\pi'}\{d(f(\pi), f(\pi'))\} \tag{8}$$

$$d(f(\pi_1), f(\pi_1^R)) = d(f(\pi_2), f(\pi_2^R)) \text{ for all } \pi_1 \text{ and } \pi_2. \tag{9}$$

Less formally, we want the reversal of a chain to be furthest away from it in the vector space (8), and the distance between π and π^R should be the same for all chains (9). We proceed by first defining a mapping for total orders that satisfy the conditions above and then generalize this for chains. In both cases the mappings have an intuitive geometrical interpretation.

3.2.1 A MAPPING FOR TOTAL ORDERS

We define a function f that maps total orders to $\mathbb{R}^m$ as follows: Let τ be a total order on M, and let $\tau(u)$ denote the position of $u \in M$ in τ. For example, if $M = \{1, \dots, 8\}$ and $\tau = (5,1,6,3,7,2,8,4)$, we have $\tau(5) = 1$. Consider the vector $\mathbf{f}_\tau$ where

$$\mathbf{f}_\tau(u) = -\frac{m+1}{2} + \tau(u) \tag{10}$$

for all $u \in M$. We define the mapping f such that $f(\tau) = \mathbf{f}_\tau / \|\mathbf{f}_\tau\| = \hat{\mathbf{f}}_\tau$. Note that this mapping is a simple transformation of the Borda count (see, e.g., Moulin, 1991), where candidates in an election are given points based on their position in the order specified by a vote. Returning to the example, according to Equation 10 we have

$$\mathbf{f}_\tau = (-2.5, 1.5, -0.5, 3.5, -3.5, -1.5, 0.5, 2.5),$$

and as $\|\mathbf{f}_\tau\| = 6.48$, we have

$$f(\tau) = \hat{\mathbf{f}}_\tau = (-0.39, 0.23, -0.08, 0.54, -0.54, -0.23, 0.08, 0.39).$$

When d is the *cosine distance* between two vectors, which in this case is simply $1 - \hat{\mathbf{f}}_\tau^T \hat{\mathbf{f}}_{\tau'}$ as the vectors are normalized, it is straightforward to check that $\hat{\mathbf{f}}_\tau$ satisfies Equations 8 and 9. This mapping has a geometrical interpretation: all permutations are points on the surface of an m-dimensional unit-sphere centered at the origin. Moreover, the permutation τ and its reversal τ^R are on exactly opposite sides of the sphere. That is, the image of τ^R is found by mirroring the image of τ at the origin.

3.2.2 A Mapping for Chains

To extend the above for chains we apply the technique used also by Critchlow (1985) and later by Busse et al. (2007). The idea is to represent a chain π on M by the set of total orders on M that are compatible with π. That is, we view π as a partial order on M and use the set of linear extensions[1] of π to construct the representation $f(\pi)$. More precisely, we want $f(\pi)$ to be the *center* of the points in the set $\{f(\tau) : \tau \in \mathcal{E}(\pi)\}$, where f is the mapping for permutations defined in the previous section, and $\mathcal{E}(\pi)$ is the set of linear extensions of π. Our main contribution in this section is that despite the size of $\mathcal{E}(\pi)$ is $\binom{m}{l}(m-l)!$, we can compute $f(\pi)$ very efficiently. We start by giving a definition for $f(\pi)$ that is unrelated to $\mathcal{E}(\pi)$.

Definition 4 *Let π be a chain over M and define the vector $\mathbf{f}_\pi$ so that*

$$\mathbf{f}_\pi(u) = \begin{cases} -\frac{|\pi|+1}{2} + \pi(u) & \textit{iff } u \in \pi, \\ 0 & \textit{iff } u \notin \pi, \end{cases} \tag{11}$$

for all $u \in M$. The mapping f is defined so that $f(\pi) = \mathbf{f}_\pi / \|\mathbf{f}_\pi\| = \hat{\mathbf{f}}_\pi$.

This is a generalization of the mapping for total orders to the case where only a subset of the items has been ordered. The following theorem states that this definition makes $f(\pi)$ the center of the set $\{f(\tau) : \tau \in \mathcal{E}(\pi)\}$.

Theorem 5 *If the vector $\mathbf{f}_\tau$ is defined as in Equation 10, and the vector $\mathbf{f}_\pi$ is defined as in Equation 11, then there exists a constant Q so that*

$$\mathbf{f}_\pi(u) = Q \sum_{\tau \in \mathcal{E}(\pi)} \mathbf{f}_\tau(u) \tag{12}$$

for all $u \in M$.

1. A linear extension of a partial order π is a total order τ so that $(u,v) \in \pi \rightarrow (u,v) \in \tau$.

Proof See Appendix A.2. ■

What does this theorem mean in practice? We want $f(\pi)$ to be the mean of the points that represent the linear extensions of π, normalized to unit length. Theorem 5 states that this mean has a simple explicit formula that is given by Equation 11. Thus, when normalizing $\mathbf{f}_\pi$ we indeed get the normalized mean vector *without having to sum over all linear extensions of* π. This is very important, as $\mathcal{E}(\pi)$ is so large that simply enumerating all its members is computationally infeasible.

The first advantage of the hypersphere representation over the agreement and disagreement graphs is efficiency. Computing the vectors $\mathbf{f}_\pi$ for all chains in the input is of order $O(nm)$, which is considerably less than the requirement of $O(n^2m^2)$ for the graph based approach. As a downside we lose the property of having a shorter distance between chains generated by the same component than between chains generated by different components. The second advantage of the hypersphere mapping is size. Storing the full graph representation requires $O(n^2)$ memory, while storing the hypersphere representation needs only $O(nm)$ of storage. This is the same as needed for storing D, and in most cases less than $O(n^2)$ as usually we have $m \ll n$.

4. Assessing the Significance of Clusterings

Clustering algorithms will in general always produce a clustering of the input objects. However, it is not obvious that these clusterings are meaningful. If we run one of the algorithms discussed above on a random set of chains, we obtain a clustering as a result. But clearly this clustering has in practice no meaning. To assess the significance of a clustering of the input D, we compare its reconstruction error with the errors of clusterings obtained from random (in a sense made precise below) sets of chains. If the error from real data is smaller than the errors from random data, we have evidence for the clustering to be meaningful. The random sets of chains must share certain aspects with our original input D. In this section we define these aspects precisely, and devise a method for sampling randomized sets of chains that share these aspects with a given input D.

4.1 Randomization Testing and Empirical p-values

For a thorough discussion of randomization testing, we refer the reader to the textbook by Good (2000). Below we give only a brief outline and necessary definitions. Denote by $\mathcal{A}$ a data analysis algorithm that takes D as the input and produces some output, denoted $\mathcal{A}(D)$. We can assume that $\mathcal{A}(D)$ is in fact the value of a *test statistic* that we are interested in. For the remainder of this paper $\mathcal{A}$ is a clustering algorithm and $\mathcal{A}(D)$ is the reconstruction error of the clustering found by $\mathcal{A}$. Moreover, denote by $\tilde{D}_1, \ldots, \tilde{D}_h$ a sequence of random sets of chains that share certain properties with D. These will be defined more formally later.

If the value $\mathcal{A}(D)$ considerably deviates from the values $\mathcal{A}(\tilde{D}_1), \ldots, \mathcal{A}(\tilde{D}_h)$, we have some evidence for the output of $\mathcal{A}$ to be meaningful. In practice this means we can rule out the common properties of the real and random data sets as the sole causes for the results found. As usual in statistical testing we can speak of a *null hypothesis* H_0 and an *alternative hypothesis* H_1. These are defined as follows:

$$\begin{aligned} H_0 : \mathcal{A}(D) &\geq \min_i \{\mathcal{A}(\tilde{D}_i)\}, \\ H_1 : \mathcal{A}(D) &< \min_i \{\mathcal{A}(\tilde{D}_i)\}. \end{aligned}$$

In statistics the *p-value* of a test usually refers to the probability of making an error when rejecting H_0 (and accepting H_1). In order to determine the p-value one typically needs to make some assumptions of the distribution of the test statistic. In general, if we cannot, or do not want to make such assumptions, we can compute the *empirical p-value* based on the randomized data sets. This is defined simply as the fraction of cases where the value of $\mathcal{A}(\tilde{D}_i)$ is more extreme than the value $\mathcal{A}(D)$. Or more formally, for the one-tailed case where $\mathcal{A}(D)$ is expected to be small according to H_1, we have

$$\hat{p} = \frac{|\{\tilde{D}_i : \mathcal{A}(\tilde{D}_i) \leq \mathcal{A}(D)\}| + 1}{h+1}.$$

One problem with using $\hat{p}$ is that in order to get useful values the number of randomized data sets must be fairly high. For instance, to have $\hat{p} = 0.001$ we must sample at least 999 data sets. Depending on the complexity of generating one random data set this may be difficult. Of course, already with 99 data sets we can obtain an empirical p-value of 0.01 if all random data sets have a larger value of the test statistic. This should be enough for many practical applications.

4.2 Equivalence Classes of Sets of Chains

The random data sets must share some characteristics with the original data D. Given D, we define an equivalence class of sets of chains, so that all sets belonging to this equivalence class have the same properties as D.

Definition 6 *Let D_1 and D_2 be two sets of chains on items of the set M. D_1 and D_2 belong to the same equivalence class whenever the following three conditions hold.*

1. *The number of chains of length l is the same in D_1 as in D_2 for all l.*
2. *For all $M' \subseteq M$, the number of chains that contain M' as a subset is the same in D_1 and D_2.*
3. *We have $C_{D_1}(u,v) = C_{D_2}(u,v)$ for all $u,v \in M$, where $C_D(u,v)$ is the number of chains in D that rank u before v.*

Given a set D of chains, we denote the equivalence class specified by D with $\mathcal{C}(D)$. Next we discuss an algorithm for sampling uniformly from $\mathcal{C}(D)$. But first we elaborate why it is useful to maintain the properties listed above when testing the significance of $\mathcal{A}(D)$.

When analyzing chains over the items in M, the most interesting property is how the chains actually order the items. In other words, the clustering should reflect the *ordering information* present in D. This is only one property of D, however. Other properties are those that we mention in the conditions above. Condition 1 is used to rule out the possibility that the value of $\mathcal{A}(D)$ is somehow caused only by the length distribution of the chains in D. Note that this requirement also implies that D_1 and D_2 are of the same size. Likewise, condition 2 should rule out the possibility that the result is not a consequence of the rankings, but simply the co-occurrences of the items.

Maintaining $C_D(u,v)$ is motivated from a slightly different point of view. If D contained real-valued vectors instead of chains, it would make sense to maintain the empirical mean of the observations. The intuition with chains is the same: we view D as a set of points in the space of chains. The random data sets should be located in the same region of this space as D. By maintaining $C_D(u,v)$ the randomized data sets $\tilde{D}_i$ will (in a way) have the same mean as D. This is because the rank aggregation problem, that is, finding the mean of a set of permutations, can be solved using only the $C_D(u,v)$ values (Ukkonen, 2008).

4.3 An MCMC Algorithm for Sampling from $\mathcal{C}(D)$

Next we will discuss a Markov chain Monte Carlo algorithm that samples uniformly from a subset of $\mathcal{C}(D)$ given D. We can only guarantee that the sample will be from a neighborhood of D in $\mathcal{C}(D)$. Whether this neighborhood covers all of $\mathcal{C}(D)$ is an open problem.

4.3.1 ALGORITHM OVERVIEW

The MCMC algorithm we propose can be seen as a random walk on an undirected graph with $\mathcal{C}(D)$ as the set of vertices. Denote this graph by $G(D)$. The vertices D_1 and D_2 of $G(D)$ are connected by an edge if we obtain D_2 from D_1 by performing a small local modification to D_1 (and vice versa). We call this local modification a *swap* and will define it in detail below. First, let us look at a high level description of the algorithm.

In general, when using MCMC to sample from a distribution, we must construct the Markov Chain so that its *stationary distribution* equals the target distribution we want to sample from. If all vertices of $G(D)$ are of equal degree, the stationary distribution will be the uniform distribution. As we want to sample uniformly from $\mathcal{C}(D)$, this would be optimal. However, it turns out that the way we have defined the graph $G(D)$ does not result in the vertices having the same number of neighboring vertices. To remedy this, we use the *Metropolis-Hastings* algorithm (see, e.g., Gelman et al., 2004) for picking the next state. Denote by $N(D_i)$ the set of neighbors of the vertex D_i in $G(D)$. When the chain is at D_i, we pick uniformly at random the vertex D_{i+1} from $N(D_i)$. The chain moves to D_{i+1} with probability

$$\min\left(\frac{|N(D_i)|}{|N(D_{i+1})|}, 1\right), \tag{13}$$

that is, the move is accepted always when D_{i+1} has a smaller degree, and otherwise we move with a probability that decreases as the degree of D_{i+1} increases. If the chain does not move, it stays at the state D_i and attempts to move again (possibly to some other neighboring vertex) in the next step.

It is easy to show that this modified random walk has the desired property of converging to a uniform distribution over the set of vertices. Denote by $p(D_i)$ the *target distribution* we want to sample from. In this case $p(D_i)$ is the uniform distribution over $\mathcal{C}(D)$. Hence, we must have $p(D_i) = p(D_{i+1}) = |\mathcal{C}(D)|^{-1}$. The Metropolis-Hastings algorithm jumps to the next state D_{i+1} with probability $\min(r, 1)$, where

$$r = \frac{p(D_{i+1})/J(D_{i+1}|D_i)}{p(D_i)/J(D_i|D_{i+1})}. \tag{14}$$

Above $J(\cdot|\cdot)$ is a *proposal distribution*, which in this case is simply the uniform distribution over the neighbors of D_i for all i. That is, we have $J(D_{i+1}|D_i) = |N(D_i)|^{-1}$ and $J(D_i|D_{i+1}) = |N(D_{i+1})|^{-1}$. When this is substituted into Equation 14 along with the fact that $p(D_i) = p(D_{i+1})$ we obtain Equation 13.

Given D, a simple procedure for sampling one $\tilde{D}$ uniformly from $\mathcal{C}(D)$ works as follows: we start from $D = D_0$, run the Markov chain resulting in slightly modified data D_i on every step i. After s steps we are at the set D_s which is our $\tilde{D}$. We repeat this process until enough samples from $\mathcal{C}(D)$ have been obtained. It is very important to run the Markov chain long enough (have a large enough s), so that the samples are as uncorrelated as possible with the starting point D, as well as independent of each other. We will discuss a heuristic for assessing the correct number steps below.

However, guaranteeing that the samples are independent is nontrivial. Therefore we only require the samples to be *exchangeable*. The following approach, originally proposed by Besag and Clifford (1989), draws h sets of chains from $\mathcal{C}(D)$ so that the samples satisfy the exchangeability condition. We first start the Markov chain from D and run it *backwards* for s steps. (In practice the way we define our Markov chain, running it backwards is equivalent to running it forwards.) This gives us the set $\tilde{D}_0$. Next, we run the chain forwards $h-1$ times for s steps, each time starting from $\tilde{D}_0$. This way the samples are not dependent on each other, but only on $\tilde{D}_0$. And since we obtained $\tilde{D}_0$ by running the Markov chain backwards from D, the samples depend on $\tilde{D}_0$ in the same way as D depends on $\tilde{D}_0$. Note that a somewhat more efficient approach is proposed by Besag and Clifford (1991).

4.3.2 THE SWAP

Above we defined the Markov chain as a random walk over the elements of $\mathcal{C}(D)$, where two states D and D' are connected if one can be obtained from the other by a local modification operator. We call this local modification a *swap* for reasons that will become apparent shortly. Since the Markov chain must remain in $\mathcal{C}(D)$, the swap may never result in a set of chains $\hat{D} \notin \mathcal{C}(D)$. More precisely, if D_{i+1} is obtained from D_i by the swap and $D_i \in \mathcal{C}(D)$, then D_{i+1} must belong to $\mathcal{C}(D)$ as well. Next we define a swap that has this property.

Formally we define a swap as the tuple (π_1, π_2, i, j), where π_1 and π_2 are chains, i is an index of π_1, and j an index of π_2. To execute the swap (π_1, π_2, i, j), we transpose the items at positions i and $i+1$ in π_1, and at positions j and $j+1$ in π_2. For example, if $\pi_1 = (1,2,3,4,5)$ and $\pi_2 = (3,2,6,4,1)$, the swap $(\pi_1, \pi_2, 2, 1)$ will result in the chains $\pi'_1 = (1,3,2,4,5)$ and $\pi'_2 = (2,3,6,4,1)$. The positions of items 2 and 3 are changed in both π_1 and π_2.

Clearly this swap does not affect the number of chains, lengths of any chain, nor the occurrence frequencies of any itemset as items are not inserted or removed. To guarantee that also the $C_D(u,v)$s are preserved, we must pose one additional requirement for the swap. When transposing two adjacent items in the chain π_1, say, u and v with u originally before v, $C_D(u,v)$ is decremented by one as there is one instance less of u preceding v after the transposition, and $C_D(v,u)$ is incremented by one as now there is one instance more where v precedes u. Obviously, if the swap would change only π_1, the resulting data set would no longer belong to $\mathcal{C}(D)$ as $C_D(u,v)$ and $C_D(v,u)$ are changed. But the second transposition we carry out in π_2 cancels out the effect the first transposition had on $C_D(u,v)$ and $C_D(v,u)$, and the resulting set of chains remains in $\mathcal{C}(D)$.

Definition 7 *Let D be a set of chains and let π_1 and π_2 belong to D. The tuple (π_1, π_2, i, j) is a* valid swap *for D, if the item at the ith position of π_1 is the same as the item at the $j+1$th position of π_2, and if the item at $i+1$th position of π_1 is the same as the item at the jth position of π_2.*

The swap we show in the example above is thus a valid swap.

Given the data D, we may have several valid swaps to choose from. To see how the set of valid swaps evolves in a single step of the algorithm, consider the following example. Let D_i contain the three chains below:

$$\pi_1 : (1,2,3,4,5) \qquad \pi_2 : (7,8,4,3,6) \qquad \pi_3 : (3,2,6,4,1)$$

The valid swaps in this case are $(\pi_1, \pi_3, 2, 1)$ and $(\pi_1, \pi_2, 3, 3)$. If we apply the swap $(\pi_1, \pi_2, 3, 3)$ we obtain the chains

$$\pi'_1 : (1,2,4,3,5) \qquad \pi'_2 : (7,8,3,4,6) \qquad \pi_3 : (3,2,6,4,1)$$

Obviously $(\pi_1,\pi_2,3,3)$ is still a valid swap, as we can always revert the previous swap. But notice that $(\pi_1,\pi_2,2,1)$ is no longer a valid swap as the items 2 and 3 are not adjacent in π'_1. Instead $(\pi'_2,\pi_3,4,3)$ is introduced as a new valid swap since now 4 and 6 are adjacent in π'_2.

Given this definition of the swap, is $\mathcal{C}(D)$ connected with respect to the valid swaps? Meaning, can we reach every member of $\mathcal{C}(D)$ starting from D? This is a desirable property as we want to sample uniformly from $\mathcal{C}(D)$, but so far this remains an open question.

4.3.3 CONVERGENCE

Above it was mentioned that we must let the Markov chain run long enough to make sure $\tilde{D}_s$ is not correlated with the starting state D_0. The chain should have *mixed*, meaning that when we stop it the probability of landing at a particular state D_s actually corresponds to the probability D_s has in the stationary distribution of the chain. Determining when a simulated Markov chain has converged to its stationary distribution is not easy.

Hence we resort to a fairly simple heuristic. An indicator of the current sample D_i being uncorrelated to $D_0 = D$ is the following measure:

$$\delta(D,D_i) = |D|^{-1} \sum_{j=1}^{|D|} d_K(D(j),D_i(j)), \tag{15}$$

where $D(j)$ is the jth chain in D. Note that $\delta(D,D_i)$ is always defined, as the chain $D_i(j)$ is a permutation of $D(j)$. The distance defined in Equation 15 is thus the average Kendall distance between the permutations in D and D_i. To assess the convergence we observe how $\delta(D,D_i)$ behaves as i grows. When $\delta(D,D_i)$ has converged to some value or is not increasing only at a very low rate, we assume the current sample is not correlated with D_0 more strongly than with most other members of $\mathcal{C}(D)$.

Note that here we are assuming that the chains in D are *labeled*. To see what this means consider the following example with the sets D and D_i both containing four chains.

$D(1) : 1,2,3$	$D_i(1) : 2,1,3$
$D(2) : 4,5,6$	$D_i(2) : 6,5,4$
$D(3) : 2,1,3$	$D_i(3) : 1,2,3$
$D(4) : 6,5,4$	$D_i(4) : 4,5,6$

Here we have obtained D_i from D with the multiple swap operations. The distance $\delta(D,D_i)$ is 2 even though D and D_i clearly are identical as sets. Hence, the measure of Equation 15 can not be used for testing this identity. To do this we should compute the Kendall distance between $D(j)$ and $D_i(h(j))$, where h is a bijective mapping between chains in D and D_i that minimizes the sum of the pairwise distances. However, we consider this simple approach sufficient for the purposes of this paper.

4.3.4 IMPLEMENTATION ISSUES

Until now we have discussed the approach at a general level. There's also a practical issue when implementing the proposed algorithm. The number of valid swaps at a given state is of order $O(m^2n^2)$ in the worst case, which can get prohibitively large for storing each valid swap as a tuple explicitly. Hence, we do not store the tuples, but only maintain two sets that represent the entire set of swaps but use a factor of n less space. We let

$$A_D = \{\{u,v\} \mid \exists \pi_1 \in D \,\text{st.}\, uv \in \pi_1 \wedge \exists \pi_2 \in D \,\text{st.}\, vu \in \pi_2\},$$

where $uv \in \pi$ denotes that u and v are adjacent in π with u before v. This is the set of *swappable pairs* of items. The size of A_D is of order $O(m^2)$ in the worst case. In addition, we also have the sets

$$S_D(u,v) = \{\pi \in D \mid uv \in \pi\}$$

for all (u,v) pairs. This is simply a list that contains the set of chains where we can transpose u and v. Note that $S_D(u,v)$ and $S_D(v,u)$ are not the same set. In $S_D(u,v)$ we have chains where u appears before v, while in $S_D(v,u)$ are chains where v appears before u. The size of each $S_D(u,v)$ is of order $O(n)$ in the worst case, and the storage requirement for A_D and S_D is hence only $O(m^2n)$, a factor of n less than storing the tuples explicitly.

The sets A_D and S_D indeed fully represent all possible valid swaps. A valid swap is constructed from A_D and S_D by first picking a swappable pair $\{u,v\}$ from A_D, and then picking two chains, one from $S_D(u,v)$ and the other from $S_D(v,u)$. It is easy to see that a swap constructed this way must be a valid swap. Also, verifying that there are no valid swaps not described by A_D and S_D is straightforward.

There is still one concern. Recall that we want to use the Metropolis-Hastings approach to sample from the uniform distribution over $\mathcal{C}(D)$. In order to do this we must be able to sample uniformly from the neighbors of D_i, and we have to know the precise size of D_i's neighborhood. The size of the neighborhood $N(D_i)$ is precisely the number of valid swaps at D_i, and is given by

$$|N(D_i)| = \sum_{\{u,v\} \in A_{D_i}} |S_{D_i}(u,v)| \cdot |S_{D_i}(v,u)|,$$

which is easy to compute given A_{D_i} and S_{D_i}.

To sample a neighbor of D_i uniformly at random using A_{D_i} and S_{D_i}, we first pick the swappable pair $\{u,v\}$ from A_{D_i} with the probability

$$Pr(\{u,v\}) = \frac{|S_{D_i}(u,v)| \cdot |S_{D_i}(v,u)|}{|N(D_i)|}, \tag{16}$$

which is simply the fraction of valid swaps in $N(D_i)$ that affect items u and v. Then π_1 and π_2 are sampled uniformly from $S_D(u,v)$ and $S_D(v,u)$ with probabilities $|S_D(u,v)|^{-1}$ and $|S_D(v,u)|^{-1}$, respectively. Thus we have

$$Pr(\{u,v\}) \cdot |S_D(u,v)|^{-1} \cdot |S_D(v,u)|^{-1} = \frac{1}{|N(D_i)|}$$

as required.

The final algorithm that we call SWAP-PAIRS is given in Algorithm 2. It takes as arguments the data D and the integer s that specifies the number of rounds the algorithm is run. On lines 2–6 we initialize the sets A_D and S_D, while lines 8–20 contain the main loop. First, on line 9 the pair $\{u,v\}$ is sampled from A_D with the probability given in Equation 16. The SAMPLE-UNIFORM function simply samples an element from the set it is given as the argument. On lines 13 and 15 we compute the neighborhood sizes before and after the swap, respectively. The actual swap is carried out by the APPLY-SWAP function, that modifies π and τ in D and updates A_D and S_D accordingly. Lines 16–18 implement the Metropolis-Hastings step. Note that it is easier to simply perform the swap and backtrack if the jump should not have been accepted. A swap can be canceled simply by applying it a second time. The function RAND() returns a uniformly distributed number from the interval $[0,1]$.

Algorithm 2 The SWAP-PAIRS algorithm for sampling uniformly from $\mathcal{C}(D)$.

```
1:  SWAP-PAIRS($D, s$)
2:  $A_D \leftarrow \{\{u,v\} \mid \exists \pi_1 \in D \,\mathrm{st.}\, uv \in \pi_1 \wedge \exists \pi_2 \in D \,\mathrm{st.}\, vu \in \pi_2\}$
3:  for all $\{u,v\} \in A_D$ do
4:      $S_D(u,v) \leftarrow \{\pi \in D \mid uv \in \pi\}$
5:      $S_D(v,u) \leftarrow \{\pi \in D \mid vu \in \pi\}$
6:  end for
7:  $i \leftarrow 0$
8:  while $i < n$ do
9:      $\{u,v\} \leftarrow$ SAMPLE-PAIR$(A_D, S_D)$
10:     $\pi \leftarrow$ SAMPLE-UNIFORM$(S_D(u,v))$
11:     $\tau \leftarrow$ SAMPLE-UNIFORM$(S_D(v,u))$
12:     $s \leftarrow (\pi, \tau, \pi(u), \tau(v))$
13:     $N_{\text{before}} \leftarrow \sum_{\{u,v\} \in A_D} |S_D(u,v)| \cdot |S_D(v,u)|$
14:     APPLY-SWAP$(s, D, A_D, S_D)$
15:     $N_{\text{after}} \leftarrow \sum_{\{u,v\} \in A_D} |S_D(u,v)| \cdot |S_D(v,u)|$
16:     if RAND$() \geq \frac{N_{\text{before}}}{N_{\text{after}}}$ then
17:         APPLY-SWAP$(s, D, A_D, S_D)$
18:     end if
19:     $i \leftarrow i+1$
20: end while
21: return $D$
```

5. Experiments

In this section we discuss experiments that demonstrate how our algorithms perform on various artificial and real data sets. We consider a two-step algorithm that either starts with random initial clusters (RND), or a clustering that is computed with standard k-means (initialized with random centroids) in the graph (GR) or hypersphere (HS) representation. This initial clustering is subsequently refined with the variant of Lloyd's algorithm discussed in Section 2 to obtain the final clustering. We also compare our method against existing approaches by Kamishima and Akaho (2006). These algorithms, called TMSE and EBC, are similar clustering algorithms for sets of chains, but they are based on slightly different distance functions and types of centroid. We used original implementations of TMSE and EBC that were obtained from the authors.

5.1 Data Sets

The artificial data sets are generated by the procedure described in Section 3.1.1. In addition to artificial data we use four real data sets that are all based on publicly available sources. The data consist of preference rankings that are either explicit, derived, or observed. We say a preference ranking is *explicit* if the preferences are directly given as a ranked list of alternatives. A preference ranking is *derived* if the ranking is based on item-specific scores, such as movie ratings. Finally, a preference ranking is *observed* if it originates from a source where preferences over alternatives only manifest themselves indirectly in different types of behavior, such as web server access logs.

	SUSHI	MLENS	DUBLIN	MSNBC
n	5000	2191	5000	5000
m	100	207	12	17
min. l	10	6	4	6
avg. l	10	13.3	4.8	6.5
max. l	10	15	6	8

Table 1: Key statistics for different real data sets. The number of chains, the number of items, and the length of a chain are denoted by n, m, l, respectively.

Key statistics of the data sets are summarized in Table 1. More details are given below for each data set.

5.1.1 SUSHI

These data are explicit preference rankings of subsets of 100 items. Each chain is a response from a survey[2] where participants were asked to rank 10 flavors of sushi in order of preference. Each set of 10 flavors was chosen randomly from a total set of 100 flavors. The data consists of 5000 such responses.

5.1.2 MLENS

These data are derived preference rankings of subsets of 207 items. The original data consists of movie ratings (1–5 stars) collected by the GroupLens[3] research group at University of Minnesota. We discarded movies that had been ranked by fewer than 1000 users and were left with 207 movies. Next we pruned users who have not used the entire scale of five stars in their ratings and were left with 2191 users. We generate one chain per user by first sampling a subset of movies the user has rated, so that at most three movies having the same rating are in the sample. Finally we order the sample according to the ratings and break ties in ratings arbitrarily.

5.1.3 DUBLIN

These data are explicit preference rankings of subsets of 12 items. Each chain is a vote placed in the 2002 general elections in Ireland.[4] and ranks a subset of 12 candidates from the electoral district of northern Dublin. We only consider votes that rank at least 4 and at most 6 candidates and are left with 17737 chains. Of this we took a random sample of 5000 chains for the analysis.

5.1.4 MSNBC

These data are observed preference rankings over 17 items. Each chain shows the order in which a user accessed a subset of 17 different sections of a web site (`msnbc.com`).[5] Each chain contains only the first occurrence of a category, subsequent occurrences were removed. Also, we selected a

2. The SUSHI data be found at `http://www.kamishima.net/sushi` (29 April 2011).
3. The MLENS data can be found at `http://www.grouplens.org/node/12` (29 April 2011).
4. At the time of publication this data can be found by accessing old versions of `http://www.dublincountyreturningofficer.com/` in the Internet Archive at `http://waybackmachine.org`.
5. MSNBC data can be found at `http://kdd.ics.uci.edu/databases/msnbc/` (29 April 2011).

subset of the users who had visited at least 6 and at most 8 different categories and were left with 14598 chains. Again we used a random subset of 5000 chains for the analysis.

5.2 Recovering a Planted Clustering

In this section we discuss experiments on artificial data, with the emphasis on studying the performance of the algorithms under different conditions. These conditions can be characterized by parameters of the input data, such as length of the chains or total number of items. The task is to recover a "true" clustering that was planted in the input data.

5.2.1 EXPERIMENTAL SETUP

The notion of correctness is difficult to define when it comes to clustering models. With real data we do not in general know the correct structure, or if there even is any structure to be found. To have a meaningful definition of a correct clustering, we generate synthetic data that contains a planted clustering. We compare this with the clusterings found by the algorithms.

To measure the similarity between two clusterings we use a variant of the Rand Index (Rand, 1971) called the Adjusted Rand Index (Lawrence and Phipps, 1985). The basic Rand Index essentially counts the number of pairs of points where two clusterings agree (either both assign the points in the same cluster, or both assign the points in different clusters), normalized by the total number of pairs. The maximum value for two completely agreeing clusterings is thus 1. The downside with this approach is that as the number of clusters increases, even random partitions will have a score close to 1, which makes it difficult to compare algorithms. The Adjusted Rand Index corrects for this by normalizing the scores with respect to the expected value of the score under the assumption that the random partition follows a generalized hypergeometric distribution (Lawrence and Phipps, 1985).

Artificial sets of chains are created with the procedure described in Section 3.1.1. Instead of arbitrary partial orders as the components, we use bucket orders (or *ordered partitions*) of M. More specifically, a bucket order on M is a totally ordered set of disjoint subsets (buckets) of M that cover all items in M. If the items u and v both belong to the bucket $M_i \subseteq M$, they are unordered. If $u \in M_i \subseteq M$ and $v \in M_j \subseteq M$, and M_i precedes M_j, then also u precedes v. We used bucket orders with 10 buckets in the experiments.

Input size n is fixed to 2000. We varied the following parameters: length of a chain l, total number of items m, and number of clusters in the true clustering κ. We ran the algorithms on various combinations of these with different values of k, that is, we also wanted to study how the algorithms behave when the correct number of clusters is not known in advance.

5.2.2 COMPARING INITIALIZATION STRATEGIES

Results for our variant of Lloyd's algorithm with the three different initialization strategies (HS, GR, and RND) are shown in Figure 1 for a number of combinations of k and m. Here we only plot cases where $k = \kappa$, meaning that the algorithm was given the correct number of clusters in advance. The grey lines are 95 percent confidence intervals. As on one hand suggested by intuition, and on the other hand by Theorem 2, finding a planted clustering becomes easier as the length of the chains increase. With $l = 9$ the original clustering is found almost always independent of the values of m and k. For smaller values of l the effect of m and k is stronger. The problem becomes more difficult as m and k increase. When comparing the initialization strategies, HS and GR outperform RND.

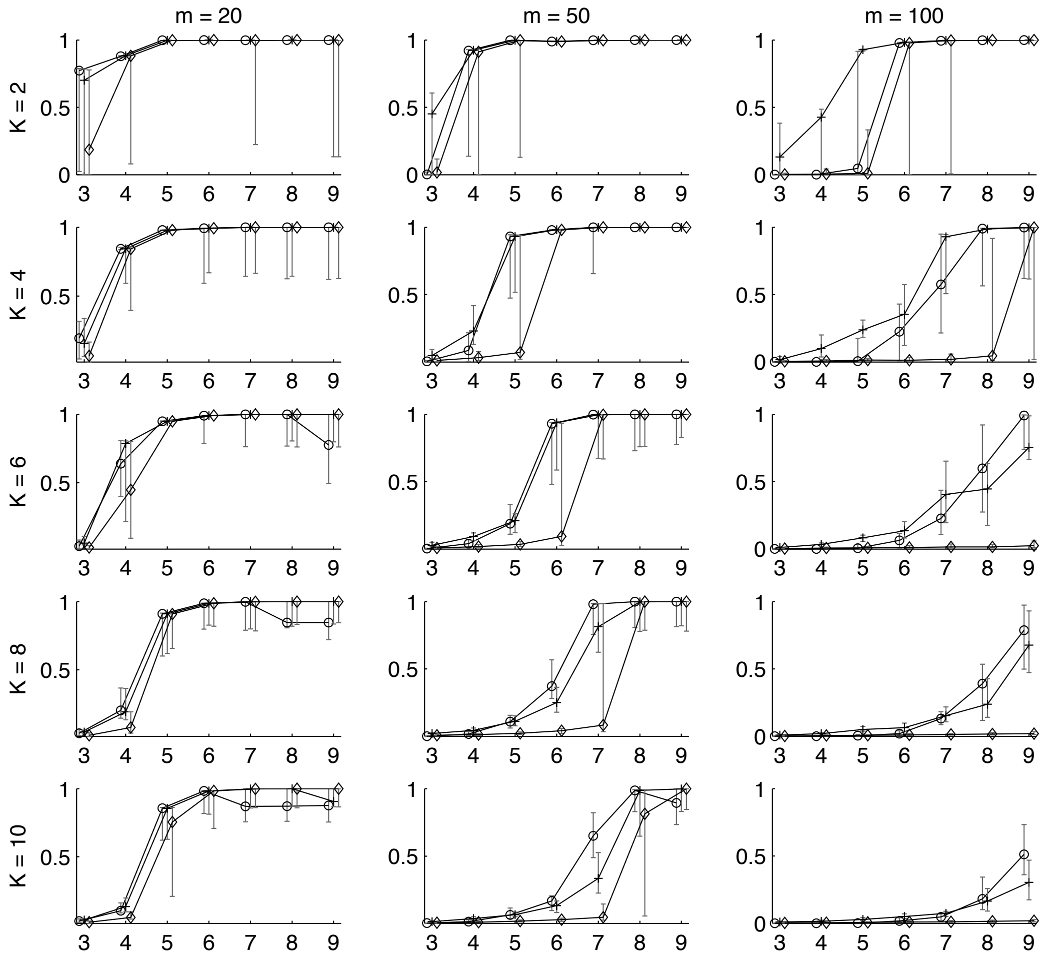


Figure 1: The Adjusted Rand Index (median over 25 trials) between a recovered clustering and the true clustering as a function of the length of a chain in random data sets consisting of 2000 chains each. Initialization methods are ○: GR, +: HS, and ◇: RND. Gray lines indicate 95 percent confidence intervals.

5.2.3 COMPARING AGAINST EXISTING METHODS

We compared how our approach using the HS initialization compares with existing algorithms. The HS-based variant was chosen because of fairness: The process we use to generate artificial data exactly matches the assumption underlying the GR approach, and hence may give this algorithm an unfair advantage. Also, the HS initialization is faster to compute.

Results are shown in Figure 2 for $m = 10$ and $m = 100$, and $k \in \{2, 6, 10\}$. The total number of items m has a strong effect on the performance. As above, the problem or recovering the clustering becomes harder as m increases and l decreases. Our algorithm suffers from very poor performance

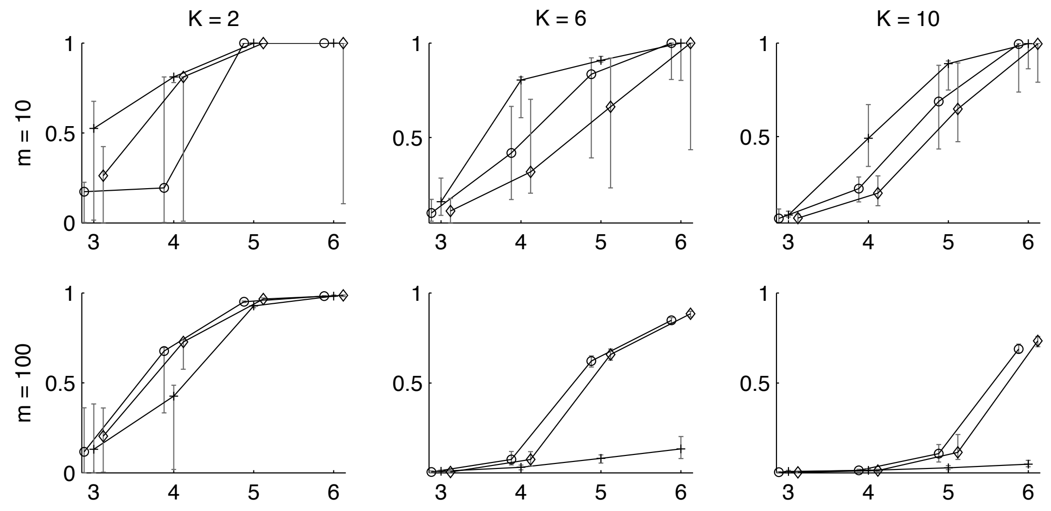


Figure 2: The Adjusted Rand Index (median over 25 trials) between a recovered clustering and the true clustering as a function of the length of a chain. Labels are: $+$: our algorithm initialized using HS, $\circ$: EBC, $\diamond$: TMSE.

with $m = 100$, while the EBC and TMSE algorithms can recover the planted clustering rather well also in this case. In contrast, for $m = 10$ and small l, our approach yields better results especially for $k > 2$. Recall that our algorithm relies on the pairwise probabilities of one item to precede an other. When $m = 100$ we have 4950 distinct pairs of items, when $m = 10$ this number is merely 45. With a large m it is therefore likely that our estimates of the pairwise probabilities are noisy simply because there are less observations of individual pairs since the input size is fixed. By increasing the size of the input these estimates should become more accurate.

We tested this hypothesis by running an experiment with random data sets ten times larger, that is, with an input of 20000 chains on $m = 100$ items. We concentrated on two cases: $k = 2$ with $l = 4$, and $k = 6$ with $l = 6$. The first corresponds to a situation where there is a small gap between the performance of TMSE/EBC and our method, and all algorithms show mediocre performance (see Fig. 2, 2nd row, left column). The second combination of k and l covers a case where this gap is considerably bigger, and TMSE/EBC both do rather well in recovering the planted clustering (see Fig. 2, 2nd row, middle column). Results are shown in Table 2. Increasing the size of the input leads to a considerable increase in performance of our algorithm. This suggests that for large data sets the approach based on pairwise probabilities may yield results superior to those obtained with existing algorithms.

5.2.4 UNKNOWN SIZE OF TRUE CLUSTERING

So far we have only considered cases where $k = \kappa$, that is, the algorithms were given the correct number of clusters. When analyzing real data κ is obviously unknown. We studied the algorithms' sensitivity to the value of k. Figure 3 shows the Adjusted Rand Index for our algorithm with HS initialization, and the EBC and TMSE algorithms when $m = 20$, and $\kappa = 6$. All three algorithms

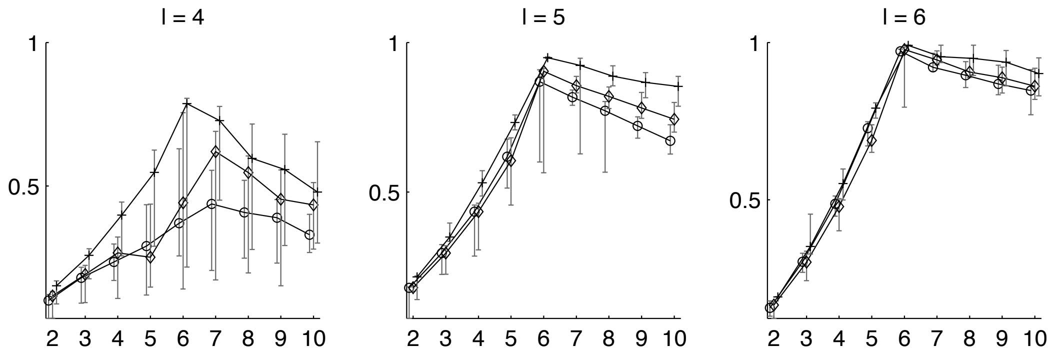


Figure 3: Adjusted Rand Index (median over 25 trials) as a function of k for different algorithms. We have $\kappa = 6$, and $m = 20$ in each case, the value of l is shown above each curve. Labels are: $+$: our algorithm initialized using HS, $\circ$: EBC, $\diamond$: TMSE.

perform similarly. For short chains ($l = 4$) the differences are somewhat stronger. While our HS-based method seems to be marginally better in recovering the original clustering, there is a lot of overlap in the confidence intervals, and none of the algorithms is able to find the true clustering exactly. We also acknowledge that the stronger performance of our method with $l = 5$ and $l = 6$ may be attributed to an implementation detail: Our algorithm is not guaranteed to return k clusters, it may return a number less than k if one of the clusters becomes empty during the computation. It is not clear how the implementations of TMSE and EBC deal with empty clusters.

5.3 Experiments with Real Data

This experiment was carried out by computing a k-way clustering of each data set described in Section 5.1 with k ranging from 2 to 10. Performance is measured by the clustering error as defined in Equation 1, using the centroid and distance function that are described in Section 2.3. Each combination of algorithm, data, and k was repeated 25 times with a randomly chosen initial clustering. (Note that even if we initialize our method by computing a clustering using either of the vector space representations, the algorithms that compute these must be initialized somehow.)

Figure 4 shows the reconstruction error as a function of k. Note that values on the y-axis have been normalized by the baseline error of having all chains in the same cluster. The error bars indicate 95 percent confidence intervals. The EBC algorithm is omitted from the figures, as this method was consistently outperformed by the TMSE algorithm. This result is also in line with previous empirical

	EBC		TMSE		HS init.	
$k = 2, l = 4$	0.817	$(0.816, 0.822)$	0.818	$(0.816, 0.822)$	**0.891**	$(0.891, 0.892)$
$k = 6, l = 6$	0.935	$(0.932, 0.938)$	0.937	$(0.934, 0.939)$	**0.974**	$(0.973, 0.976)$

Table 2: Adjusted Rand Index (median over 25 trials) for different methods computed from artificial data consisting of 20000 chains with $m = 100$ and the shown values for k and l. Numbers in parenthesis indicate 95 percent confidence intervals.

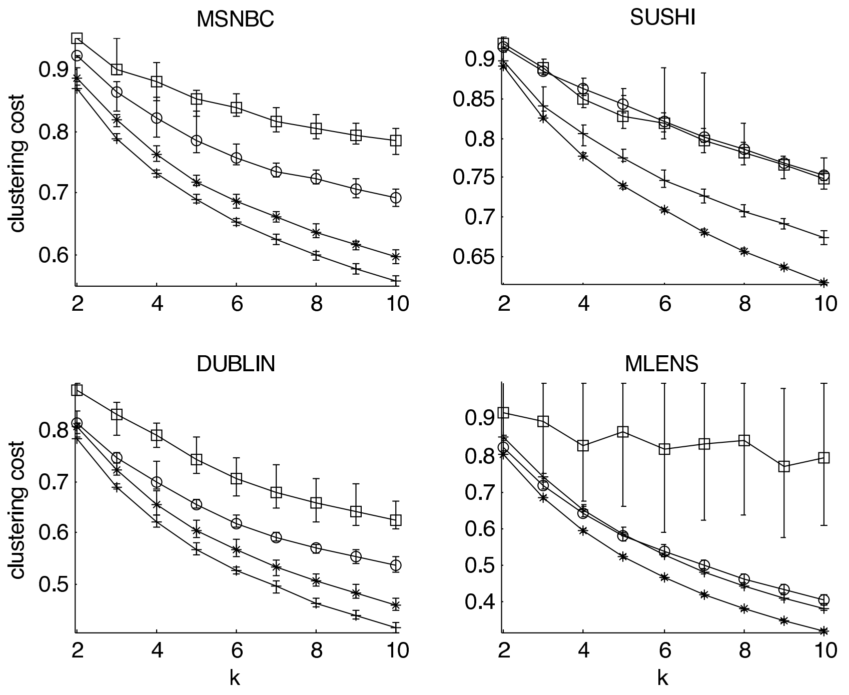


Figure 4: Reconstruction error as expressed in Equation 1 with the definitions of distance and centroid from Section 2.3 as a function of k. The y-axis has been normalized to show the error as a fraction of the baseline error of $k = 1$. Legend: $\square$: *only* standard k-means in GR representation, $\circ$: *only* standard k-means in HS representation, $+$: our variant of Lloyd's algorithm with RND init., $*$: the TMSE algorithm.

evidence reported by Kamishima and Akaho (2006). We also left out results obtained with our algorithm using either of the vector space representations to compute an initial clustering. (The curves for HS and GR therefore show performance that is obtained simply by mapping chains to the respective vector spaces and running standard k-means.) Contrary to random data (see results of Section 5.2), these initialization strategies did not give significantly better results than simple random initialization.

Our k-means procedure outperforms the TMSE algorithm with the MSNBC and DUBLIN data sets. With SUSHI and MLENS the situation is reversed. This statement holds for all values of k, and seems robust as the confidence intervals do not overlap. Also, when measuring clustering quality in this way, the results obtained by using only the vector space representations are considerably inferior to the other methods. Of course this is not an entirely fair comparison as the objective functions differ. In Figure 5 we plot the reconstruction error computed with the distance function and centroid representation used by TMSE. For details, please see Kamishima and Akaho (2006, Section 3.1). Using this measure, the SUSHI and MLENS data sets demonstrate an even stronger difference between the methods. With MSNBC and DUBLIN our algorithm continues to perform somewhat better, albeit this time the confidence intervals overlap. Interestingly, if an algorithm is better, it is better independent of the cost function used to evaluate the result. For instance, with MSNBC and DUBLIN our algorithm marginally outperforms TMSE even in terms of TMSE's own

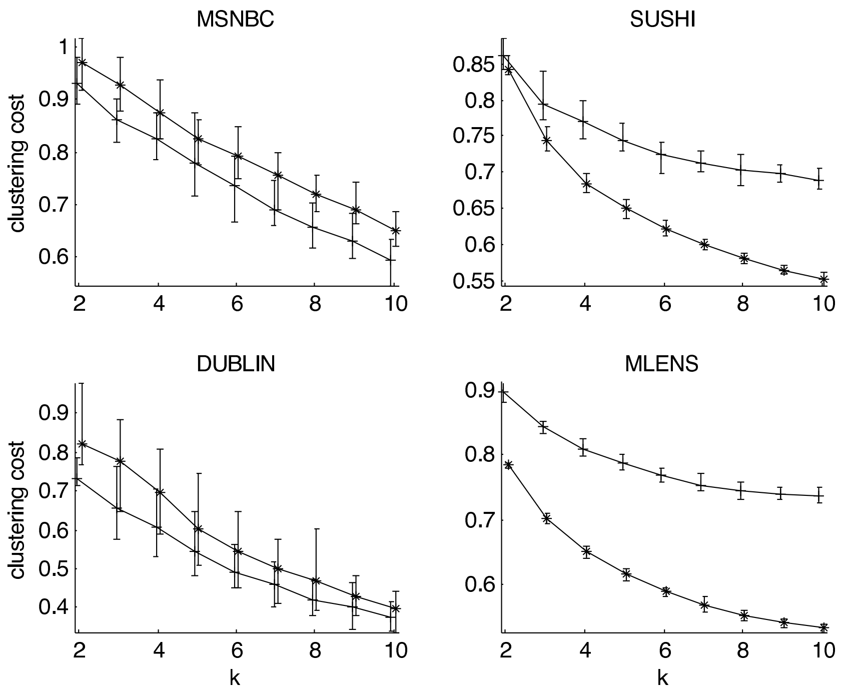


Figure 5: Reconstruction error of a clustering as expressed in Equation 1 with the definitions of distance and centroid from Kamishima and Akaho (2006) as a fraction of the baseline error of $k = 1$. Legend: $+$: our algorithm with random initialization, $*$: the TMSE algorithm.

cost function, and vice versa with SUSHI and MLENS. These results are in line with the ones we obtain with artificial data. As can be seen in Table 1, the data sets MSNBC and DUBLIN have a considerably smaller m. The experiments in Section 5.2.3 suggest that by collecting more data we could improve our result for the SUSHI and MLENS data sets.

5.4 Testing Clustering Validity

We use the randomization method of Section 4 to test the interestingness of the found clusterings. A clustering is assumed to be interesting if its test statistic substantially differs from the ones we obtain from randomized data. The test statistic we use is the reconstruction error given in Equation 1. The methods use their respective definitions of distance and centroid to compute the error.

To carry out the test, we must first estimate how many swaps are needed to obtain a single sample that is uncorrelated with the original data. To this end we run the SWAP-PAIRS algorithm for 10×10^6 swaps on each data set and measure $\delta(D, D_i)$ every 0.1×10^6 swaps. The assumption is that the data D_i are uncorrelated with the initial state D when $\delta(D, D_i)$ no longer increases. Figure 6 shows how the distance $\delta(D, D_i)$ develops with the number of swaps i for the data sets. From this we can read the number of swaps that are needed to obtain approximately uncorrelated samples. For SUSHI and MLENS the Markov chain seems to converge after approximately 5×10^6 swaps, for DUBLIN the distance $\delta(D, D_i)$ stabilizes already after about 0.5×10^6 swaps, while with MSNBC this happens after roughly 3×10^6 swaps. The randomized data used in the remaining analysis are

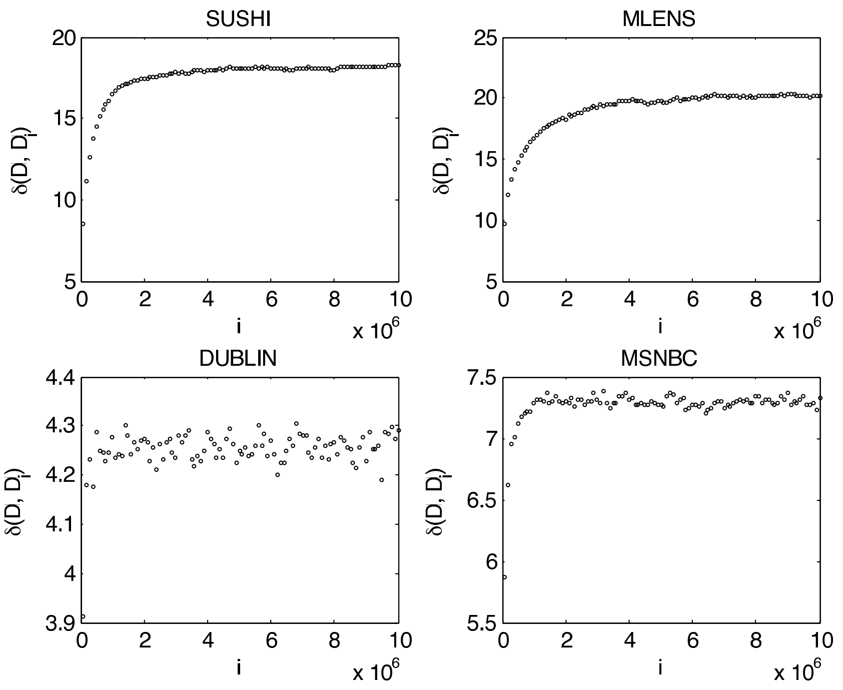


Figure 6: The distance $\delta(D, D_i)$ as a function of the number of swaps i.

computed using these swap counts, respectively. Here we also want to point out that randomization is computationally intensive. The table below shows the times to perform 10×10^6 swaps for the different data sets.

	SUSHI	MLENS	DUBLIN	MSNBC
t (seconds)	297	670	73	81

We observe that n, the number of chains in the input, does not affect the running time t, but the number of items m plays a significant part. (See also Table 1.)

For the actual analysis we sample 99 random instances from the equivalence class of each data set, and compare the test statistic with the one obtained from real data. Figure 7 shows the histogram of the reconstruction error in randomized data together with the minimum, maximum, and median error over 25 trials with real data. If this interval is clearly to the left of the histogram, it is unlikely to observe an error of the same magnitude in randomized data. If the interval overlaps with the histogram, the results should be considered as not significant according to this test.

In general the results suggest that the clusterings we obtain from the actual data sets have a smaller reconstruction error than a clustering computed with the same algorithm from a randomized data. There are some interesting exceptions, however. For MSNBC, SUSHI, and DUBLIN the clusterings obtained by our method from real data seem considerably better than those we obtain in random data, independent of K. In case of MLENS the results are clearly not significant for any K. For the TMSE algorithm the test suggests a significant outcome in case of SUSHI and MLENS, while for MSNBC and DUBLIN the clustering from real data is not considerably better than a clustering from randomized data.

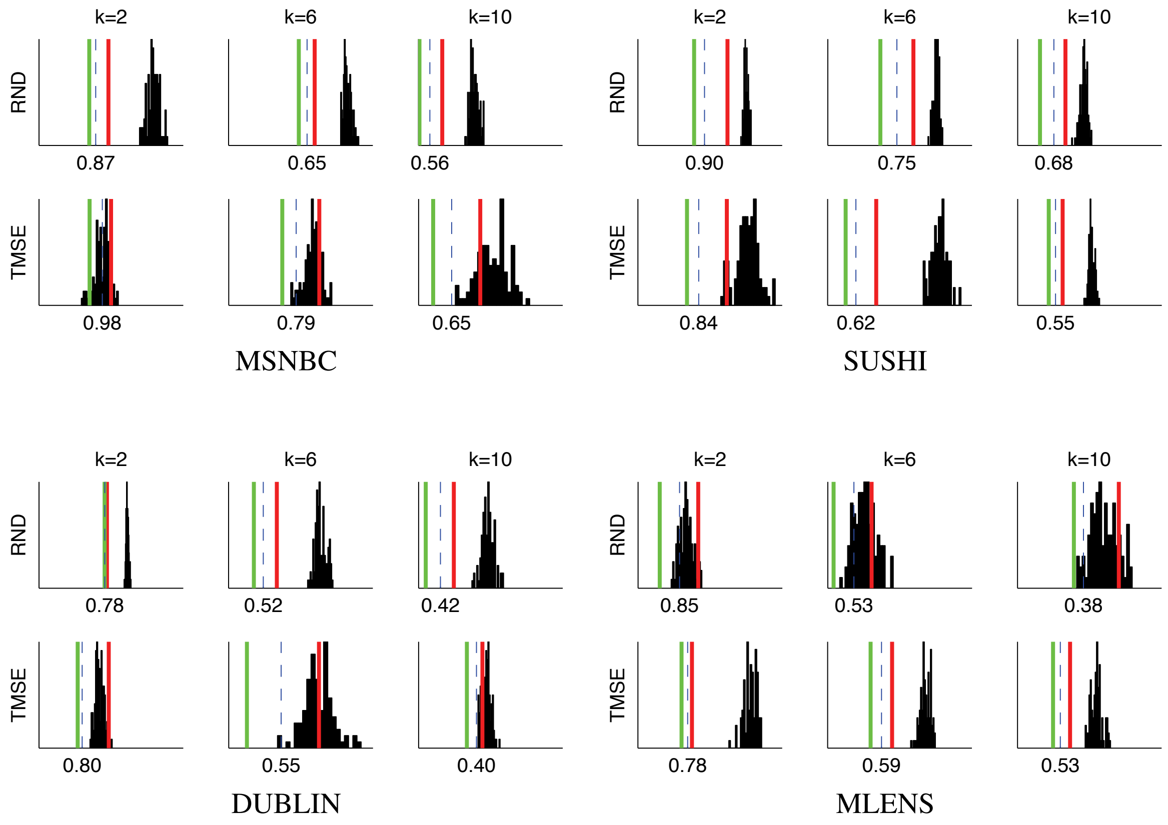


Figure 7: Results of randomization testing with our algorithm using RND initialization and the TMSE algorithm for different values of K. The numbers are normalized by the clustering error for $K = 1$. The histograms show the distribution of the clustering error on the randomized data. The light gray (green online), dashed, and dark gray (red online) lines indicate the minimum, median, and maximum of the clustering error on the original data. Both algorithms use their own cost functions.

6. Conclusion

We have discussed the problem of clustering chains. First, in Section 2 we gave simple definitions of a centroid and a distance function that can be used together with Lloyd's algorithm (k-means) for computing a clustering directly using chains. In Section 3 we gave two methods for mapping chains to a high-dimensional vector space. These representations have the advantage that any clustering algorithm can be used. Moreover, a clustering obtained in this way can still be further refined using the technique of Section 2. Mapping chains to vector spaces is an interesting subject in its own right and can have many other uses in addition to clustering. For example, they can be used to visualize of sets of chains, as was done by Ukkonen (2007), as well as by Kidwell et al. (2008). Also, we believe that the connections to the planted partition model (Condon and Karp, 2001; Shamir and Tsur, 2002) are very interesting at least from a theoretical point of view.

We also proposed a method for testing if a clustering found in a set of chains is any different from a clustering of random data. If the value of a suitable test statistic, such as the reconstruction error, does not substantially differ between the original input and the randomized data sets the clustering found in real data is probably not very meaningful. To this end we devised an MCMC algorithm for sampling sets of chains that all belong to the same equivalence class as a given set of chains.

In the experiments we compared our methods with the TMSE and EBC algorithms by Kamishima and Akaho (2009). We observe that for some data sets our algorithm yields better results, while for some other data sets the TMSE algorithm is a preferred choice. Interestingly, these differences can also be seen in the randomization tests. When an algorithm performs poorly, the results tend to be not significant according to the randomization test. Moreover, it seems that in cases where the TMSE algorithm is superior, our algorithm does not have enough data to perform well. Experiments on artificial data indicate that as the size of the input is increased (and other variables left unchanged), the performance of our algorithm increases considerably, and even outperforms the TMSE algorithm. Therefore, we suspect that by increasing data size we could improve the performance of our algorithm also with real data.

The main difference between the algorithms is the notion of distance. TMSE essentially uses a modified version of Spearman's rank correlation coefficient that is a "positional" distance for permutations, as it only considers the positions in which different items appear. We propose a "pairwise" distance that considers how pairs of items are related to each other. The experiments suggest that the pairwise approach is more powerful as long as there is enough data, but for smaller data sets positional distances seem more robust. Finding the tipping point in terms of input size and other data parameters where the pairwise approach becomes favorable over positional distances is an interesting open question.

Acknowledgments

I would like to thank the anonymous reviewers for their valuable feedback that helped to improve this manuscript considerably. This work was partly funded by the Academy of Finland (grant 131276).

Appendix A. Proofs of Theorems

Proofs of Theorem 2 and Theorem 5 are given below.

A.1 Proof of Theorem 2

The proof is a simple matter of upper bounding Equation 6. First we note that using Vandermonde's convolution (Graham et al., 1994, Equation 5.22) the sum in Equation 6 can be rewritten as

$$\binom{m}{l} - \underbrace{\left(\binom{l}{1}\binom{m-l}{l-1} + \binom{m-l}{l}\right)}_{A}.$$

Essentially Vandermonde's convolution states that $\sum_{i=0}^{l}\binom{l}{i}\binom{m-l}{l-i} = \binom{m}{l}$, and we simply subtract the first two terms indicated by A, because above the sum starts from $i = 2$. Using simple manipulations we obtain

$$A = \binom{m-l}{l}\Big(\frac{l^2}{m-2l+1}+1\Big),$$

which gives the following:

$$p = \binom{m}{l}^{-1}\Big(\binom{m}{l} - \binom{m-l}{l}\Big(\frac{l^2}{m-2l+1}+1\Big)\Big).$$

With $l < m/2$ the part $\frac{l^2}{m-2l+1}+1$ is lower bounded by 1, and we have

$$\begin{aligned}
p &< \binom{m}{l}^{-1}\Big(\binom{m}{l} - \binom{m-l}{l}\Big) = 1 - \binom{m}{l}^{-1}\binom{m-l}{l} \\
&= 1 - \frac{(m-l)!}{l!(m-2l)!}\cdot\frac{l!(m-l)!}{m!} \\
&= 1 - \frac{(m-l)(m-l-1)\cdots(m-2l+1)}{m(m-1)\cdots(m-l+1)} \\
&< 1 - \frac{(m-l)(m-l-1)\cdots(m-2l+1)}{m^l} \\
&< 1 - \frac{(m-2l+1)^l}{m^l} < \frac{m^l-(m-2l)^l}{m^l}.
\end{aligned}$$

We can factor $m^l-(m-2l)^l$ as follows:

$$\begin{aligned}
m^l-(m-2l)^l &= (m-(m-2l))\Big(m^{l-1}(m-2l)^0 + m^{l-2}(m-2l)^1 + \dots \\
&\quad \dots + m^1(m-2l)^{l-2} + m^0(m-2l)^{l-1}\Big) \\
&= 2l\sum_{i=0}^{l-1} m^{l-1-i}(m-2l)^i.
\end{aligned}$$

Using this we write

$$\frac{m^l-(m-2l)^l}{m^l} = 2l\sum_{i=0}^{l-1}(\frac{1}{m})^l m^{l-1-i}(m-2l)^i.$$

Letting $a = l-1$ and taking one $\frac{1}{m}$ out of the sum we get

$$\begin{aligned}
\frac{1}{m}2(a+1)\sum_{i=0}^{a}(\frac{1}{m})^a m^{a-i}(m-2(a+1))^i &= \frac{1}{m}2(a+1)\sum_{i=0}^{a}(\frac{1}{m})^i(m-2(a+1))^i \\
&= \frac{1}{m}2(a+1)\sum_{i=0}^{a}(1-\frac{2(a+1)}{m})^i.
\end{aligned}$$

We assume $l = a+1$ is considerably smaller than m, and hence $(1-\frac{2(a+1)}{m})^i$ is at most 1. There are $a+1$ terms in the sum, so the above is upper bounded by $\frac{1}{m}2(a+1)(a+1) = 2\frac{l^2}{m}$, which concludes the proof of the theorem.

A.2 Proof of Theorem 5

Let $u \in \pi$: We start by showing that the claim of Equation 12 holds for all u that belong to π. That is, we will show that

$$\sum_{\tau \in \mathcal{E}(\pi)} f_\tau(u) = Q\big(-\frac{|\pi|+1}{2} + \pi(u)\big) \tag{17}$$

for all $u \in \pi$. First, note that $\sum_{\tau \in \mathcal{E}(\pi)} f_\tau(u)$ can be rewritten as follows

$$\sum_{\tau \in \mathcal{E}(\pi)} -\frac{m+1}{2} + \tau(u) = \sum_{i=\pi(u)}^{m-|\pi|+\pi(u)} \#\{\tau(u) = i\}\Big(-\frac{m+1}{2} + i\Big), \tag{18}$$

where $\#\{\tau(u) = i\}$ denotes the number of times u appears at position i in the linear extensions of π. The sum is taken over the range $\pi(u), \ldots, m - |\pi| + \pi(u)$, as $\tau(u)$ can not be less than $\pi(u)$, because the items that appear before u in π must appear before it in τ as well, likewise for the other end of the range.

To see what $\#\{\tau(u) = i\}$ is, consider how a linear extension τ of π is structured. When u appears at position i in τ, there are exactly $\pi(u) - 1$ items belonging to π that appear in the $i-1$ indices to the left of u, and $|\pi| - \pi(u)$ items also belonging to π that appear in the $m-i$ indices to the right of u. The ones on the left may choose their indices in $\binom{i-1}{\pi(u)-1}$ different ways, while the ones on the right may choose their indices in $\binom{m-i}{|\pi|-\pi(u)}$ different ways. The remaining items that do not belong to π are assigned in an arbitrary fashion to the remaining $m - |\pi|$ indices. We have thus,

$$\#\{\tau(u) = i\} = \binom{i-1}{\pi(u)-1}\binom{m-i}{|\pi|-\pi(u)}(m-|\pi|)!.$$

When this is substituted into the right side of (18), and after rearranging the terms slightly, we get

$$\sum_{\tau \in \mathcal{E}(\pi)} f_\tau(u) = (m-|\pi|)! \sum_{i=\pi(u)}^{m-|\pi|+\pi(u)} \binom{i-1}{\pi(u)-1}\binom{m-i}{|\pi|-\pi(u)}\Big(-\frac{m+1}{2} + i\Big).$$

This can be written as

$$\sum_{\tau \in \mathcal{E}(\pi)} f_\tau(u) = (m-|\pi|)!(S_1 + S_2), \tag{19}$$

where

$$\begin{aligned} S_1 &= -\frac{m+1}{2} \sum_{i=\pi(u)}^{m-|\pi|+\pi(u)} \binom{i-1}{\pi(u)-1}\binom{m-i}{|\pi|-\pi(u)}, \quad \text{and} \\ S_2 &= \sum_{i=\pi(u)}^{m-|\pi|+\pi(u)} i\binom{i-1}{\pi(u)-1}\binom{m-i}{|\pi|-\pi(u)}. \end{aligned}$$

Let us first look at S_2. The part $i\binom{i-1}{\pi(u)-1}$ can be rewritten as follows:

$$i\binom{i-1}{\pi(u)-1} = \frac{i\ (i-1)!}{(\pi(u)-1)!(i-\pi(u))!} \cdot \frac{\pi(u)}{\pi(u)} = \pi(u)\frac{i!}{\pi(u)!(i-\pi(u))!} = \pi(u)\binom{i}{\pi(u)}.$$

This gives

$$S_2 = \pi(u) \sum_{i=\pi(u)}^{m-|\pi|+\pi(u)} \binom{i}{\pi(u)} \binom{m-i}{|\pi|-\pi(u)} = \pi(u) \binom{m+1}{|\pi|+1},$$

where the second equality is based on Equation 5.26 in Graham et al. (1994). Next we must show that $\binom{m+1}{|\pi|+1}$ will appear in S_1 as well. We can rewrite the sum as follows:

$$\sum_{i=\pi(u)}^{m-|\pi|+\pi(u)} \binom{i-1}{\pi(u)-1} \binom{m-i}{|\pi|-\pi(u)} = \sum_{i=\pi(u)-1}^{m-|\pi|+\pi(u)-1} \binom{i}{q} \binom{r-i}{p-q},$$

where $q = \pi(u)-1$, $r = m-1$ and $p = |\pi|-1$. Again we apply Equation 5.26 of Graham et al. (1994) to get

$$S_1 = -\frac{m+1}{2} \binom{r+1}{p+1} = -\frac{m+1}{2} \binom{m}{|\pi|},$$

which we multiply by $\frac{|\pi|+1}{|\pi|+1}$ and have

$$S_1 = -\frac{|\pi|+1}{2} \cdot \frac{m+1}{|\pi|+1} \binom{m}{|\pi|} = -\frac{|\pi|+1}{2} \binom{m+1}{|\pi|+1}.$$

When S_1 and S_2 are substituted into (19) we have

$$\sum_{\tau \in \mathcal{E}(\pi)} f_\tau(u) = (m-|\pi|)! \Big(-\frac{|\pi|+1}{2} \binom{m+1}{|\pi|+1} + \pi(u) \binom{m+1}{|\pi|+1} \Big),$$

which is precisely Equation 17 when we let $Q = (m-|\pi|)!\binom{m+1}{|\pi|+1}$.

Let $u \notin \pi$: To complete the proof we must still show that Equation 12 also holds for items u that do not appear in the chain π. For such u we have $f_\pi(u) = 0$ by definition. Since we showed above that $Q > 0$, we have to show that $\sum_{\tau \in \mathcal{E}(\pi)} f_\tau(u) = 0$ when $u \notin \pi$ to prove the claim.

We'll partition $\mathcal{E}(\pi)$ to disjoint groups defined by index sets I. Let $S(I)$ denote the subset of $\mathcal{E}(\pi)$ where the items that belong to π appear at indices $I = \{i_1, \ldots, i_{|\pi|}\}$. Furthermore, let $I^R = \{m-i_1+1, \ldots, m-i_{|\pi|}+1\}$. See Figure 8 for an illustration of the structure of the permutations that belong to $S(I)$ and $S(I^R)$.

Now we can write for every $u \notin \pi$:

$$\sum_{\tau \in \mathcal{E}(\pi)} f_\tau(u) = \frac{1}{2} \sum_I \sum_{\tau \in \{S(I) \cup S(I^R)\}} f_\tau(u). \tag{20}$$

That is, we first sum over all possible index sets I, and then sum over all τ that belong to the union of $S(I)$ and $S(I^R)$. Each I is counted twice (once as I and once as I^R), so we multiply the right hand side by $\frac{1}{2}$. To make sure that Equation 20 equals zero, it is enough to show that $\sum_{\tau \in \{S(I) \cup S(I^R)\}} f_\tau(u) = 0$ for each I.

Note that we have $f_\tau(u) + f_{\tau^R}(u) = 0$ because $\tau^R(u) = m - \tau(u) + 1$. That is, the values at indices j and $m-j+1$ cancel each other out. This property will give us the desired result if we can show that for each permutation $\tau \in \{S(I) \cup S(I^R)\}$ where an item $u \notin \pi$ appears at position j, there exists a corresponding permutation τ', also in $\{S(I) \cup S(I^R)\}$, where u appears at position $m-j+1$. Denote by $\#(S,u,j)$ the size of the set $\{\tau \in S \mid \tau(u) = j\}$.

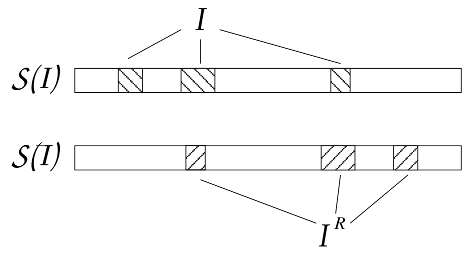


Figure 8: Permutations in $S(I)$ have the positions I occupied by items that belong to the chain π, while permutations in $S(I^R)$ have the positions I^R occupied by items of π. See proof of Theorem 5.

An index is *free* if it *does not* belong to the set $\{I \cup I^R\}$. Let j be a free index. By definition of the sets I and I^R, $m-j+1$ is also a free index. We have $\#(S(I),u,j) = \#(S(I),u,m-j+1)$. This holds for $S(I^R)$ as well. As a consequence, when we sum over all permutations in $\{S(I) \cup S(I^R)\}$, the values corresponding to index j and $m-j+1$ cancel each other out because u appears equally many times at positions j and $m-j+1$. The total contribution to the sum $\sum_{\tau \in \{S(I) \cup S(I^R)\}} f_\tau(u)$ of u appearing at the free indices is therefore zero.

Let j belong to I, meaning it is not free. By definition of the sets I and I^R, the index $m-j+1$ now belongs to I^R, and is also not free. However, because of symmetry we have $\#(S(I^R),u,j) = \#(S(I),u,m-j+1)$. That is, the number of times the item u appears at position j in a permutation belonging to $S(I^R)$ is the same as the number of times it appears at position $m-j+1$ in a permutation belonging to $S(I)$. When we sum over the permutations in $\{S(I) \cup S(I^R)\}$, the values of u appearing at position j in $S(I^R)$ are cancelled out by the values of u appearing at position $m-j+1$ in $S(I)$. The total contribution to the sum $\sum_{\tau \in \{S(I) \cup S(I^R)\}} f_\tau(u)$ of u appearing at an index in I is therefore zero as well. This concludes the proof of Theorem 5.

References

N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 684–693, 2005.

E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.

D Arthur and S Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

G. H. Ball and D. J. Hall. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12:153–155, 1967.

A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the Sixth Annual International Conference on Computational Biology*, pages 49–57, 2002.

P Berkhin. *Grouping Multidimensional Data*, chapter A Survey of Clustering Data Mining Techniques, pages 25–71. Springer, 2006.

J. Besag and P. Clifford. Generalized Monte Carlo significance tests. *Biometrika*, 76(4):633–642, 1989.

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

L. M. Busse, P. Orbanz, and J. M. Buhmann. Cluster analysis of heterogeneous rank data. In *Proceedings of the 24th international conference on Machine learning*, pages 113–120, 2007.

S Clémençon and J Jakubowicz. Kantorovich distances between rankings with applications to rank aggregation. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010*, 2010.

A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 776–782, 2006.

D. Critchlow. *Metric Methods for Analyzing Partially Ranked Data*, volume 34 of *Lecture Notes in Statistics*. Springer-Verlag, 1985.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference*, 2001.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Texts in Statistical Science. Chapman & Hall, CRC, 2004.

A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data*, 1(3), 2007.

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

R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 2nd edition, 1994.

D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2001.

T. Kamishima and S. Akaho. Efficient clustering for orders. In *Workshops Proceedings of the 6th IEEE International Conference on Data Mining*, pages 274–278, 2006.

T. Kamishima and S. Akaho. *Mining Complex Data*, volume 165 of *Studies in Computational Intelligence*, chapter Efficient Clustering for Orders, pages 261–279. Springer, 2009.

T. Kamishima and J. Fujiki. Clustering orders. In *Proceedings of the 6th International Conference on Discovery Science*, pages 194–207, 2003.

P Kidwell, G Lebanon, and W S Cleveland. Visualizing incomplete and partially ranked data. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1356–1363, 2008.

H Lawrence and A Phipps. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982.

H. Moulin. *Axioms of Cooperative Decision Making*. Cambride Universiy Press, 1991.

T. B. Murphy and D. Martin. Mixtures of distance-based models for ranking data. *Computational Statistics & Data Analysis*, 41:645–655, 2003.

W M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

R. Shamir and D. Tsur. Improved algorithms for the random cluster graph model. In *Proceedings of Scandanavian Workshop on Algorithms Theory*, pages 230–239, 2002.

N. Tideman. The single transferable vote. *Journal of Economic Perspectives*, 9(1):27–38, 1995.

A. Ukkonen. Visualizing sets of partial rankings. In *Advances in Intelligent Data Analysis VII*, pages 240–251, 2007.

A. Ukkonen. *Algorithms for Finding Orders and Analyzing Sets of Chains*. PhD thesis, Helsinki University of Technology, 2008.

A. Ukkonen and H. Mannila. Finding outlying items in sets of partial rankings. In *Knowledge Discovery in Databases: PKDD 2007*, pages 265–276, 2007.

R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

Journal of Machine Learning Research 12 (2011) 1425-1428 Submitted 3/11; Published 4/11

Introduction to the Special Topic on Grammar Induction, Representation of Language and Language Learning

Dorota Głowacka D.GLOWACKA@CS.UCL.AC.UK
John Shawe-Taylor JST@CS.UCL.AC.UK
Department of Computer Science
University College London
London WC1E 6BT
United Kingdom

Alexander Clark ALEXC@CS.RHUL.AC.UK
Department of Computer Science,
Royal Holloway, University of London
Egham, Surrey, TW20 0EX
United Kingdom

Colin de la Higuera CDLH@UNIV-NANTES.FR
Laboratoire LINA
University of Nantes
44322 Nantes
France

Mark Johnson MARK.JOHNSON@MQ.EDU.AU
Department of Computing
Macquarie University
Sydney NSW 2109
Australia

Editor: Lawrence Saul

Abstract

Grammar induction refers to the process of learning grammars and languages from data; this finds a variety of applications in syntactic pattern recognition, the modeling of natural language acquisition, data mining and machine translation. This special topic contains several papers presenting some of recent developments in the area of grammar induction and language learning, as applied to various problems in Natural Language Processing, including supervised and unsupervised parsing and statistical machine translation.

Keywords: machine translation, Bayesian inference, grammar induction, natural language parsing

1. Introduction

Grammar induction was the subject of an intense study in the early days of Computational Learning Theory, with the theory of query learning (Angluin, 1988) largely developing out of this research. More recently the study of new methods of representing language and grammars through complex kernels and probabilistic modelling together with algorithms such as structured output learning has enabled machine learning methods to be applied successfully to a range of language related tasks from simple topic classification through parts of speech tagging to statistical machine translation. These methods typically rely on more fluid structures than those derived from formal grammars and yet are able to compete favourably with classical grammatical approaches that require significant input from domain experts, often in the form of annotated data and hand-coded rules.

2. JMLR Special Topic

This special topic arose from a NIPS 2009 workshop on ”Grammar Induction, Representation of Language and Language Learning” held at the Whistler Resort, Vancouver, Canada. Contributions to the special topic were also open to researchers who had not presented their work at the workshop. We received thirteen submissions and after considering the reviews for each submission, we selected five papers to be included in this special topic.

Probabilistic grammars offer great flexibility in modeling discrete sequential data like natural language text. Recently, there has been an increased interest in using probabilistic grammars in the Bayesian setting, focusing mostly on the use of a Dirichlet prior. Cohen and Smith (2010) propose a family of logistic normal distributions as an alternative to the Dirichlet prior. A variational inference algorithm for estimating the parameters of the probabilistic grammar provides a fast, parallelizable, and deterministic alternative to MCMC methods to approximate the posterior over derivations and grammar parameters. Experiments with dependency grammar induction on six different languages demonstrate performance improvements with the new priors. The experiments include a novel promising setting, in which syntactic trees are inferred in a bilingual setting that uses multilingual, non-parallel corpora. Notably, the proposed approach tends to generalize better to longer sentences, despite learning on short sentences.

Despite decades of research, inducing a grammar from text has proven to be a notoriously challenging learning task. The majority of existing work on grammar induction has favoured model simplicity (and thus learnability) over representational capacity by using context free grammars and first order dependency grammars, which are not sufficiently expressive to model many common linguistic constructions. Cohn, Blunsom, and Goldwater (2010) propose a novel compromise by inferring a Probabilistic Tree Substitution Grammar (PTSG), a formalism which allows for arbitrarily large tree fragments and thereby better represents complex linguistic structures. A PTSG is an extension to the Probabilistic Context Free Grammar (PCFG) in which nonterminals can rewrite as entire tree fragments (elementary trees), not just immediate children. These large fragments can be used to encode non-local context, such as argument frames, gender agreement and idioms. The model's complexity is reduced by employing a Bayesian non-parametric prior which biases the model towards a sparse grammar with shallow productions. The experimental results demonstrate the model's efficacy on supervised phrase-structure parsing, where a latent segmentation of the training treebank is induced, and on unsupervised dependency grammar induction. In both cases the model uncovers interesting latent linguistic structures while producing competitive results.

Henderson and Titov (2010) propose a new class of graphical models for structured prediction problems called incremental sigmoid belief networks (ISBNs) and apply it to natural language grammar learning. ISBNs make decoding possible because inference with partial output structures does not require summing over the unboundedly many compatible model structures, due to their directed edges and incrementally specified model structure. ISBNs are particularly applicable to natural language parsing, where learning the domain's complex statistical dependencies benefits from large numbers of latent variables. Exact inference in ISBNs is not tractable, but two efficient approximations are proposed: a coarse mean-field approximation and a feed-forward neural network approximation. Experimental results show that these models achieve accuracy competitive with the state-of-the-art.

Machine translation is a challenging problem in artificial intelligence. Natural languages are characterised by large variabilities of expressions, exceptions to grammatical rules and context dependent changes, making automatic translation a very difficult task. While early work in machine translation was dominated by rule based approaches (Bennett and Slocum, 1985), the availability of large corpora has paved the way for statistical methods to be applied. Ni, Saunders, Szedmak, and Niranjan (2011) propose a distance phrase reordering model (DPR) for statistical machine translation, where the aim is to learn the grammatical rules and context dependent changes using a phrase reordering classification framework. Techniques are compared and evaluated on a Chinese-English corpus, a language pair known for the high reordering characteristics which cannot be adequately captured with current models. In the reordering classification task, the method significantly outperforms the baseline against which it was tested, and further, when integrated as a component of the state-of-the-art machine translation system, MOSES, it achieves improvement in translation results.

Gillenwater, Ganchev, Graça, Pereira, and Taskar (2011) present a new method for unsupervised learning of dependency parsers. In contrast with previous approaches that impose a sparsity bias on the model parameters using discounting Dirichlet distributions, the proposed technique imposes a sparsity bias on the model posteriors. This is done by using the posterior regularization (PR) framework (Graça et al., 2007) with constraints that favor posterior distributions that have a small number of unique parent-child relations. In experiments with 12 different languages, the proposed method achieves significant gains in directed accuracy over the standard expectation maximization (EM) baseline for 9 of the languages, while for 8 out of 12 languages, the new technique outperforms models based on standard Bayesian sparsity-inducing parameter priors.

3. Concluding Remarks

We feel these papers provide a useful snapshot of the current state-of-the-art techniques being employed by researchers in the fields of grammar induction, language parsing, machine translation and related areas.

Acknowledgments

We are much indebted to our authors and reviewers, who put a tremendous amount of effort and dedication to make this issue a reality. We are also thankful to the JMLR editorial team for their valuable support.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2:319 – 342, 1988.

W. S. Bennett and J. Slocum. The lrc machine translation system. *Computational Linguistics*, 11: 111 – 121, 1985.

S. B. Cohen and N. A. Smith. Covariance in unsupervised learning of probabilistic grammars. *Journal of Machine Learning Research*, 11:3017 – 3051, 2010.

T. Cohn, P. Blunsom, and S. Goldwater. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, 11:3053 – 3096, 2010.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. Posterior sparsity in unsupervised dependency parsing. *Journal of Machine Learning Research*, 11, 2011.

J. Graça, K. Ganchev, and B. Taskar. Expectation maximization and posterior constraints. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2007. MIT Press.

J. Henderson and I. Titov. Incremental sigmoid belief networks for grammar learning. *Journal of Machine Learning Research*, 11:3541 – 3570, 2010.

Y. Ni, C. Saunders, S. Szedmak, and M. Niranjan. Exploitation of machine learning techniques in modelling phrase movements for machine translation. *Journal of Machine Learning Research*, 12:1 – 30, 2011.

Journal of Machine Learning Research 12 (2011) 1429-1458 Submitted 6/10; Revised 12/10; Published 5/11

Learning a Robust Relevance Model for Search Using Kernel Methods

Wei Wu[*] V-WEW@MICROSOFT.COM
MOE -Microsoft Key Laboratory of Statistics and Information Technology
Department of Probability and Statistics
Peking University
No.5 Yiheyuan Road, Haidian District, Beijing, 100871, P. R. China

Jun Xu JUNXU@MICROSOFT.COM
Hang Li HANGLI@MICROSOFT.COM
Microsoft Research Asia
13F Building 2
No. 5 Danling Street, Haidian District, Beijing, 100080, P.R. China

Satoshi Oyama [*] OYAMA@IST.HOKUDAI.AC.JP
Graduate School of Information Science and Technology
Hokkaido University
Kita 14, Nishi 9, Kita-ku, 060-0814, Japan

Editor: Corinna Cortes

Abstract

This paper points out that many search relevance models in information retrieval, such as the Vector Space Model, BM25 and Language Models for Information Retrieval, can be viewed as a similarity function between pairs of objects of different types, referred to as an S-function. An S-function is specifically defined as the dot product between the images of two objects in a Hilbert space mapped from two different input spaces. One advantage of taking this view is that one can take a unified and principled approach to address the issues with regard to search relevance. The paper then proposes employing a kernel method to learn a robust relevance model as an S-function, which can effectively deal with the term mismatch problem, one of the biggest challenges in search. The kernel method exploits a positive semi-definite kernel referred to as an S-kernel. The paper shows that when using an S-kernel the model learned by the kernel method is guaranteed to be an S-function. The paper then gives more general principles for constructing S-kernels. A specific implementation of the kernel method is proposed using the Ranking SVM techniques and click-through data. The proposed approach is employed to learn a relevance model as an extension of BM25, referred to as Robust BM25. Experimental results on web search and enterprise search data show that Robust BM25 significantly outperforms baseline methods and can successfully tackle the term mismatch problem.

Keywords: search, term mismatch, kernel machines, similarity learning, s-function, s-kernel

1. Introduction

There are many applications such as search, collaborative filtering, and image annotation, that can be viewed as a task employing a similarity function defined on pairs of instances from two different spaces. For example, search is a task as follows. Given a query, the system retrieves documents

$*$. This work was conducted when the first and fourth authors visited Microsoft Research Asia.

relevant to the query and ranks the documents based on the degree of relevance. The relevance of a document with respect to a query can be viewed as a kind of similarity, and the search task is essentially one based on a similarity function between query and document pairs, where query and document are instances from two spaces: query space and document space.

In this paper, we formally define the similarity function as the dot product of the images of two objects in a Hilbert space mapped from two different spaces. For simplicity, we call the similarity function S-function. In fact, the state-of-the-art relevance models in information retrieval (IR), such as the Vector Space Model (VSM) (Salton and McGill, 1986), BM25 (Robertson et al., 1994) and Language Models for Information Retrieval (LMIR) (Ponte and Croft, 1998; Zhai and Lafferty, 2004), are all S-functions. We prove some properties of the S-function and show that it becomes a positive semi-definite kernel under certain conditions. One advantage of taking this view to search is that it provides us with a unified and principled approach to using and learning relevance models.

In this paper, we focus on the learning of a robust relevance model as an S-function, to deal with term mismatch, one of the critical challenges for search. We show that we can define a new type of positive semi-definite kernel function called S-kernel and learn a robust relevance model using a kernel method based on S-kernel. Recently, the learning of similarity function has emerged as a hot research topic in machine learning (cf., Abernethy et al., 2009; Grangier and Bengio, 2008). Our work is novel and unique in that it learns a similarity function for search using a kernel method .

The conventional relevance models are all based on term matching. That is, they look at the matched words in a query and document, and calculate the similarity (relevance) based on the degree of matching. A good match at term level does not necessarily mean high relevance, however, and vice versa. For example, if the query is “NY” and the document only contains “New York”, then the BM25 score of the query and document pair will be low (i.e., the two will be viewed less relevant), although the query and document are relevant. Similar problems occur with LMIR and other relevance models. This is the so-called term mismatch problem, which all existing relevance models suffer from. In other words, the scores from the relevance models may not be reliable and the question of how to learn a more robust similarity function for search arises; this is exactly the problem we want to address in this paper.

In this paper, we tackle the term mismatch problem with a kernel method based on the notion of S-function. Intuitively, we calculate a more reliable score between a query document pair by using the scores between the pairs of similar query and similar document. Our kernel method exploits a special positive semi-definite kernel, referred to as S-kernel, defined based upon the S-function.

An S-kernel is formally defined as a positive semi-definite kernel such that the reproducing kernel Hilbert space (RKHS) generated by the kernel is also a space of S-functions. Therefore, the model learned by a kernel method is guaranteed to be an S-function. We further give general principles for constructing S-kernels, and thus offer a formulation for learning similarity functions with S-kernels. An S-kernel can be viewed as an extension of the hyper kernel proposed by Ong et al. (2005).

We provide a method for implementing the kernel method using the Ranking SVM techniques and click-through data. The method is used to train a relevance model named ‘Robust BM25’ to deal with term mismatch, as an extension of BM25. The learned Robust BM25 model determines the relevance score of a query document pair on the basis of not only the BM25 score of the query document pair, but also the BM25 scores of similar query and similar document pairs. All calculations are naturally incorporated in the kernel method. Experimental results on two large scale data sets show that Robust BM25 can indeed solve term mismatch and significantly outperform the baselines.

This paper has the following contributions: 1) proposal of a kernel method for dealing with term mismatch in search, 2) proposal of a unified view to search using S-function, 3) proposal of a family of kernel functions, S-kernel.

The rest of the paper is organized as follows. A survey of related work is conducted in Section 2, and then the definition of S-function and interpretation of traditional relevance models as S-functions are given in Section 3. Section 4 first introduces the term mismatch problem in search, and then proposes a kernel method for learning a robust relevance model to deal with the problem, such as Robust BM25. Section 5 defines S-kernel and proposes learning a similarity function with S-kernel. Section 6 describes how to implement the learning Robust BM25 method. Section 7 reports experimental results and Section 8 concludes this paper.

2. Related Work

Kernel methods, including the famous Support Vector Machines (SVM) (Vapnik, 1995), refer to a class of algorithms in machine learning which can be employed in a variety of tasks such as classification, regression, ranking, correlation analysis, and principle component analysis (Hofmann et al., 2008; Schölkopf and Smola, 2002). Kernel methods make use of kernel functions which map a pair of data in the input space (Euclidean space or discrete set) into the feature space (Hilbert space) and compute the dot product between the images in the feature space. Many kernels have been proposed for different applications (Zhou, 2004; Vishwanathan and Smola, 2004; Haussler, 1999; Watkins, 1999; Gartner et al., 2003; Kashima et al., 2004). Conventional kernels are defined over one single input space and are symmetric and positive semi-definite. The kernel function is called Mercer kernel when it is continuous. The similarity function, S-function, which we define in this paper, is related to the conventional kernel function. An S-function is defined as the dot product in a Hilbert space between the images of inputs from two spaces, and a conventional kernel function is defined as the dot product in a Hilbert space between the images of inputs from the same input space. If the two spaces in an S-function are the same, the S-function becomes a kernel function.

Koide and Yamashita (2006) defined a similarity function called asymmetric kernel and applied it to Fisher's linear discriminant. The asymmetric kernel defined by Koide and Yamashita (2006) is similar to S-function. We use the term S-function instead of asymmetric kernel in this paper, because further investigation of the properties of S-function (or asymmetric kernel), particularly the necessary and sufficient condition, is still necessary.

The learning of a similarity function between pairs of objects has been studied. When the pair of objects are from the same space, the similarity function becomes a positive semi-definite kernel; a typical approach is kernel learning (cf., Lanckriet et al., 2002; Bach et al., 2004; Ong et al., 2005; Micchelli and Pontil, 2005; Bach, 2008; Cortes, 2009; Varma and Babu, 2009). Lanckriet et al. (2002) as well as Bach et al. (2004) have proposed methods for multiple kernel learning, in which the optimal kernel (similarity function) is selected from a class of linear combinations of kernels. Besides this, Ong et al. (2005) have proposed learning a kernel function (similarity function) by using kernel methods, in which the optimal kernel is chosen from RKHS generated by the 'hyperkernel'. Our method can be viewed as an extension of Ong et al.'s method. Recently, the learning of a similarity function between pairs of objects from two different spaces has also emerged as a hot research topic (cf., Abernethy et al., 2009; Grangier and Bengio, 2008). In this paper, we propose a kernel approach for performing the learning task.

Term mismatch is one of the major challenges for search, because most of the traditional relevance models, including VSM (Salton and McGill, 1986), BM25 (Robertson et al., 1994), and LMIR (Ponte and Croft, 1998; Zhai and Lafferty, 2004), are based on term matching and the ranking result will be inaccurate when term mismatch occurs. To solve the problem, heuristic methods of query expansion or (pseudo) relevance feedback (cf., Salton and Buckley, 1997; Xu and Croft, 1996; Salton and McGill, 1986; Baeza-Yates and Ribeiro-Neto, 1999; Mitra et al., 1998; Broder et al., 2009; Zhuang and Cucerzan, 2006) and Latent Semantic Indexing (LSI) (Deerwester et al., 1990) or Probabilistic Latent Semantic Indexing (PLSI) (Hofmann, 1999) have been proposed and certain improvements have been made. The former approach tackles the problem at the term level and the latter at the topic level. In this paper, we demonstrate that we can learn a relevance model Robust BM25 to address the term mismatch challenge at the term level. The learned Robust BM25 is also an S-function.

Click-through data, which records the URLs clicked by users after their query submissions at a search engine, has been widely used in web search (Agichtein et al., 2006; Joachims, 2002; Craswell and Szummer, 2007). For example, click-through data has been used in the training of a Ranking SVM model, in which preference pairs on documents given queries are derived from click-through data (Joachims, 2002). Click-through data has also been used for calculating query similarity, because queries which link to the same URLs in click-through data may represent the same search intent (Beeferman and Berger, 2000; Cui et al., 2003; Wen et al., 2002). In this paper, we use click-through data for training a Robust BM25 as well as calculating query similarity.

Learning to rank refers to supervised learning techniques for constructing ranking models using training data (cf., Liu, 2009). Several approaches to learning to rank have been proposed and it has become one of the important technologies in the development of modern search engines (e.g., Herbrich et al., 1999; Joachims, 2002; Crammer and Singer, 2001; Agarwal and Niyogi, 2005; Freund et al., 2003; Rudin et al., 2005; Burges et al., 2006; Cao et al., 2006; Xu and Li, 2007; Cao et al., 2007). The method for learning Robust BM25 in this paper can also be viewed as a learning to rank method. Robust BM25 runs on the top of conventional learning to rank methods. Specifically, it trains a 're-ranking' model online to deal with term mismatch, while conventional learning to rank methods train a ranking model offline for basic ranking. The method of learning Robust BM25 is similar to Ranking SVM proposed by Herbrich et al. (1999) and Joachims (2002), a popular learning to rank algorithm. However, there are some differences. For example, the Robust BM25 method uses a different kernel function.

3. Similarity Function

This section describes the definition and properties of S-function, the similarity function between pairs of objects of different types.

3.1 Definition

An S-function measures the similarity between two objects from two different spaces. It is in fact the dot product between the images in the feature space mapped from two objects in the two input spaces.

Definition 1 (S-function) *Let $\mathcal{X}$ and $\mathcal{Y}$ be two input spaces, and $\mathcal{H}$ be a feature space (Hilbert space). S-function is a function $k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, satisfying $k(x,y) = \langle \varphi_X(x), \varphi_Y(y)\rangle_{\mathcal{H}}$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, where φ_X and φ_Y are mapping functions from $\mathcal{X}$ and $\mathcal{Y}$ to $\mathcal{H}$, respectively.*

A positive semi-definite kernel is defined as a function $K(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which satisfies that there is a mapping $\phi(\cdot)$ from $\mathcal{X}$ to a Hilbert space $\mathcal{H}$ with inner product $< \cdot,\cdot >_{\mathcal{H}}$, such that $\forall x, x' \in \mathcal{X}$, $K(x,x') =< \phi(x), \phi(x') >_{\mathcal{H}}$. A positive semi-definite kernel measures the similarity of pairs of objects in a single space by using the dot product of their images in a Hilbert space. In contrast, S-function measures the similarity between pairs of objects in two different spaces. If the two input spaces (also the two mapping functions) are identical in Definition 1, then S-function becomes a positive semi-definite kernel. Moreover, S-function also has some properties similar to those of positive semi-definite kernels, as shown below.

3.2 Properties

S-function has properties as shown below; they are similar to those in conventional positive semi-definite kernels, but there are also differences. Note that for a conventional kernel, α must be non-negative in property (1) of Lemma 2. The properties will enable us to construct more complicated S-functions from simple S-functions.

Lemma 2 (Properties of S-function) *Let $k_1(x,y)$ and $k_2(x,y)$ be S-functions on $\mathcal{X} \times \mathcal{Y}$, then the following functions $k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ are also S-functions: (1) $\alpha \cdot k_1$ (for all $\alpha \in \mathbb{R}$), (2) $k_1 + k_2$, (3) $k_1 \cdot k_2$.*

Proof Since $k_1(x,y)$ and $k_2(x,y)$ are S-functions, suppose that $k_1(x,y) = \langle \varphi^1_X(x), \varphi^1_Y(y)\rangle_1$ and $k_2(x,y) = \langle \varphi^2_X(x), \varphi^2_Y(y)\rangle_2$, where $\langle \cdot,\cdot\rangle_1$ is the dot product in N_1-dimensional Hilbert space and $\langle \cdot,\cdot\rangle_2$ is the dot product in N_2-dimensional Hilbert space. N_1 and N_2 can be finite or infinite.

Let $\varphi^1_{Xi}(\cdot)$ and $\varphi^1_{Yi}(\cdot)$ be the i^{th} elements of vectors $\varphi^1_X(\cdot)$ and $\varphi^1_Y(\cdot)$, respectively ($i = 1, 2, \ldots, N_1$), and $\varphi^2_{Xi}(\cdot)$ and $\varphi^2_{Yi}(\cdot)$ be the i^{th} elements of vectors $\varphi^2_X(\cdot)$ and $\varphi^2_Y(\cdot)$, respectively ($i = 1, 2, \ldots, N_2$).

(1) Let $\varphi^{1\prime}_X(x) = \alpha \cdot \varphi^1_X(x)$, we obtain $\alpha \cdot k_1(x,y) = \langle \varphi^{1\prime}_X(x), \varphi^1_Y(y)\rangle_1$, which proves that $\alpha \cdot k_1$ is an S-function, $\forall \alpha \in \mathbb{R}$.

(2) Let $\varphi_X(x) = (\varphi^1_X(x), \varphi^2_X(x))$, and $\varphi_Y(y) = (\varphi^1_Y(y), \varphi^2_Y(y))$, we obtain $\langle \varphi_X(x), \varphi_Y(y)\rangle = \langle \varphi^1_X(x), \varphi^1_Y(y)\rangle_1 + \langle \varphi^2_X(x), \varphi^2_Y(y)\rangle_2 = k_1(x,y) + k_2(x,y)$, which proves that $k_1 + k_2$ is an S-function.

(3) Let $\varphi_X(x) = \varphi^1_X(x) \otimes \varphi^2_X(x)$ and $\varphi_Y(y) = \varphi^1_Y(y) \otimes \varphi^2_Y(y)$. $\varphi_X(x)$ is a vector whose elements are $\{\varphi^1_{Xi}(x)\varphi^2_{Xj}(x)\}$, $1 \leqslant i \leqslant N_1$, $1 \leqslant j \leqslant N_2$ and $\varphi_Y(y)$ is a vector whose elements are $\{\varphi^1_{Yi}(y)\varphi^2_{Yj}(y)\}$, $1 \leqslant i \leqslant N_1$, $1 \leqslant j \leqslant N_2$. We obtain

$$\begin{aligned}
\langle \varphi_X(x), \varphi_Y(y)\rangle &= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \varphi^1_{Xi}(x)\varphi^2_{Xj}(x)\varphi^1_{Yi}(y)\varphi^2_{Yj}(y) \\
&= \sum_{i=1}^{N_1} \varphi^1_{Xi}(x)\varphi^1_{Yi}(y) \sum_{j=1}^{N_2} \varphi^2_{Xj}(x)\varphi^2_{Yj}(y) \\
&= \sum_{i=1}^{N_1} \varphi^1_{Xi}(x)\varphi^1_{Yi}(y) k_2(x,y) \\
&= k_1(x,y)k_2(x,y),
\end{aligned}$$

which proves that $k_1 \cdot k_2$ is an S-function.

■

3.3 Relevance Models as Similarity Functions

Traditional relevance models, including VSM (Salton and McGill, 1986), BM25 (Robertson et al., 1994) and LMIR (Ponte and Croft, 1998; Zhai and Lafferty, 2004), can be viewed as S-functions.[1] In fact, all these models measure the similarity of a query and document from query space and document space. In VSM, query space and document space are treated as the same space, while in the other two models, query space and document space are two different spaces.

3.3.1 VSM

Let $\mathcal{Q}$ and $\mathcal{D}$ denote query and document spaces. Each dimension in the two spaces corresponds to a term, and query and document are respectively represented as vectors in the two spaces. Let $\mathcal{H}$ denote a Hilbert space endowed with dot product $\langle \cdot, \cdot \rangle$ (it is in fact an n-dimensional Euclidean space where n is the number of unique terms).

Given query $q \in \mathcal{Q}$ and document $d \in \mathcal{D}$, VSM is calculated as

$$\text{VSM}(q,d) = \langle \varphi_Q^{\text{VSM}}(q), \varphi_D^{\text{VSM}}(d) \rangle,$$

where $\varphi_Q^{\text{VSM}}(q)$ and $\varphi_D^{\text{VSM}}(d)$ are mappings to $\mathcal{H}$ from $\mathcal{Q}$ and $\mathcal{D}$, respectively.

$$\varphi_Q^{\text{VSM}}(q)_t = idf(t) \cdot tf(t,q)$$

and

$$\varphi_D^{\text{VSM}}(d)_t = idf(t) \cdot tf(t,d),$$

where t is a term, $tf(t,q)$ is the frequency of term t in query q, $tf(t,d)$ is the frequency of term t in document d, $idf(t)$ is the inverse document frequency of term t. That is to say, VSM is a linear positive semi-definite kernel, and is an S-function as well.

3.3.2 BM25

Given query $q \in \mathcal{Q}$ and document $d \in \mathcal{D}$, BM25 is calculated as

$$\text{BM25}(q,d) = \langle \varphi_Q^{\text{BM25}}(q), \varphi_D^{\text{BM25}}(d) \rangle, \tag{1}$$

where $\varphi_Q^{\text{BM25}}(q)$ and $\varphi_D^{\text{BM25}}(d)$ are mappings to $\mathcal{H}$ from $\mathcal{Q}$ and $\mathcal{D}$, respectively.

$$\varphi_Q^{\text{BM25}}(q)_t = \frac{(k_3+1) \times tf(t,q)}{k_3 + tf(t,q)}$$

and

$$\varphi_D^{\text{BM25}}(d)_t = idf(t) \frac{(k_1+1) \times tf(t,d)}{k_1 \left(1 - b + b \cdot \frac{\text{len}(d)}{\text{avgDocLen}}\right) + tf(t,d)},$$

where $k_1 \geq 0$, $k_3 \geq 0$, and $b \geq 0$ are parameters. Moreover, $\text{len}(d)$ is the length of document d and avgDocLen is the average length of documents in the collection.

1. " The matching function between a query and document should be defined as an asymmetric function." - Stephen Robertson, personal communication.

3.3.3 LMIR

We use Dirichlet smoothing as an example. Other smoothing methods such as Jelinek-Mercer (JM) can also be used. Given query $q \in \mathcal{Q}$ and document $d \in \mathcal{D}$, the LMIR with Dirichlet smoothing is calculated as

$$\text{LMIR}(q,d) = \langle \varphi_Q^{\text{LMIR}}(q), \varphi_D^{\text{LMIR}}(d) \rangle,$$

where $\phi_Q^{\text{LMIR}}(q)$ and $\phi_D^{\text{LMIR}}(d)$ are $(n+1)$-dimensional mappings to $\mathcal{H}$ from $\mathcal{Q}$ and $\mathcal{D}$, respectively. For $t = 1,2,\dots,n$, $\phi_Q^{\text{LMIR}}(q)_t$ and $\phi_D^{\text{LMIR}}(d)_t$ are defined as

$$\varphi_Q^{\text{LMIR}}(q)_t = tf(t,q)$$

and

$$\varphi_D^{\text{LMIR}}(d)_t = \log\left(1 + \frac{tf(t,d)}{\mu P(t)}\right),$$

where $\mu > 0$ is a smoothing parameter, $P(t)$ is the probability of term t in the whole collection. $P(t)$ plays a similar role as inverse document frequency $idf(t)$ in VSM and BM25. The $(n+1)^{th}$ entries of $\varphi_Q^{\text{LMIR}}(q)$ and $\varphi_D^{\text{LMIR}}(d)$ are defined as

$$\varphi_Q^{\text{LMIR}}(q)_{n+1} = \text{len}(q)$$

and

$$\varphi_D^{\text{LMIR}}(d)_{n+1} = \log \frac{\mu}{\text{len}(d) + \mu},$$

where $\text{len}(q)$ and $\text{len}(d)$ are the lengths of query q and document d, respectively.

There are several advantages of applying the similarity function view to search. First, it gives a general and unified framework to relevance models. Although BM25 and LMIR are derived from different probability models, they work equally well in practice. It was difficult to understand the phenomenon. The S-function interpretation of the relevance models can give a better explanation of it. BM25 and LMIR are nothing but similarity functions representing query and document matching with different formulations. Second, it is easy to make an extension of the conventional relevance models based on the S-function definition. In (Xu et al., 2010), we show that the conventional relevance models can be naturally extended from unigram based models to n-gram based models to improve search relevance, with the S-function interpretation. In this paper, we demonstrate that we can deal with the term mismatch problem in a principled way on the basis of S-function.

An S-function measures the similarity of pairs of objects from two different spaces. It is an essential model not only for search, but also for many other applications such as collaborative filtering (Abernethy et al., 2009) and image retrieval (Grangier and Bengio, 2008). In the tasks, there exist two spaces and given an object in one space the goal is to find the most similar (relevant) objects in the other space. The spaces are defined over query and document, user and item, and image and text, respectively. In all these problems, the model can be represented as an S-function.

4. Learning a Robust Relevance Model

In this section, we first describe term mismatch, then propose using Robust BM25 to deal with term mismatch, and finally propose employing a kernel method to learn Robust BM25.

yutube	yuotube	yuo tube
ytube	youtubr	yu tube
youtubo	youtuber	youtubecom
youtube om	youtube music videos	youtube videos
youtube	youtube com	youtube co
youtub com	you tube music videos	yout tube
youtub	you tube com yourtube	your tube
you tube	you tub	you tube video clips
you tube videos	www you tube com	wwww youtube com
www youtube	www youtube com	www youtube co
yotube	www you tube	www utube com
ww youtube com	www utube	www u tube
utube videos	our tube	utube
u tube	my tube	toutube

Table 1: Example queries representing search intent “finding YouTube website”.

4.1 Term Mismatch in Search

Search is basically based on term match. For example, if the query is “soccer” and the term “soccer” occurs several times in the document, then the document is regarded as ‘relevant’. The relevance models of VSM, BM25 and LMIR will give high scores to the document and the document will be ranked highly. This term matching paradigm works quite well. However, the so-called term mismatch problem also inevitably occurs. That is, even if the document and the query are relevant, but they do not match at term level, in other words, they do not share a term, then they will not be viewed as relevant. For example, if the query is “New York” and the document contains “NY”, then the document will not be regarded relevant. Similarly, “aircraft” and “airplane” refer to the same concept; but if one of them is used in the query and the other in the document, then the document will be considered irrelevant. Term mismatch due to the differences in expressions including typos, acronyms, and synonyms can easily happen and deteriorate the performance of search.

In web search, users are more diverse and so are the web contents. The term mismatch problem becomes more severe than traditional search. Although modern search engines exploit more sophisticated models for retrieval and ranking, they still heavily rely on the term matching paradigm. Therefore, term mismatch is still one of the most critical challenges for web search. For example, we have observed over 200 different forms for representing the same search intent “finding YouTube website” from the query log of a commercial web search engine. Table 1 lists some examples.

The relevance models of VSM, BM25, LMIR are all based on term frequencies of $tf(t,d)$ and $tf(t,q)$. If the query and document share a term t, then the term frequencies will be non-zero values and the relevance score between the query and document will become high. That is, the value of the S-function between the query and document will be large. When term mismatch occurs, either $tf(t,d)$ or $tf(t,q)$ will be zero, and the relevance score will be low, although it should not be so. In that case, the value of S-function will be unnecessarily small.

More generally, term mismatch corresponds to the fact that some S-function values are reliable while the others are not. The question is whether it is possible to ‘smooth’ the S-function values based on some training data and to do it in a theoretically sound way. The kernel approach that we propose in this paper can exactly solve the problem.

4.2 Robust BM25 Model

We try to learn a more reliable relevance model (S-function) from data. The model is an extension of BM25 but more robust to term mismatch. We call the model 'Robust BM25'. Without loss of generality, we use BM25 as the basic relevance model; one can easily extend the techniques here to other relevance models.

We give the definition of Robust BM25 and then explain why it has the capability to cope with term mismatch.

Robust BM25 (RBM25) is defined as follows

$$k_{RBM25}(q,d) = \sum_{i=1}^{N} \alpha_i \cdot k_{BM25}(q,d) k_Q(q,q_i) k_D(d,d_i) k_{BM25}(q_i,d_i), \tag{2}$$

where $k_{BM25}(q,d)$ is the BM25 model, $k_Q : \mathcal{Q} \times \mathcal{Q} \to \mathbb{R}$ and $k_D : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ are positive semi-definite kernels in query space and document space, which represent query similarity and document similarity, respectively. N is the number of training instances. $\{\alpha_i\}_{i=1}^N$ are weights and can be learned from training data. In fact, Robust BM25 is also an S-function that measures the similarity of query q and document d through a dot product in a Hilbert space, as will be explained in Section 5.

Here we assume that $k_{BM25}(q,d) > 0, \forall q \in \mathcal{Q}, d \in \mathcal{D}$, otherwise, we can add a small positive value ε to Equation (1). Furthermore, we assume that $0 \leq k_Q(\cdot,\cdot) \leq 1$ and $0 \leq k_D(\cdot,\cdot) \leq 1$.

Robust BM25 is actually a linear combination of BM25 scores of similar queries and similar documents. Because it is based on smoothing, it can be more robust, particularly when the weights are learned from data.

Figure 1 gives an intuitive explanation on why Robust BM25 can effectively deal with term mismatch. Suppose that the query space contains queries as elements and has the kernel function k_Q as a similarity function. Given query q, one can find its similar queries q_i based on $k_Q(q,q_i)$ (its neighbors). Similarly, the document space contains documents as elements and has the kernel function k_D as a similarity function. Given document d, one can find its similar documents d_i based on $k_D(d,d_i)$ (its neighbors). The relevance model BM25 is defined as an S-function between query and document over the two spaces. Term mismatch means that the BM25 score $k_{BM25}(q,d)$ is not reliable.

One possible way to deal with the problem is to use the neighboring queries q_i and documents d_i to smooth the BM25 score of q and d, as in the k-nearest neighbor algorithm (Cover and Hart, 1967; Dudani, 1976). In other words, we employ the k-nearest neighbor method in both the query and document spaces to calculate the final relevance score (cf., Figure 1). This is exactly what Robust BM25 does. More specifically, Robust BM25 determines the ranking score of query q and document d, not only based on the relevance score between q and d themselves (i.e., $k_{BM25}(q,d)$), but also based on the relevance scores between similar queries q_i and similar documents d_i (i.e., $k_{BM25}(q_i,d_i)$), and it makes a weighted linear combination of the relevance scores (2).

To help further understand why Robust BM25 can tackle the term mismatch problem, we give an example. If q is "NY", and d is about "New York", then $k_{BM25}(q,d)$ will fail to match them because q and d do not share any term. On the other hand, if q' is "New York", and we know that q and q' are similar ($k_Q(q,q')$ is high), and $k_{BM25}(q',d)$ should have a high matching score, then

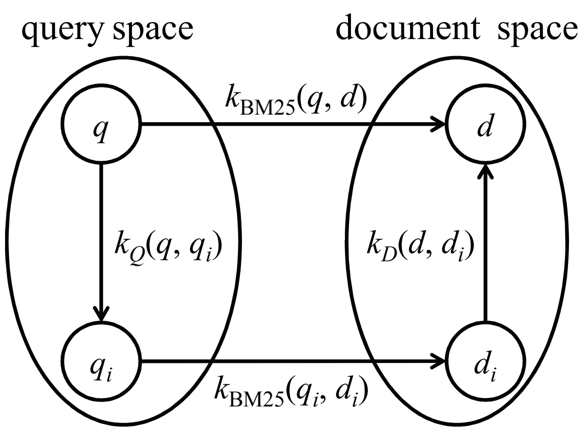


Figure 1: Robust BM25 deals with term mismatch by using the neighbors in query space and document space.

we can use $k_{BM25}(q',d)$ to boost $k_{BM25}(q,d)$. Note here that we assume $d = d'$ and $k_D(d,d') = 1$. Therefore, Robust BM25 can overcome the term mismatch problem and outperform conventional IR models.

4.3 Learning Robust BM25

We learn the weights $\{\alpha_i\}_{i=1}^N$ in Robust BM25 by using training data and a kernel method. In the kernel method, we use the following kernel on $\mathcal{Q} \times \mathcal{D}$:

$$k_{HBM25}((q,d),(q',d')) = k_{BM25}(q,d)k_Q(q,q')k_D(d,d')k_{BM25}(q',d'), \tag{3}$$

where $k_{BM25}(q,d)$ is the BM25 model, k_Q and k_D are the query and document similarity kernels.

Suppose that the reproducing kernel Hilbert space (RKHS) generated by k_{HBM25} is $\mathcal{H}_{k_{HBM25}}$. Given some training data $\{(q_i,d_i,r_i)\}_{i=1}^N$ where r_i represents the relevance degree between query q_i and document d_i, the learning problem is then as follows

$$\operatorname*{arg\,min}_{k \in \mathcal{H}_{k_{HBM25}}} \frac{1}{N}\sum_{i=1}^{N} l(k(q_i,d_i),r_i) + \frac{\lambda}{2}\|k\|^2_{\mathcal{H}_{k_{HBM25}}}, \tag{4}$$

where $l(\cdot,\cdot)$ is a loss function and $\|\cdot\|_{\mathcal{H}_{k_{HBM25}}}$ is the norm defined in $\mathcal{H}_{k_{HBM25}}$.

According to the representer theorem of kernel methods (Hofmann et al., 2008; Schölkopf and Smola, 2002), the optimal relevance model $k^*(q,d)$ has exactly the same form as Robust BM25 in Equation (2).

Robust BM25 (2) is also an S-function, because k_{HBM25} (3) belongs to a specific kernel class referred to as S-kernel in this paper.

5. S-kernel

In this section, we give the definition of S-kernel and also explain the kernel method of learning an S-function using an S-kernel.

Suppose that we are given training data $\mathcal{S} = \{(x_i, y_i), t_i\}_{i=1}^N$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ are a pair of objects, and $t_i \in \mathcal{T}$ is their response. The training data can be that for classification, regression, or ranking. Suppose that the hypothesis space $\mathcal{K}$ is a space of S-functions. Our goal is to learn the optimal S-function from the hypothesis space given the training data. We consider employing a kernel method to perform the learning task. That is, we specifically assume that the hypothesis space is also an RKHS generated by a positive semi-definite kernel.

The learning problem then becomes the following optimization problem:

$$\arg\min_{k \in \mathcal{K}} \frac{1}{N} \sum_{i=1}^N l(k(x_i, y_i), t_i) + \frac{\lambda}{2} \|k\|_{\mathcal{K}}^2, \tag{5}$$

where $\lambda > 0$ is a coefficient, $\mathcal{K}$ is a subspace of S-functions endowed with norm $||\cdot||_{\mathcal{K}}$, and $||k||_{\mathcal{K}}$ denotes regularization on space $\mathcal{K}$. Here $\mathcal{K}$ is also an RKHS generated by a positive semi-definite kernel $k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$, that is, for each S-function $k(x, y) \in \mathcal{K}$, $k(x, y) = \langle k(\cdot, \cdot), k((\cdot, \cdot), (x, y)) \rangle_{\mathcal{K}}$.

According to the representer theorem of kernel methods, the optimal solution of problem (5) is in the form

$$k^*(x, y) = \sum_{i=1}^N \alpha_i k((x_i, y_i), (x, y)),$$

where $\alpha_i \in \mathbb{R}, 1 \le i \le N$, and N denotes the number of training instances.

The question then is whether there exists space $\mathcal{K}$, or equivalently kernel k. We show below that it is the case and refer to the kernel k as S-kernel.

We formally define S-kernel and give two families of S-kernels.

Definition 3 (S-kernel) *Let $\mathcal{X}$ and $\mathcal{Y}$ be two input spaces. $k((x, y), (x', y'))$ is called S-kernel, if it has the following properties. (1) $k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$ is a positive semi-definite kernel. (2) All the elements in the RKHS generated by k are S-functions on $\mathcal{X}$ and $\mathcal{Y}$.*

If the two input spaces $\mathcal{X}$ and $\mathcal{Y}$ are identical in Definition 3, then S-kernel degenerates to the hyperkernel proposed by Ong et al. (2005).

We give two families of S-kernels based on power series and multiple kernels.

Theorem 4 (Power Series Construction) *Given two Mercer kernels $k_X : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $k_Y : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, for any S-function $g(x, y)$ and $\{c_i\}_{i=0}^\infty \subset \mathbb{R}^+$, k_P defined below is an S-kernel.*

$$k_P((x, y), (x', y')) = \sum_{i=0}^\infty c_i \cdot g(x, y) \left(k_X(x, x') k_Y(y, y')\right)^i g(x', y'), \tag{6}$$

where the convergence radius of $\sum_{i=0}^\infty c_i \xi^i$ is R, $|k_X(x, x')| < \sqrt{R}$, $|k_Y(y, y')| < \sqrt{R}$, for any x, x', y, y'.

Theorem 5 (Multiple Kernel Construction) *Given two finite sets of Mercer kernels $K_X = \left\{k_i^X(x, x')\right\}_{i=1}^n$ and $K_Y = \left\{k_i^Y(y, y')\right\}_{i=1}^n$. For any S-function $g(x, y)$ and $\{c_i\}_{i=1}^n \subset \mathbb{R}^+$, k_M defined below is an S-kernel.*

$$k_M((x, y), (x', y')) = \sum_{i=1}^n c_i \cdot g(x, y) k_i^X(x, x') k_i^Y(y, y') g(x', y'). \tag{7}$$

rank	URL	click
1	`www.walmart.com`	Yes
2	`en.wikipedia.org/wiki/Wal*Mart`	No
3	`www.walmartstores.com`	Yes
4	`instoresnow.walmart.com`	No
5	`mp3.walmart.com`	No

Table 2: A record of click-through data for query "walmart". Only the top 5 URLs are shown.

Proofs of Theorem 4 and Theorem 5 are given in Appendix A and Appendix B, respectively.

Note that if space $\mathcal{X}$ and space $\mathcal{Y}$ are identical, k_X and k_Y are identical, k_i^X and k_i^Y are identical for any $1 \leqslant i \leqslant n$, and $g(x,y) = 1$, then k_P and k_M are exactly the hyperkernels given in Section 4.1 and Section 4.3 respectively by Ong et al. (2005).

With the theorems one can easily verify that the following kernel is an S-kernel.

$$g(x,y)k_X(x,x')k_Y(y,y')g(x',y'), \tag{8}$$

where $g(x,y)$ is an S-function, and $k_X(x,x')$ and $k_Y(y,y')$ are positive semi-definite kernels on spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively. In fact, the S-kernel in Equation (8) is a member of the families of S-kernels in both Equation (6) and Equation (7).

It is obvious that in the learning of Robust BM25 (4), we specify $g(x,y)$ as BM25 (an S-function), and k_X and k_Y as query similarity kernel k_Q and document similarity kernel k_D, respectively. Therefore, the learning problem (4) is a specific case of learning with S-kernel (5), and Robust BM25 (2) is an S-function.

Basilico and Hofmann (2004) propose a pairwise kernel for collaborative filtering. The pairwise kernel is defined as $k_C((u,i),(u',i')) = k_U(u,u') \cdot k_I(i,i')$, where k_U and k_I are kernels defined on the spaces of users and items, respectively. Obviously, k_C is an S-kernel and their learning problem is another specific case of learning with S-kernel (5).

6. Implementation

In this section, we describe a specific implementation to learn Robust BM25 (4).

To learn Robust BM25, we need to decide the query similarity $k_Q(q,q')$, document similarity $k_D(d,d')$, training data, and optimization technique. We explain one way of implementing them.

Click-through data has been proven to be useful for improving search relevance (cf., Cui et al., 2003; Joachims, 2002). An instance of click-through data consists of a query, a ranked list of URLs, and a user's clicks. Table 2 shows a click-through instance. In this case, the user submitted the query "walmart" and received the ranked list of URLs, and the user clicked on the URLs at ranks 1 and 3 but skipped the URLs at ranks 2, 4, and 5. Every time when a search is conducted using a search engine, this kind of data is recorded. The amount of click-through data is usually extremely large. Obviously users do not click on URLs at random, but based on their relevance judgments. Though click-through data is noisy, it still conveys users' implicit feedback to search results.

To calculate query similarity, we represent query and URL click-through relationships in a bipartite graph in which queries and URLs are nodes in two sets and clicks are edges between nodes in the two sets. A weight is associated with each edge representing the total number of times that the URL is clicked after the query is issued. Figure 2 illustrates a click-through bipartite graph.

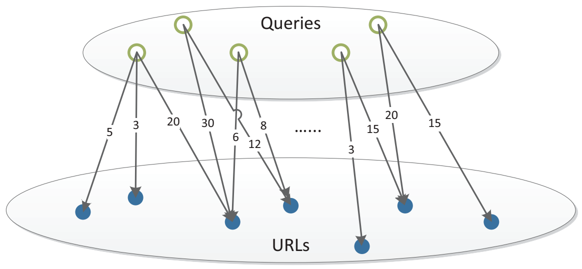


Figure 2: Click-through bipartite graph.

We specifically define query similarity using co-clicked URLs in the click-through bipartite graph. Intuitively, if two queries share many clicked URLs, then they will be regarded as similar. Since queries with the same search intent tend to be linked to the same URLs, query similarity defined in this way actually represents the degree of being the same search intent. We calculate the query similarity function $k_Q(q,q')$ as a Pearson Correlation Coefficient between the co-clicked URLs of two queries:

$$k_Q(q,q') = \frac{\sum_{i=1}^{n}(u_i - u)(v_i - v)}{\sqrt{\sum_{i=1}^{n}(u_i - u)^2}\sqrt{\sum_{i=1}^{n}(v_i - v)^2}}, \tag{9}$$

where u_i and v_i denote the numbers of clicks on URL i by queries q and q' respectively, u and v denote the average numbers of clicks of q and q' respectively, and n denotes the total number of clicked URLs by q and q'. Note that query similarity $k_Q(q,q')$ defined in Equation (9) is a positive semi-definite kernel, because it is the dot product of two vectors $(\frac{u_1-u}{\sqrt{\sum_{i=1}^{n}(u_i-u)^2}},\ldots,\frac{u_n-u}{\sqrt{\sum_{i=1}^{n}(u_i-u)^2}})$ and $(\frac{v_1-v}{\sqrt{\sum_{i=1}^{n}(v_i-v)^2}},\ldots,\frac{v_n-v}{\sqrt{\sum_{i=1}^{n}(v_i-v)^2}})$ in $\mathbb{R}^n$.

Our experimental results also show that by using the similarity function, one can really find similar queries with high quality.[2] Table 3 shows some examples of similar queries found by using our method. In fact, with the use of click-through bipartite and query similarity measure, different types of similar queries can be found, including spelling error (e.g., "wallmart" v.s. "walmart"), word segmentation ("ironman" v.s. "iron man"), stemming (e.g., "knives" v.s. "knifes" and "knife"), synonym (e.g., "aircraft for sale" v.s. "airplanes for sale"), and acronym (e.g., "ucsd" v.s. "university of california san diego").

Document similarity $k_D(d,d')$ is simply defined as the cosine similarity between the titles and URLs of two documents, which is certainly a kernel (cosine similarity is the dot product in an Euclidean space).

Following the proposal given by Joachims (2002), we generate pairwise training data from click-through data. More precisely, for each query q_i we derive preference pairs (d_i^+, d_i^-), where d_i^+ and d_i^- mean that document d_i^+ is more preferred than d_i^- with respect to query q_i (e.g., d_i^- is skipped even though it is ranked higher than d_i^+).

Finally, we take the pairwise training data as input and learn the optimal S-function, Robust BM25. We use hinge loss as the loss function, the learning problem (4) then becomes

$$\arg\min_{k \in \mathcal{H}_{k_{HBM25}}} \sum_{i=1}^{M} \left[1 - (k(q_i, d_i^+) - k(q_i, d_i^-))\right]_+ + \frac{\lambda}{2}\|k\|^2_{\mathcal{H}_{k_{HBM25}}}, \tag{10}$$

2. We evaluated the precision of several similar measures. the Pearson Correlation Coefficient and the Jensen-Shannon Divergence work the best, followed by the Jaccard Coefficient.

original query	**similar queries**
wallmart	wall mart, walmart, wal mart, walmarts
ironman	iron man, ironman movie, irnman, www.iron man.com
knives	knifes, knives.com, knife outlet, knife
aircraft for sale	aircraft sales, airplanes for sale, used airplanes for sale, used planes for sale
ucsd	ucsd.edu, uc san diego, uscd, university of california san diego

Table 3: Similar queries extracted from web search click-through data.

where M is the number of preference pairs in the training data. Note that this is similar to Ranking SVM (Herbrich et al., 1999). The major difference is that in our case the kernel function used is an S-kernel.

According to the representer theorem and Equation (2), when using pairwise training data, the optimal solution is given as follows

$$k_{RBM25}(q,d) = k_{BM25}(q,d) \cdot \sum_{i=1}^{M} \theta_i \cdot k_Q(q,q_i) \left[k_{BM25}(q_i,d_i^+) k_D(d_i^+,d) - k_{BM25}(q_i,d_i^-) k_D(d_i^-,d) \right], \tag{11}$$

where θ_i is a parameter to learn.

Reformulating the non-constrained optimization in Equation (10) as a constrained optimization by using Equation (11) and slack variables $\{\xi_i\}$, we obtain the following primal problem:

$$\arg\min_{\{\theta_i\}_{i=1}^M} \sum_{i=1}^{M} \xi_i + \frac{\lambda}{2} \sum_{i,j=1}^{M} \theta_i \theta_j \mathcal{W}(i,j) \tag{12}$$
$$k_{RBM25}(q_i,d_i^+) - k_{RBM25}(q_i,d_i^-) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0 \; \forall i,$$

where calculating $\mathcal{W}(i,j)$ using the reproducing kernel property is given by

$$\begin{aligned}\mathcal{W}(i,j) = k_Q(q_i,q_j) \cdot [&k_D(d_i^+,d_j^+) k_{BM25}(q_i,d_i^+) k_{BM25}(q_j,d_j^+) - k_D(d_i^+,d_j^-) k_{BM25}(q_i,d_i^+) k_{BM25}(q_j,d_j^-) \\ &- k_D(d_i^-,d_j^+) k_{BM25}(q_i,d_i^-) k_{BM25}(q_j,d_j^+) + k_D(d_i^-,d_j^-) k_{BM25}(q_i,d_i^-) k_{BM25}(q_j,d_j^-)].\end{aligned}$$

With Lagrange multipliers $\{\beta_i\}_{i=1}^M$ and $\{\gamma_i\}_{i=1}^M$, the objective function becomes

$$\begin{aligned} L &= \sum_{i=1}^{M} \xi_i + \frac{\lambda}{2} \sum_{i,j=1}^{M} \theta_i \theta_j \mathcal{W}(i,j) + \sum_{i=1}^{M} \beta_i \left[1 - \xi_i - \left(k_{RBM25}(q_i,d_i^+) - k_{RBM25}(q_i,d_i^-) \right) \right] - \sum_{i=1}^{M} \gamma_i \xi_i \\ &= \sum_{i=1}^{M} \xi_i + \frac{\lambda}{2} \sum_{i,j=1}^{M} \theta_i \theta_j \mathcal{W}(i,j) + \sum_{i=1}^{M} \beta_i \left[1 - \xi_i - \sum_{j=1}^{M} \theta_j \mathcal{W}(i,j) \right] - \sum_{i=1}^{M} \gamma_i \xi_i, \end{aligned}$$

Differentiating L by ξ_i and θ_i, we have

$$\frac{\partial L}{\partial \xi_i} = 1 - \beta_i - \gamma_i = 0, \tag{13}$$

and

$$\frac{\partial L}{\partial \theta_i} = \sum_{j=1}^{M} (\lambda\theta_j - \beta_j)\mathcal{W}(i,j) = 0. \tag{14}$$

Thus, according to Equation (13), we have

$$\gamma_i = 1 - \beta_i.$$

Since $\beta_i \geqslant 0$ and $\gamma_i \geqslant 0$, we have $0 \leqslant \beta_i \leqslant 1$. According to Equation (14), we have

$$\sum_{j=1}^{M} \lambda\theta_j \mathcal{W}(i,j) = \sum_{j=1}^{M} \beta_j \mathcal{W}(i,j).$$

Substituting the above two formulas into L, we obtain the dual problem:

$$\arg\max_{\{\beta\}_{i=1}^{M}} \sum_{i=1}^{M} \beta_i - \frac{1}{2\lambda}\sum_{i=1}^{M}\sum_{j=1}^{M} \beta_i\beta_j \mathcal{W}(i,j) \quad s.t. \quad 0 \leq \beta_i \leq 1. \tag{15}$$

By solving the dual problem (15) we obtain the optimal values $\{\beta_i^\star\}_{i=1}^{M}$. We can further get the optimal values $\{\theta_i^\star\}_{i=1}^{M}$ by solving equation (14), and using $\theta_i^\star = \frac{1}{\lambda}\beta_i^\star$. Note that when $(\mathcal{W}(i,j))_{M\times M}$ is not strictly positive, the solution of (14) is not unique. In such a case, we can still take $\theta_i^\star = \frac{1}{\lambda}\beta_i^\star$ as a solution for simplicity, because all solutions will make the objective function achieve the same minimum (12).

In online search, given a query, we first retrieve the queries similar to it, then individually retrieve documents with the original query and similar queries, combine the retrieved documents, train a Robust BM25 model using click-through data, and rank the documents with their Robust BM25 scores (note that a Robust BM25 model is trained for each query). When training Robust BM25, we solve the dual problem (15) using a standard QP solver LOQO.[3] The time complexity is of order $O(M^2)$, where M is the number of preference pairs. Since the number of retrieved documents is small, a search with Robust BM25 can be carried out efficiently. In our experiments, we observe that on average it takes about 1.5 seconds per query to train a model on a workstation with Quad-Core Intel Xeon E5410 2.33GHz CPU and 16GB RAM.

7. Experiments

We conducted experiments to test the performances of Robust BM25.

7.1 Experimental Data

In our experiments, we used two large scale data sets from a commercial web search engine and an enterprise search engine running in an IT company. The two data sets consist of query-URL pairs and their relevance judgments. The relevance judgments can be ‘Perfect’, ‘Excellent’, ‘Good’, ‘Fair’, or ‘Bad’. Besides this, we also collected large scale click-through data from both search engines. Table 4 shows the statistics on the two data sets. The click-through data in both data sets was split into two parts, one for learning query similarity and the other for learning Robust BM25.

3. LOQO can be found at `http://www.princeton.edu/~rvdb/loqo/LOQO.html`.

	Web search	Enterprise search
# of judged queries	8,294	2,864
# of judged query-URL pairs	1,715,844	282,130
# of search impressions in click-through	490,085,192	17,383,935
# of unique queries in click-through	14,977,647	2,368,640
# of unique URLs in click-through	30,166,304	2,419,866
# of clicks in click-through	2,605,404,156	4,996,027

Table 4: Statistics on web search and enterprise search data sets.

7.2 Baselines

BM25 was selected as a baseline, whose parameters were tuned by using the validation set. Query expansion (Xu and Croft, 1996) was also chosen as a baseline. Query expansion is a state-of-the-art technique to tackle term mismatch in search. The key idea in query expansion is to add into the original query terms extracted from relevant queries or documents. Thus, even though the original query and document do not share a term, after expansion, the query is enriched and it is likely to be matched with relevant documents. On the other hand, query expansion may also suffer from the so-called topic drift problem. That is, irrelevant terms can be added to the original query. As a result, the accuracy of search may drop, rather than improve. In contrast, our method can effectively address the problem. First, similar queries mined from click-through data are used in search, which represent the same or similar intent. Thus, the documents retrieved are more likely to be relevant. Second, the final ranking of results is based on Robust BM25 which is trained specifically for the query using click-through data. Therefore, the accuracy of the final ranking will be high.

In our experiment, we tried several different ways to conduct query expansion and chose the one performing the best as the baseline. In our method, we first use the title of the most clicked URL in the retrieved result to do expansion. If there is no such a URL, we use the terms of the most similar query to do expansion.

The pairwise kernel, which is initially proposed for collaborative filtering (Basilico and Hofmann, 2004), was also chosen as a baseline. The difference between our method and the pairwise kernel is that the pairwise kernel does not use a traditional relevance model $k_{BM25}(q,d)$.

7.3 Evaluation Measures

As evaluation measures, we used Mean Average Precision (MAP) (Baeza-Yates and Ribeiro-Neto, 1999) and Normalized Discounted Cumulative Gain (NDCG) (Jarvelin and Kekalainen, 2000) at positions 1, 3, and 5, which are standard measures in IR.

MAP assesses the accuracy of a ranking algorithm by looking at how well it ranks relevant documents against irrelevant documents. MAP denotes mean average precision (AP). Average precision is defined as

$$AP(q) = \frac{\sum_{r=1}^{N_q} P(r) \times rel(r)}{\sum_{r=1}^{N_q} rel(r)},$$

where N_q is the number of documents retrieved, $rel(r) \in \{0,1\}$, and if the document ranked at position r is relevant, $rel(r) = 1$, otherwise, $rel(r) = 0$. $P(r)$ is precision at position r:

$$P(r) = \frac{\sum_{i=1}^{r} rel(i)}{r}.$$

Finally, MAP is defined as

$$\text{MAP} = \frac{\sum_q AP(q)}{\#q},$$

where $\#q$ is the number of queries. If relevant documents are ranked higher than irrelevant documents, the value of MAP will be high.

NDCG is usually used to assess a ranking algorithm when documents have multiple relevance grades (e.g., “Bad”, “Good”, “Fair”, “Excellent”, and “Perfect”). Given a query q, NDCG at position n is defined as

$$\text{NDCG@}n(q) = \frac{\text{DCG@}n(q)}{\text{IDCG@}n(q)},$$

where DCG@$n(q)$ is defined as

$$\text{DCG@}n(q) = \sum_{i=1}^{n_q} \frac{2^{rel(i)} - 1}{\log_2(i+1)},$$

where $rel(i)$ is the relevance grade of a document ranked at position i. The DCG@$n(q)$ score is normalized by IDCG@$n(q)$, which is an ideal DCG@$n(q)$ score when documents are ranked in decreasing order of their relevance grades.

Finally, NDCG is averaged over queries.

$$\text{NDCG@}n = \frac{\sum_q \text{NDCG@}n(q)}{\#q}$$

A high NDCG score means that relevant documents are ranked higher in the ranking list than irrelevant documents.

In our experiment, when calculating MAP, we view the documents with judgments ‘Perfect’ and ‘Excellent’ as relevant and the documents with the other three judgments as irrelevant.

7.4 Experimental Results

We trained a model for each query, as described in Section 6. On average, about 207.6 and 174.7 training pairs were used for each query in web search data and enterprise data, respectively. The only parameter λ in Equation (15) was heuristically set as 1. In fact, we found that λ does not affect the results so much. Table 5 reports the results on the web search data and enterprise data. We can see that Robust BM25 outperforms the baselines, in terms of all measures on both data sets. We conducted significant tests (t-test) on the improvements. The results show that the improvements are all statistically significant (p-value < 0.05). We conducted analysis on the cases in which Robust BM25 performs better and found that the reason is that Robust BM25 can indeed effectively address the term mismatch problem. The pairwise kernel outperforms BM25 and query expansion, which indicates that it is better to learn a relevance model in search. However, its performance is still lower than Robust BM25, suggesting that it is better to include BM25 in the final relevance model, as in Robust BM25.

		MAP	NDCG@1	NDCG@3	NDCG@5
Web search	Robust BM25	0.1192	0.2480	0.2587	0.2716
	Pairwise Kernel	0.1123	0.2241	0.2418	0.2560
	Query Expansion	0.0963	0.1797	0.2061	0.2237
	BM25	0.0908	0.1728	0.2019	0.2180
Enterprise search	Robust BM25	0.3122	0.4780	0.5065	0.5295
	Pairwise Kernel	0.2766	0.4465	0.4769	0.4971
	Query Expansion	0.2755	0.4076	0.4712	0.4958
	BM25	0.2745	0.4246	0.4531	0.4741

Table 5: Ranking accuracies on web search and enterprise search data.

Query	wal**l**mart
Similar queries	wall mart, walmart, wal mart, walmarts
Page	`http://www.walmart.com`
Title	Walmart.com: Save money. Live better
Rate	Perfect

Table 6: Example 1 from web search.

7.5 Discussions

We investigated the reasons that Robust BM25 can outperform the baselines, using the experiments on web search data as examples. It seems that Robust BM25 can effectively deal with term mismatch with its mechanisms: using query similarity and document similarity.

Our approach can effectively deal with term mismatch with similar queries. Table 6 gives an example. The query, web page, and label are respectively "wallmart", which is a typo, "`http://www.walmart.com`" with title "Walmart.com: Save money. Live better", and "Perfect", which means that the page should be ranked in first position. There is a mismatch between query and page, the basic relevance model BM25 cannot give a high score to the page (note that there is a difference between the query term "wallmart" and the document term "walmart".). Query expansion cannot rank the page high, either. The web page "`http://www.walmartstores.com`" with title "Walmartstores.com" is the most clicked web page with respect to the original query in the click-through data. Query expansion uses the title to conduct term expansion, that is, uses the words in the title. Because it does not have sufficient knowledge to break "Walmartstores" into "walmart" and "stores", query expansion cannot add good terms to the original query. When query expansion adds more terms to the original query, "walmart" will appear, but at the same time noise terms will also be included. In contrast, our approach can effectively leverage similar queries such as "walmart", "wal mart", and "walmarts" and rank the web page to first position.

Table 7 gives another example. The query is "mensmagazines", which is a tail query and does not have a similar query found in the click-through data. The web page is "`http://en.wikipedia.org/wiki/List/_of/_men's/_magazines`" (referred to as Page1) and the relevance label is "Excellent". There is a mismatch, because there is not sufficient knowledge to break query "mensmagazines" into "mens" and "magazines". As a result, BM25 cannot rank Page1 high. In contrast, Robust BM25 uses similar documents to calculate the relevance. Specifically, it uses a similar web page "`http://www.askmen.com/links/sections/mensmagazines.html`" (referred to

Query	mensmagazines
Page1	`http://en.wikipedia.org/wiki/List/_of/_men's/_magazines`
Title1	List of men's magazines - Wikipedia, the free encyclopedia
Rate1	Excellent
Page2	`http://www.askmen.com/links/sections/mensmagazines.html`
Title2	AskMen.com - Men's magazines

Table 7: Example 2 from web search.

Query	southwest airlines
Page1	`http://www.southwest-airlines.net`
Title1	Southwest Airlines
Rate1	Perfect
Page2	`http://www.southwestvacations.com/index.asp`
Title2	Southwest Vacations - Vacation Packages - Cheap Airline Tickets, Hotels, Rental Cars, Activities & Attractions
Rate2	Fair

Table 8: Example 3 from web search.

as Page2), which contains the term "mensmagazines" in its URL. The original query can match well with Page2. Besides, Page1 and Page2 are also similar because they have common terms "men" and "magazines" in titles. Therefore, Robust BM25 can assign a high score to Page1.

Compared with the pairwise kernel, Robust BM25 successfully leveraged the traditional matching model BM25 when its score is reliable to reflect relevance between query and document. We show an example in Table 8. The query is "southwest airlines". The two web pages are "`http://www.southwest-airlines.net`" (referred to as Page1) with label "Perfect" and "`http://www.southwestvacations.com/index.asp`" (referred to as Page2) with label "Fair". In the pairwise kernel, the ranking score of Page2 is larger than Page1. In Robust BM25, however, the ranking score of Page1 is larger. This is because the pairwise kernel does not consider the match between query and documents using BM25, while Robust BM25 does.

8. Conclusion and Future Work

We have formally defined a similarity function between pairs of objects from two different spaces and named it S-function. We have shown that traditional relevance models in search proposed in information retrieval can be viewed as S-functions. We have proposed a new kernel method for learning a robust relevance model as an S-function for search. The learned model can deal with the term mismatch problem which traditional relevance models suffer from. The kernel method employs a new kernel called S-kernel. An S-kernel is a kernel that can generate an RKHS which is also a space of S-functions. We have provided a theoretical basis for constructing S-kernels. Finally, we have shown that we can apply our method to learn a Robust BM25 model to deal with term mismatch in search.

There are several directions for future research from the current work:

1. **S-function as generalization of kernel:** In this paper, we give a formal definition of S-function and show that it is related to a positive semi-definite kernel. S-function is also similar to the asymmetric kernel defined by Koide and Yamashita (2006). To make S-function a generalization of a positive semi-definite kernel, there are still some open questions that we need to answer. For example, what is a necessary and sufficient condition for a two-argument function over two spaces to be an S-function? Is there a theorem like the Mercer theorem for S-function?

2. **S-kernel:** We define two families of S-kernels in this paper, that is, to give two sufficient conditions for a positive semi-definite kernel to be an S-kernel. It is still an open question: what is a necessary and sufficient condition for a positive semi-definite kernel to be an S-kernel?

3. **Similarity function learning:** We employ a kernel method to learn a similarity function for search. An interesting research direction is to study the general problem of similarity function learning, particularly, the learning of a similarity function for pairs of objects from two different spaces. The learning task can be applied to a wide range of applications and is becoming a popular research topic.

4. **Learning of S-Kernel:** Our kernel method employs S-kernel which contains free parameters. How to automatically learn the parameters from data, and thus a better S-function is also an interesting issue.

Acknowledgments

We would like to thank Stephen Robertson for his suggestions on this work. We also acknowledge Corinna Cortes, the anonymous reviewers, and Matt Callcut for their many valuable comments on this paper.

Appendix A. Proof of Theorem 4

Theorem 4 *Given two Mercer kernels $k_X : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $k_Y : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, for any S-function $g(x,y)$ and $\{c_i\}_{i=0}^{\infty} \subset \mathbb{R}^+$, k_P defined below is an S-kernel.*

$$k_P((x,y),(x',y')) = \sum_{i=0}^{\infty} c_i \cdot g(x,y)\left(k_X(x,x')k_Y(y,y')\right)^i g(x',y'),$$

where the convergence radius of $\sum_{i=0}^{\infty} c_i \xi^i$ is R, $|k_X(x,x')| < \sqrt{R}, |k_Y(y,y')| < \sqrt{R}$, for any x,x',y,y'.

According to Definition 3, to prove Theorem 4, we first need to prove that $k_P((x,y),(x',y'))$ is a positive semi-definite kernel. Note that $g(x,y)g(x',y')$ is a positive semi-definite kernel, since it is symmetric and $\forall\{\alpha\}_{i=1}^n, \{(x_i,y_i)\}_{i=1}^n$, $\sum_{i,j=1}^n \alpha_i\alpha_j g(x_i,y_i)g(x_j,y_j) = (\sum_{i=1}^n \alpha_i g(x_i,y_i))^2 \geq 0$. Moreover, for any $i \in \mathbb{Z}^+$, $c_i \in \mathbb{R}^+$, $c_i\left(k_X(x,x')k_Y(y,y')\right)^i$ is also a positive semi-definite kernel. Hence, $c_i g(x,y)\left(k_X(x,x')k_Y(y,y')\right)^i g(x',y')$ is a positive semi-definite kernel. Since the summation of positive semi-definite kernels is also a positive semi-definite kernel, we conclude that $k_P((x,y),(x',y'))$ is a positive semi-definite kernel.

Second, we need to prove that all elements in the reproducing kernel Hilbert space $\mathcal{K}_P$ generated by k_P are S-functions. We need two lemmas:

Lemma 6 *Suppose that $g(x,y)$ is an S-function, and k_X and k_Y are Mercer kernels. Given any finite example set $\{(x_j,y_j)\}_{j=1}^N \subset \mathcal{X} \times \mathcal{Y}$, and any $\{\alpha_j\}_{j=1}^N \subset \mathbb{R}$, $\sum_{j=1}^N \alpha_j k_P((x,y),(x_j,y_j))$ is an S-function.*

Proof

$$\begin{aligned}\sum_{j=1}^N \alpha_j k_P((x,y),(x_j,y_j)) &= \sum_{j=1}^N \alpha_j g(x,y) \sum_{i=0}^\infty c_i \left(k_X(x,x_j)k_Y(y,y_j)\right)^i g(x_j,y_j)\\ &= g(x,y)\sum_{j=1}^N \alpha_j \tilde{k}_P((x,y),(x_j,y_j)),\end{aligned}$$

where

$$\tilde{k}_P((x,y),(x_j,y_j)) = \sum_{i=0}^\infty c_i \left(k_X(x,x_j)k_Y(y,y_j)\right)^i g(x_j,y_j).$$

Since $g(x,y)$ is an S-function, according to Lemma 2, to prove $\sum_{j=1}^N \alpha_j k_P((x,y),(x_j,y_j))$ is an S-function, we only need to show that $\sum_{j=1}^N \alpha_j \tilde{k}_P((x,y),(x_j,y_j))$ is an S-function.

For any $i \geqslant 0, i \in \mathbb{Z}^+$, since $\sqrt{c_i}k_X^i(x,x')$ and $\sqrt{c_i}k_Y^i(y,y')$ are both Mercer kernels, we obtain

$$\sqrt{c_i}k_X^i(x,x') = \langle \psi_X^i(x), \psi_X^i(x')\rangle_{\mathcal{H}_X^i}$$

and

$$\sqrt{c_i}k_Y^i(y,y') = \langle \psi_Y^i(y), \psi_Y^i(y')\rangle_{\mathcal{H}_Y^i},$$

where $\psi_X^i(\cdot) : \mathcal{X} \to \mathcal{H}_X^i$ and $\psi_Y^i(\cdot) : \mathcal{Y} \to \mathcal{H}_Y^i$ are feature mappings, and $\mathcal{H}_X^i$ and $\mathcal{H}_Y^i$ are Hilbert spaces with respect to ψ_X^i and ψ_Y^i, respectively.

Let $\mathcal{H}_i = \mathcal{H}_X^i$, $\varphi_X^i(x) = \psi_X^i(x)$, and $\varphi_{YN}^i(y) = \sum_{j=1}^N \alpha_j g(x_j,y_j)\psi_X^i(x_j){\psi_Y^i}^\top(y_j)\psi_Y^i(y)$. Note that $\varphi_{YN}^i = \Gamma_N^i \psi_Y^i$, where $\Gamma_N^i = \sum_{j=1}^N \alpha_j g(x_j,y_j)\psi_X^i(x_j){\psi_Y^i}^\top(y_j)$ is a linear operator from $\mathcal{H}_Y^i$ to $\mathcal{H}_X^i = \mathcal{H}_i$. Thus, we have

$$\sum_{j=1}^N \alpha_j c_i (k_X(x,x_j)k_Y(y,y_j))^i g(x_j,y_j) = \langle \varphi_X^i(x), \varphi_{YN}^i(y)\rangle_{\mathcal{H}_i}.$$

Let $\mathcal{H} = \mathcal{H}_0 \times \mathcal{H}_1 \times \ldots \mathcal{H}_k \times \ldots$,

$$\begin{aligned}\varphi_X :& \mathcal{X} \to \mathcal{H}\\ & x \mapsto (\varphi_X^0(x), \varphi_X^1(x), \ldots, \varphi_X^k(x), \ldots),\end{aligned}$$

and

$$\begin{aligned}\varphi_{YN} :& \mathcal{Y} \to \mathcal{H}\\ & y \mapsto (\varphi_{YN}^0(y), \varphi_{YN}^1(y), \ldots, \varphi_{YN}^k(y), \ldots),\end{aligned}$$

we have

$$\begin{aligned}\sum_{j=1}^{N}\alpha_j\tilde{k}_P((x,y),(x_j,y_j)) &= \sum_{j=1}^{N}\alpha_j\sum_{i=0}^{\infty}c_i(k_X(x,x_j)k_Y(y,y_j))^i g(x_j,y_j)\\ &= \sum_{i=0}^{\infty}\sum_{j=1}^{N}\alpha_j c_i(k_X(x,x_j)k_Y(y,y_j))^i g(x_j,y_j)\\ &= \sum_{i=0}^{\infty}\langle\varphi_X^i(x),\varphi_{YN}^i(y)\rangle_{\mathcal{H}_i}\\ &= \langle\varphi_X(x),\varphi_{YN}(y)\rangle_{\mathcal{H}},\end{aligned}$$

where the inner product in $\mathcal{H}$ is naturally defined as $\sum_{i=0}^{\infty}\langle\cdot,\cdot\rangle_{\mathcal{H}_i}$. Note that $\sum_{i=0}^{\infty}\langle\cdot,\cdot\rangle_{\mathcal{H}_i}$ can be defined only when $(z_0,\dots,z_k,\dots)\in\mathcal{H},\sum_{i=0}^{\infty}\langle z_i,z_i\rangle_{\mathcal{H}_i}<\infty$. Obviously, for any $x\in\mathcal{X}$ and $y\in\mathcal{Y}$, $\varphi_X(x)$ and $\varphi_{YN}(y)$ satisfy this condition. ■

Lemma 7 *Given any two positive semi-definite kernels $k_X:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ and $k_Y:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$. Suppose $\psi_Y:\mathcal{Y}\to\mathcal{H}_Y$ is the feature mapping of $k_Y(\cdot,\cdot)$. $\mathcal{H}_Y$ is a Hilbert space endowed with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}_Y}$. Given any sets $\{x_i\}_{i=1}^N\subset\mathcal{X}$ and $\{y_i\}_{i=1}^N\subset\mathcal{Y}$, for an arbitrary $z\in\mathcal{H}_Y$, the following matrix inequality holds:*

$$\left(k_X(x_i,x_j)\langle\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\langle\psi_Y(y_j),z\rangle_{\mathcal{H}_Y}\right)_{N\times N}\preccurlyeq\left(k_X(x_i,x_j)k_Y(y_i,y_j)\langle z,z\rangle_{\mathcal{H}_Y}\right)_{N\times N}.$$

Proof Since $k_X(\cdot,\cdot)$ is a positive semi-definite kernel, following the conclusion given in Proposition 4 by Hofmann et al. (2008), we only need to prove

$$\left(k_Y(y_i,y_j)\langle z,z\rangle_{\mathcal{H}_Y}-\langle\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\langle\psi_Y(y_j),z\rangle_{\mathcal{H}_Y}\right)_{N\times N}$$

is positive semi-definite, which means given any $\{\alpha_i\}_{i=1}^N\subset\mathbb{R}$, we need to prove

$$\sum_{i,j=1}^{N}\alpha_i\alpha_j\left(k_Y(y_i,y_j)\langle z,z\rangle_{\mathcal{H}_Y}-\langle\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\langle\psi_Y(y_j),z\rangle_{\mathcal{H}_Y}\right)\geqslant 0.$$

Since

$$\begin{aligned}\sum_{i,j=1}^{N}\alpha_i\alpha_j k_Y(y_i,y_j)\langle z,z\rangle_{\mathcal{H}_Y} &= \sum_{i,j=1}^{N}\alpha_i\alpha_j\langle\psi_Y(y_i),\psi_Y(y_j)\rangle_{\mathcal{H}_Y}\langle z,z\rangle_{\mathcal{H}_Y}\\ &= \left\langle\sum_{i=1}^{N}\alpha_i\psi_Y(y_i),\sum_{i=1}^{N}\alpha_i\psi_Y(y_i)\right\rangle_{\mathcal{H}_Y}\langle z,z\rangle_{\mathcal{H}_Y},\end{aligned}$$

and

$$\sum_{i,j=1}^{N}\alpha_i\alpha_j\langle\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\langle\psi_Y(y_j),z\rangle_{\mathcal{H}_Y}=\left(\langle\sum_{i=1}^{N}\alpha_i\psi_Y(y_i),z\rangle_{\mathcal{H}_Y}\right)^2,$$

according to the Cauchy inequality, we reach the conclusion. ■

With Lemma 6 and Lemma 7, we can complete the proof of Theorem 4:

Proof Given a function $k(x,y)$ in $\mathcal{K}_P$, there is a sequence $\{k_N(x,y)\}$ in $\mathcal{K}_P$ such that

$$\begin{aligned} k_N(x,y) &= \sum_{j=1}^{N} \alpha_j k_P((x,y),(x_j,y_j)) \\ &= \sum_{j=1}^{N} \alpha_j g(x,y) \sum_{i=0}^{\infty} c_i \left(k_X(x,x_j) k_Y(y,y_j)\right)^i g(x_j,y_j) \\ &= g(x,y) \sum_{j=1}^{N} \alpha_j \tilde{k}_P((x,y),(x_j,y_j)); \\ k(x,y) &= \lim_{N\to\infty} k_N(x,y), \end{aligned}$$

where $\tilde{k}_P((x,y),(x_j,y_j)) = \sum_{i=0}^{\infty} c_i \left(k_X(x,x_j)k_Y(y,y_j)\right)^i g(x_j,y_j)$.

We try to prove that $k(x,y)$ is an S-function. Let $\tilde{k}_N(x,y) = \sum_{j=1}^{N} \alpha_j \tilde{k}_P((x,y),(x_j,y_j))$, $k(x,y) = \lim_{N\to\infty} k_N(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y) g(x,y) = \tilde{k}(x,y) g(x,y)$, where $\tilde{k}(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y)$.

According to Lemma 2, we only need to prove that $\tilde{k}(x,y)$ is an S-function. From the proof of Lemma 6, we know $\tilde{k}_N(x,y) = \langle \varphi_X(x), \varphi_{YN}(y) \rangle_{\mathcal{H}}$, where $\mathcal{H}$ is a Hilbert space determined by $\{\sqrt{c_i} k_X^i\}_{i=0}^{\infty}$.

$$\varphi_{YN}(y) = (\varphi_{YN}^0(y), \varphi_{YN}^1(y), \ldots, \varphi_{YN}^k(y), \ldots),$$

and we define $\mathcal{H}_Y = \mathcal{H}_Y^0 \times \ldots \mathcal{H}_Y^k \times \ldots$, $\mathcal{H}_X = \mathcal{H}_X^0 \times \ldots \mathcal{H}_X^k \times \ldots$, and

$$\begin{aligned} \Gamma_N : & \mathcal{H}_Y \to \mathcal{H}_X \\ & z = (z_0, \ldots, z_k, \ldots) \mapsto \Gamma_N(z) = (\Gamma_N^0 z_0, \ldots, \Gamma_N^k z_k, \ldots), \end{aligned}$$

where $\mathcal{H}_X^i$ and $\mathcal{H}_Y^i$ are the Hilbert spaces with respect to feature mappings $\psi_X^i(\cdot)$ and $\psi_Y^i(\cdot)$ defined by Mercer kernels $\sqrt{c_i} k_X^i$ and $\sqrt{c_i} k_Y^i$, respectively, $\langle \cdot, \cdot \rangle_{\mathcal{H}_Y^i}$ is the inner product defined in $\mathcal{H}_Y^i$, and $\Gamma_N^k z_k = \sum_{j=1}^{N} \alpha_j g(x_j,y_j) \psi_X^k(x_j) \langle \psi_Y^k(y_j), z_k \rangle_{\mathcal{H}_Y^k}$. The inner products for $\mathcal{H}_X = \mathcal{H}_X^0 \times \ldots \mathcal{H}_X^k \times \ldots$ and $\mathcal{H}_Y = \mathcal{H}_Y^0 \times \ldots \mathcal{H}_Y^k \times \ldots$ are naturally defined as $\sum_{i=0}^{\infty} \langle \cdot, \cdot \rangle_{\mathcal{H}_X^i}$ and $\sum_{i=0}^{\infty} \langle \cdot, \cdot \rangle_{\mathcal{H}_Y^i}$, respectively. Note that to make the inner products well defined, we require that input $(z_0, \ldots, z_k, \ldots)$ satisfies $\sum_{i=0}^{\infty} \langle z_i, z_i \rangle_{\mathcal{H}_Y^i} < \infty$. From the following proof, we will see that this condition will guarantee that $\sum_{i=0}^{\infty} \langle \Gamma_N^i z_i, \Gamma_N^i z_i \rangle_{\mathcal{H}_X^i} < \infty$. Thus, Γ_N is well defined.

Then the key point we need to prove is that $\{\Gamma_N\}$ is a Cauchy sequence. $\forall z \in \mathcal{H}_Y$, $||z||_{\mathcal{H}_Y} < \infty$,

$$\| \Gamma_N(z) \|_{\mathcal{H}_X}^2 = \sum_{i=0}^{\infty} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) \sqrt{c_i} k_X^i(x_k,x_j) \langle \psi_Y^i(y_k), z_i \rangle_{\mathcal{H}_Y^i} \langle \psi_Y^i(y_j), z_i \rangle_{\mathcal{H}_Y^i}.$$

Using the conclusion given by Lemma 7, we have

$$\begin{aligned}
&\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)\ \sqrt{c_i}k_X^i(x_k,x_j)\langle\psi_Y^i(y_k),z_i\rangle_{\mathcal{H}_Y^i}\langle\psi_Y^i(y_j),z_i\rangle_{\mathcal{H}_Y^i}\\
\leqslant&\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)c_i(k_X(x_k,x_j)k_Y(y_k,y_j))^i\langle z_i,z_i\rangle_{\mathcal{H}_Y^i}\\
\leqslant&\left(\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)c_i(k_X(x_k,x_j)k_Y(y_k,y_j))^i\right)\left(\sum_{i=0}^{\infty}\langle z_i,z_i\rangle_{\mathcal{H}_Y^i}\right).
\end{aligned}$$

Thus,

$$\| \Gamma_N(z) \|_{\mathcal{H}_X}^2\leqslant\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)c_i(k_X(x_k,x_j)k_Y(y_k,y_j))^i \| z \|_{\mathcal{H}_Y}^2 .$$

Therefore,

$$\begin{aligned}
\| \Gamma_N \|^2\leqslant&\sum_{i=0}^{\infty}\sum_{k,j=1}^{N}\alpha_k\alpha_j g(x_k,y_k)g(x_j,y_j)c_i(k_X(x_k,x_j)k_Y(y_k,y_j))^i\\
=&\sum_{k,j=1}^{N}\alpha_k\alpha_j k_P((x_k,y_k),(x_j,y_j)).
\end{aligned}$$

Note that $\sum_{k,j=1}^{N}\alpha_k\alpha_j k_P((x_k,y_k),(x_j,y_j))$ is just the square of the norm of $k_N(x,y)$ in $\mathcal{K}_P$. From the fact that $\{k_N(x,y)\}$ is a Cauchy sequence in $\mathcal{K}_P$, we know that $\{\Gamma_N\}$ is also a Cauchy sequence. Then there is a linear operator Γ which satisfies $\Gamma = \lim_{N\to\infty}\Gamma_N$. The convergence is in the norm. Thus, for every $(x,y)\in\mathcal{X}\times\mathcal{Y}$, $\lim_{N\to\infty}\tilde{k}_N(x,y) = \langle\varphi_X(x),\varphi_Y(y)\rangle_{\mathcal{H}} = \tilde{k}(x,y)$, where φ_Y is given by

$$(\Gamma^0\psi_Y^0,\dots,\Gamma^k\psi_Y^k,\dots).$$

■

Appendix B. Proof of Theorem 5

Theorem 5 *Given two finite sets of Mercer kernels* $K_X = \left\{k_i^X(x,x')\right\}_{i=1}^n$ *and* $K_Y = \left\{k_i^Y(y,y')\right\}_{i=1}^n$. *For any S-function* $g(x,y)$ *and* $\{c_i\}_{i=1}^n \subset \mathbb{R}^+$, k_M *defined below is an S-kernel.*

$$k_M((x,y),(x',y')) = \sum_{i=1}^{n} c_i\cdot g(x,y)k_i^X(x,x')k_i^Y(y,y')g(x',y').$$

First, since $\forall i$, $c_i g(x,y)k_i^X(x,x')k_i^Y(y,y')g(x',y')$ is a positive semi-definite kernel, and the summation of positive semi-definite kernels is also a positive semi-definite kernel, we know that $k_M((x,y),(x',y'))$ is a positive semi-definite kernel on $(\mathcal{X}\times\mathcal{Y})\times(\mathcal{X}\times\mathcal{Y})$.

Second, we need one more lemma to prove that all elements in the reproducing kernel Hilbert space $\mathcal{K}_M$ generated by k_M are S-functions:

Lemma 8 *Suppose that $g(x,y)$ is an S-function, and k_X and k_Y are Mercer kernels. Given any $\{(x_j,y_j)\}_{j=1}^N \subset \mathcal{X}\times\mathcal{Y}$, and any $\{\alpha_j\}_{j=1}^N \subset \mathbb{R}$, $\sum_{j=1}^N \alpha_j k_M((x,y),(x_i,y_i))$ is an S-function.*

Proof

$$\begin{aligned}\sum_{j=1}^N \alpha_j k_M((x,y),(x_j,y_j)) &= \sum_{j=1}^N \alpha_j g(x,y)\sum_{i=1}^n c_i k_i^X(x,x_j)k_i^Y(y,y_j)g(x_j,y_j)\\ &= g(x,y)\sum_{j=1}^N \alpha_j \tilde{k}_M((x,y),(x_j,y_j)).\end{aligned}$$

Since $g(x,y)$ is an S-function, according to Lemma 2, to prove $\sum_{j=1}^N \alpha_j k_M((x,y),(x_j,y_j))$ is an S-function, we only have to prove that $\sum_{j=1}^N \alpha_j \tilde{k}_M((x,y),(x_j,y_j))$ is an S-function.

For any $0 \leqslant i \leqslant n, i \in \mathbb{Z}$, since $\sqrt{c_i}k_i^X(x,x')$ and $\sqrt{c_i}k_i^Y(y,y')$ are both Mercer kernels, we obtain

$$\sqrt{c_i}k_i^X(x,x') = \langle \psi_X^i(x), \psi_X^i(x')\rangle_{\mathcal{H}_X^i}, \quad \sqrt{c_i}k_i^Y(y,y') = \langle \psi_Y^i(y), \psi_Y^i(y')\rangle_{\mathcal{H}_Y^i},$$

where $\psi_X^i(\cdot) : \mathcal{X} \to \mathcal{H}_X^i$ and $\psi_Y^i(\cdot) : \mathcal{Y} \to \mathcal{H}_Y^i$ are feature mappings, and $\mathcal{H}_X^i$ and $\mathcal{H}_Y^i$ are Hilbert spaces with respect to ψ_X^i and ψ_Y^i, respectively.

Let $\mathcal{H}_i = \mathcal{H}_X^i$, $\varphi_X^i(x) = \psi_X^i(x)$, and $\varphi_{YN}^i(y) = \sum_{j=1}^N \alpha_j g(x_j,y_j)\psi_X^i(x_j){\psi_Y^i}^\top(y_j)\psi_Y^i(y)$. Note that $\varphi_{YN}^i = \Gamma_N^i \psi_Y^i$, where $\Gamma_N^i = \sum_{j=1}^N \alpha_j g(x_j,y_j)\psi_X^i(x_j){\psi_Y^i}^\top(y_j)$ is a linear operator from $\mathcal{H}_Y^i$ to $\mathcal{H}_X^i = \mathcal{H}_i$. Thus, we have

$$\sum_{j=1}^N \alpha_j c_i k_i^X(x,x_j)k_i^Y(y,y_j)g(x_j,y_j) = \langle \varphi_X^i(x), \varphi_{YN}^i(y)\rangle_{\mathcal{H}_i}.$$

Let $\mathcal{H} = \mathcal{H}_1 \times \mathcal{H}_2 \times \ldots \mathcal{H}_n$,

$$\begin{aligned}\varphi_X(x) :& \mathcal{X} \to \mathcal{H}\\ & x \mapsto (\varphi_X^1(x), \varphi_X^2(x), \ldots, \varphi_X^n(x)),\end{aligned}$$

and

$$\begin{aligned}\varphi_{YN}(y) :& \mathcal{Y} \to \mathcal{H}\\ & y \mapsto (\varphi_{YN}^1(y), \varphi_{YN}^2(y), \ldots, \varphi_{YN}^n(y)),\end{aligned}$$

we have

$$\begin{aligned}\sum_{j=1}^N \alpha_j \tilde{k}_M((x,y),(x_j,y_j)) &= \sum_{j=1}^N \alpha_j \sum_{i=1}^n c_i k_i^X(x,x_j)k_i^Y(y,y_j)g(x_j,y_j)\\ &= \sum_{i=1}^n\sum_{j=1}^N \alpha_j c_i k_i^X(x,x_j)k_i^Y(y,y_j)g(x_j,y_j)\\ &= \sum_{i=1}^n \langle \varphi_X^i(x), \varphi_{YN}^i(y)\rangle_{\mathcal{H}_i}\\ &= \langle \varphi_X(x), \varphi_{YN}(y)\rangle_{\mathcal{H}},\end{aligned}$$

where the inner product in $\mathcal{H}$ is naturally defined as $\sum_{i=1}^n \langle\cdot,\cdot\rangle_{\mathcal{H}_i}$. ■

We prove that k_M is an S-kernel on the basis of Lemma 8 and Lemma 7:

Proof Given a function $k(x,y)$ in $\mathcal{K}_M$, there is a sequence $\{k_N(x,y)\}$ in $\mathcal{K}_M$ such that

$$\begin{aligned} k_N(x,y) &= \sum_{j=1}^{N} \alpha_j k_M((x,y),(x_j,y_j)) \\ &= \sum_{j=1}^{N} \alpha_j g(x,y) \sum_{i=1}^{n} c_i k_i^X(x,x_j) k_i^Y(y,y_j) g(x_j,y_j) \\ &= g(x,y) \sum_{j=1}^{N} \alpha_j \tilde{k}_M((x,y),(x_j,y_j)); \\ k(x,y) &= \lim_{N\to\infty} k_N(x,y). \end{aligned}$$

We try to prove that $k(x,y)$ is an S-function. Let $\tilde{k}_N(x,y) = \sum_{j=1}^{N} \alpha_j \tilde{k}_M((x,y),(x_j,y_j))$, $k(x,y) = \lim_{N\to\infty} k_N(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y) g(x,y) = \tilde{k}(x,y) g(x,y)$, where $\tilde{k}(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y)$.

According to Lemma 2, we only have to prove that $\tilde{k}(x,y)$ is an S-function. From Lemma 8, we know $\tilde{k}_N(x,y) = \langle \varphi_X(x), \varphi_{YN}(y) \rangle_{\mathcal{H}}$, where $\mathcal{H} = \mathcal{H}_1 \times \mathcal{H}_2 \times \ldots \mathcal{H}_n$ is a Hilbert space and $\mathcal{H}_i$ is the Hilbert space of the feature mapping of $\sqrt{c_i} k_i^X(\cdot,\cdot)$.

$$\varphi_{YN}(y) = (\varphi_{NY}^1(y), \varphi_{NY}^2(y), \ldots, \varphi_{NY}^n(y)),$$

and we define $\mathcal{H}_Y = \mathcal{H}_Y^1 \times \ldots \mathcal{H}_Y^n$, $\mathcal{H}_X = \mathcal{H}_X^1 \times \ldots \mathcal{H}_X^n$, and

$$\begin{aligned} \Gamma_N : & \mathcal{H}_Y \to \mathcal{H}_X \\ & z = (z_1, \ldots, z_n) \mapsto \Gamma_N(z) = (\Gamma_N^1 z_1, \ldots, \Gamma_N^n z_n), \end{aligned}$$

where $\mathcal{H}_X^i$ and $\mathcal{H}_Y^i$ are the Hilbert spaces with respect to feature mappings $\psi_X^i(\cdot)$ and $\psi_Y^i(\cdot)$ of Mercer kernels $\sqrt{c_i} k_i^X$ and $\sqrt{c_i} k_i^Y$, respectively, $\langle \cdot,\cdot \rangle_{\mathcal{H}_Y^i}$ is the inner product defined in $\mathcal{H}_Y^i$, and $\Gamma_N^i(z_i) = \sum_{j=1}^{N} \alpha_j g(x_j,y_j) \psi_X^i(x_j) \langle \psi_Y^i(y_j), z_i \rangle_{\mathcal{H}_Y^i}$.

Then the key point we need to prove is that $\{\Gamma_N\}$ is a Cauchy sequence. $\forall z \in \mathcal{H}_Y$,

$$\| \Gamma_N(z) \|_{\mathcal{H}_X}^2 = \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) \sqrt{c_i} k_i^X(x_k,x_j) \langle \psi_Y^i(y_k), z_i \rangle_{\mathcal{H}_Y^i} \langle \psi_Y^i(y_j), z_i \rangle_{\mathcal{H}_Y^i}.$$

Using the conclusion given by Lemma 7, we have

$$\begin{aligned} & \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) \sqrt{c_i} k_i^X(x_k,x_j) \langle \psi_Y^i(y_k), z_i \rangle_{\mathcal{H}_Y^i} \langle \psi_Y^i(y_j), z_i \rangle_{\mathcal{H}_Y^i} \\ \leqslant & \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) c_i k_i^X(x_k,x_j) k_i^Y(y_k,y_j) \langle z_i, z_i \rangle_{\mathcal{H}_Y^i} \\ \leqslant & \left(\sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) c_i k_i^X(x_k,x_j) k_i^Y(y_k,y_j) \right) \left(\sum_{i=1}^{n} \langle z_i, z_i \rangle_{\mathcal{H}_Y^i} \right). \end{aligned}$$

Thus,

$$\| \Gamma_N(z) \|^2_{\mathcal{H}_X} \leqslant \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) c_i k_i^X(x_k,x_j) k_i^Y(y_k,y_j) \| z \|^2_{\mathcal{H}_Y} .$$

Therefore,

$$\| \Gamma_N \|^2 \leqslant \sum_{i=1}^{n} \sum_{k,j=1}^{N} \alpha_k \alpha_j g(x_k,y_k) g(x_j,y_j) c_i k_i^X(x_k,x_j) k_i^Y(y_k,y_j) = \sum_{k,j=1}^{N} \alpha_k \alpha_j k_M((x_k,y_k),(x_j,y_j)).$$

Note that $\sum_{k,j=1}^N \alpha_k \alpha_j k_M((x_k,y_k),(x_j,y_j))$ is just the square of the norm of $k_N(x,y)$ in $\mathcal{K}_M$. From the fact that $\{k_N(x,y)\}$ is a Cauchy sequence in $\mathcal{K}_M$, we know that $\{\Gamma_N\}$ is also a Cauchy sequence. Then there is a linear operator Γ which satisfies $\Gamma = \lim_{N\to\infty} \Gamma_N$. The convergence is in the norm. Thus, for every $(x,y) \in \mathcal{X} \times \mathcal{Y}$, $\tilde{k}(x,y) = \lim_{N\to\infty} \tilde{k}_N(x,y) = \langle \varphi_X(x), \varphi_Y(y) \rangle_{\mathcal{H}}$, where φ_Y is given by

$$(\Gamma^1 \psi_Y^1, \ldots, \Gamma^n \psi_Y^n).$$

■

References

J. Abernethy, F. Bach, T. Evgeniou, and J.P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *JMLR '09*, 10:803–826, 2009.

S. Agarwal and P. Niyogi. Stability and generalization of bipartite ranking algorithms. In *COLT'05*, pages 32–47, 2005.

E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06*, pages 19–26, 2006.

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS'08*, pages 105–112, 2008.

F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML'04*, 2004.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *ICML '04*, pages 65–72, 2004.

D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD'00*, pages 407–416, 2000.

A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *WWW '09*, pages 511–520, 2009.

C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. In *NIPS'06*, pages 395–402. 2006.

Y. Cao, J. Xu, T.Y. Liu, H. Li, Y. Huang, and H.W. Hon. Adapting ranking SVM to document retrieval. In *SIGIR '06*, pages 186–193, 2006.

Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *ICML '07*, pages 129–136, 2007.

C. Cortes. Invited talk: Can learning kernels help performance? In *ICML'09*, page 161, 2009.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. on Information Theory*, 13(1):21–27, 1967.

K. Crammer and Y. Singer. Pranking with ranking. In *NIPS'01*, pages 641–647, 2001.

N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR' 07*, page 246, 2007.

H. Cui, J. Wen, J. Nie, and W. Ma. Query expansion by mining user logs. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):829–839, 2003.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JASIS*, 41:391–407, 1990.

S. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(4):325–327, April 1976.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *JMLR'03*, 4:933–969, 2003.

T. Gartner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *COLT '03*, page 129, 2003.

D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1371–1384, 2008.

D. Haussler. Convolution kernels on discrete structures. *Technical Report UCSC-CRL-99-10, Computer Science Dept., UC Santa Cruz.*, 1999.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *NIPS'99*, pages 115–132, 1999.

T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR' 99*, pages 50–57, 1999.

T. Hofmann, B. Scholkopf, and A.J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171, 2008.

K. Jarvelin and J. Kekalainen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR' 00*, pages 41–48, 2000.

T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142, 2002.

H. Kashima, K. Tsuda, and A. Inokuchi. Kernels for graphs. *Kernel Methods in Computational Biology*, pages 155–170, 2004.

N. Koide and Y. Yamashita. Asymmetric kernel method and its application to Fisher's discriminant. In *ICPR '06*, pages 820–824, 2006.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. In *ICML'02*, pages 323–330, 2002.

T.Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 2009. ISSN 1554-0669.

C. Micchelli and M. Pontil. Learning the kernel function via regularization. *JMLR'05*, 6:1099–1125, 2005.

M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *SIGIR '98*, pages 206–214, 1998.

C.S. Ong, A.J. Smola, and R.C. Williamson. Learning the kernel with hyperkernels. *JMLR '05*, 6: 1043–1071, 2005.

J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *SIGIR' 98*, pages 275–281, 1998.

S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC*, 1994.

C. Rudin, C. Cortes, M. Mohri, and R. E. Schapire. Margin-based ranking meets boosting in the middle. In *COLT'05*, pages 63–78, 2005.

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Readings in Information Retrieval*, pages 355–364, 1997.

G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840.

B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. the MIT Press, 2002.

V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML'09*, page 134, 2009.

S.V.N. Vishwanathan and A.J. Smola. Binet-cauchy kernels. In *NIPS '04*, 2004.

C. Watkins. Dynamic alignment kernels. In *NIPS '99*, 1999.

J.R. Wen, J.Y. Nie, and H.J. Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 20(1): 59–81, 2002. ISSN 1046-8188.

J. Xu and W.B. Croft. Query expansion using local and global document analysis. In *SIGIR '96*, pages 4–11, 1996.

J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR' 07*, pages 391–398, 2007.

J. Xu, H. Li, and Z.L. Zhong. Relevance ranking using kernels. In *AIRS '10*, 2010.

C.X. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

S.K. Zhou. Trace and determinant kernels between matrices. In *NIPS '04*, 2004.

Z.M. Zhuang and S. Cucerzan. Re-ranking search results using query logs. In *CIKM '06*, pages 860–861, 2006.

Journal of Machine Learning Research 12 (2011) 1459-1500 Submitted 11/09; Revised 10/10; Published 5/11

Computationally Efficient Convolved Multiple Output Gaussian Processes

Mauricio A. Álvarez[*] MALVAREZ@UTP.EDU.CO
School of Computer Science
University of Manchester
Manchester, UK, M13 9PL

Neil D. Lawrence[†] N.LAWRENCE@SHEFFIELD.AC.UK
School of Computer Science
University of Sheffield
Sheffield, S1 4DP

Editor: Carl Edward Rasmussen

Abstract

Recently there has been an increasing interest in regression methods that deal with multiple outputs. This has been motivated partly by frameworks like multitask learning, multisensor networks or structured output data. From a Gaussian processes perspective, the problem reduces to specifying an appropriate covariance function that, whilst being positive semi-definite, captures the dependencies between all the data points and across all the outputs. One approach to account for non-trivial correlations between outputs employs convolution processes. Under a latent function interpretation of the convolution transform we establish dependencies between output variables. The main drawbacks of this approach are the associated computational and storage demands. In this paper we address these issues. We present different efficient approximations for dependent output Gaussian processes constructed through the convolution formalism. We exploit the conditional independencies present naturally in the model. This leads to a form of the covariance similar in spirit to the so called PITC and FITC approximations for a single output. We show experimental results with synthetic and real data, in particular, we show results in school exams score prediction, pollution prediction and gene expression data.

Keywords: Gaussian processes, convolution processes, efficient approximations, multitask learning, structured outputs, multivariate processes

1. Introduction

Accounting for dependencies between model outputs has important applications in several areas. In sensor networks, for example, missing signals from failing sensors may be predicted due to correlations with signals acquired from other sensors (Osborne et al., 2008). In geostatistics, prediction of the concentration of heavy pollutant metals (for example, Copper), that are expensive to measure, can be done using inexpensive and oversampled variables (for example, pH) as a proxy (Goovaerts, 1997). Within the machine learning community this approach is sometimes known as multitask learning. The idea in multitask learning is that information shared between the tasks leads to im-

$*$. Also in Faculty of Engineering, Universidad Tecnológica de Pereira, Pereira, Colombia, 660003.
†. Also at the Sheffield Institute for Translational Neuroscience, Sheffield, UK, S10 2HQ.

proved performance in comparison to learning the same tasks individually (Caruana, 1997; Bonilla et al., 2008).

In this paper, we consider the problem of modeling related outputs in a Gaussian process (GP). A Gaussian process specifies a prior distribution over functions. When using a GP for multiple related outputs, our purpose is to develop a prior that expresses correlation between the outputs. This information is encoded in the covariance function. The class of valid covariance functions is the same as the class of reproducing kernels.[1] Such kernel functions for single outputs are widely studied in machine learning (see, for example, Rasmussen and Williams, 2006). More recently the community has begun to turn its attention to covariance functions for multiple outputs. One of the paradigms that has been considered (Teh et al., 2005; Osborne et al., 2008; Bonilla et al., 2008) is known in the geostatistics literature as *the linear model of coregionalization* (LMC) (Journel and Huijbregts, 1978; Goovaerts, 1997). In the LMC, the covariance function is expressed as the sum of Kronecker products between *coregionalization matrices* and a set of underlying covariance functions. The correlations across the outputs are expressed in the coregionalization matrices, while the underlying covariance functions express the correlation between different data points.

Multitask learning has also been approached from the perspective of *regularization theory* (Evgeniou and Pontil, 2004; Evgeniou et al., 2005). These *multitask kernels* are obtained as generalizations of the regularization theory to vector-valued functions. They can also be seen as examples of LMCs applied to linear transformations of the input space.

In the linear model of coregionalization each output can be thought of as an instantaneous mixing of the underlying signals/processes. An alternative approach to constructing covariance functions for multiple outputs employs *convolution processes* (CP). To obtain a CP in the single output case, the output of a given process is convolved with a smoothing kernel function. For example, a white noise process may be convolved with a smoothing kernel to obtain a covariance function (Barry and Ver Hoef, 1996; Ver Hoef and Barry, 1998). Ver Hoef and Barry (1998) and then Higdon (2002) noted that if a single input process was convolved with different smoothing kernels to produce different outputs, then correlation between the outputs could be expressed. This idea was introduced to the machine learning audience by Boyle and Frean (2005). We can think of this approach to generating multiple output covariance functions as a non-instantaneous mixing of the base processes.

The convolution process framework is an elegant way for constructing dependent output processes. However, it comes at the price of having to consider the full covariance function of the joint GP. For D output dimensions and N data points the covariance matrix scales as DN leading to $O(N^3D^3)$ computational complexity and $O(N^2D^2)$ storage. We are interested in exploiting the richer class of covariance structures allowed by the CP framework, but reducing the additional computational overhead they imply.

In this paper, we propose different efficient approximations for the full covariance matrix involved in the multiple output convolution process. We exploit the fact that, in the convolution framework, each of the outputs is conditional independent of all others if the input process is fully observed. This leads to an approximation that turns out to be strongly related to the partially independent training conditional (PITC) (Quiñonero-Candela and Rasmussen, 2005) approximation for a single output GP. This analogy inspires us to consider a further conditional independence

1. In this paper we will use kernel to refer to both reproducing kernels and smoothing kernels. Reproducing kernels are those used in machine learning that conform to Mercer's theorem. Smoothing kernels are kernel functions which are convolved with a signal to create a smoothed version of that signal.

assumption across data points. This leads to an approximation which shares the form of the fully independent training conditional (FITC) approximation (Snelson and Ghahramani, 2006; Quiñonero-Candela and Rasmussen, 2005). This reduces computational complexity to $O(NDK^2)$ and storage to $O(NDK)$ with K representing a user specified value for the number of inducing points in the approximation.

The rest of the paper is organized as follows. First we give a more detailed review of related work, with a particular focus on relating multiple output work in machine learning to other fields. Despite the fact that there are several other approaches to multitask learning (see for example Caruana, 1997, Heskes, 2000, Bakker and Heskes, 2003, Xue et al., 2007 and references therein), in this paper, we focus our attention to those that address the problem of constructing the covariance or kernel function for multiple outputs, so that it can be employed, for example, together with Gaussian process prediction. Then we review the convolution process approach in Section 3 and Section 4. We demonstrate how our conditional independence assumptions can be used to reduce the computational load of inference in Section 5. Experimental results are shown in Section 6 and finally some discussion and conclusions are presented in Section 7.

2. Related Work

In geostatistics, multiple output models are used to model the co-occurrence of minerals or pollutants in a spatial field. Many of the ideas for constructing covariance functions for multiple outputs have first appeared within the geostatistical literature, where they are known as linear models of coregionalization (LMC). We present the LMC and then review how several models proposed in the machine learning literature can be seen as special cases of the LMC.

2.1 The Linear Model of Coregionalization

The term linear model of coregionalization refers to models in which the outputs are expressed as *linear* combinations of independent random functions. If the independent random functions are Gaussian processes then the resulting model will also be a Gaussian process with a positive semi-definite covariance function. Consider a set of D output functions $\{f_d(\mathbf{x})\}_{d=1}^{D}$ where $\mathbf{x} \in \Re^p$ is the input domain. In a LMC each output function, $f_d(\mathbf{x})$, is expressed as (Journel and Huijbregts, 1978)

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} a_{d,q} u_q(\mathbf{x}). \tag{1}$$

Under the GP interpretation of the LMC, the functions $\{u_q(\mathbf{x})\}_{q=1}^{Q}$ are taken (without loss of generality) to be drawn from a zero-mean GP with $\text{cov}[u_q(\mathbf{x}), u_{q'}(\mathbf{x}')] = k_q(\mathbf{x},\mathbf{x}')$ if $q = q'$ and zero otherwise. Some of these base processes might have the same covariance, this is $k_q(\mathbf{x},\mathbf{x}') = k_{q'}(\mathbf{x},\mathbf{x}')$, but they would still be independently sampled. We can group together the base processes that share latent functions (Journel and Huijbregts, 1978; Goovaerts, 1997), allowing us to express a given output as

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} a_{d,q}^{i} u_q^{i}(\mathbf{x}), \tag{2}$$

where the functions $\{u_q^i(\mathbf{x})\}_{i=1}^{R_q}$, $i = 1, \dots, R_q$, represent the latent functions that share the same covariance function $k_q(\mathbf{x}, \mathbf{x}')$. There are now Q groups of functions, each member of a group shares the same covariance, but is sampled independently.

In geostatistics it is common to simplify the analysis of these models by assuming that the processes $f_d(\mathbf{x})$ are stationary and ergodic (Cressie, 1993). The stationarity and ergodicity conditions are introduced so that the prediction stage can be realized through an optimal linear predictor using a single realization of the process (Cressie, 1993). Such linear predictors receive the general name of *cokriging*. The cross covariance between any two functions $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x})$ is given in terms of the covariance functions for $u_q^i(\mathbf{x})$

$$\operatorname{cov}[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = \sum_{q=1}^{Q}\sum_{q'=1}^{Q}\sum_{i=1}^{R_q}\sum_{i'=1}^{R_q} a_{d,q}^i a_{d',q'}^{i'} \operatorname{cov}[u_q^i(\mathbf{x}), u_{q'}^{i'}(\mathbf{x}')].$$

Because of the independence of the latent functions $u_q^i(\mathbf{x})$, the above expression can be reduced to

$$\operatorname{cov}[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = \sum_{q=1}^{Q}\sum_{i=1}^{R_q} a_{d,q}^i a_{d',q}^i k_q(\mathbf{x}, \mathbf{x}') = \sum_{q=1}^{Q} b_{d,d'}^q k_q(\mathbf{x}, \mathbf{x}'), \tag{3}$$

with $b_{d,d'}^q = \sum_{i=1}^{R_q} a_{d,q}^i a_{d',q}^i$.

For a number N of input vectors, let $\mathbf{f}_d$ be the vector of values from the output d evaluated at $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$. If each output has the same set of inputs the system is known as *isotopic*. In general, we can allow each output to be associated with a different set of inputs, $\mathbf{X}^{(d)} = \{\mathbf{x}_n^{(d)}\}_{n=1}^{N_d}$, this is known as *heterotopic*.[2] For notational simplicity, we restrict ourselves to the isotopic case, but our analysis can also be completed for heterotopic setups. The covariance matrix for $\mathbf{f}_d$ is obtained expressing Equation (3) as

$$\operatorname{cov}[\mathbf{f}_d, \mathbf{f}_{d'}] = \sum_{q=1}^{Q}\sum_{i=1}^{R_q} a_{d,q}^i a_{d',q}^i \mathbf{K}_q = \sum_{q=1}^{Q} b_{d,d'}^q \mathbf{K}_q,$$

where $\mathbf{K}_q \in \Re^{N\times N}$ has entries given by computing $k_q(\mathbf{x}, \mathbf{x}')$ for all combinations from $\mathbf{X}$. We now define $\mathbf{f}$ to be a stacked version of the outputs so that $\mathbf{f} = [\mathbf{f}_1^\top, \dots, \mathbf{f}_D^\top]^\top$. We can now write the covariance matrix for the joint process over $\mathbf{f}$ as

$$\mathbf{K}_{\mathbf{f},\mathbf{f}} = \sum_{q=1}^{Q} \mathbf{A}_q \mathbf{A}_q^\top \otimes \mathbf{K}_q = \sum_{q=1}^{Q} \mathbf{B}_q \otimes \mathbf{K}_q, \tag{4}$$

where the symbol $\otimes$ denotes the Kronecker product, $\mathbf{A}_q \in \Re^{D\times R_q}$ has entries $a_{d,q}^i$ and $\mathbf{B}_q = \mathbf{A}_q\mathbf{A}_q^\top \in \Re^{D\times D}$ has entries $b_{d,d'}^q$ and is known as the *coregionalization matrix*. The covariance matrix $\mathbf{K}_{\mathbf{f},\mathbf{f}}$ is positive semi-definite as long as the coregionalization matrices $\mathbf{B}_q$ are positive semi-definite and $k_q(\mathbf{x}, \mathbf{x}')$ is a valid covariance function. By definition, coregionalization matrices $\mathbf{B}_q$ fulfill the positive semi-definiteness requirement. The covariance functions for the latent processes, $k_q(\mathbf{x}, \mathbf{x}')$, can simply be chosen from the wide variety of covariance functions (reproducing kernels) that are

2. These names come from geostatistics.

used for the single output case. Examples include the squared exponential (sometimes called the Gaussian kernel or RBF kernel) and the Matérn class of covariance functions (see Rasmussen and Williams, 2006, Chapter 4).

The linear model of coregionalization represents the covariance function as a product of the contributions of two covariance functions. One of the covariance functions models the dependence between the functions independently of the input vector **x**, this is given by the coregionalization matrix $\mathbf{B}_q$, whilst the other covariance function models the input dependence independently of the particular set of functions $f_d(\mathbf{x})$, this is the covariance function $k_q(\mathbf{x},\mathbf{x}')$.

We can understand the LMC by thinking of the functions having been generated as a two step process. Firstly we sample a set of independent processes from the covariance functions given by $k_q(\mathbf{x},\mathbf{x}')$, taking R_q independent samples for each $k_q(\mathbf{x},\mathbf{x}')$. We now have $R=\sum_{q=1}^{Q}R_q$ independently sampled functions. These functions are *instantaneously mixed*[3] in a linear fashion. In other words the output functions are derived by application of a scaling and a rotation to an output space of dimension D.

2.1.1 INTRINSIC COREGIONALIZATION MODEL

A simplified version of the LMC, known as the intrinsic coregionalization model (ICM) (Goovaerts, 1997), assumes that the elements $b^q_{d,d'}$ of the coregionalization matrix $\mathbf{B}_q$ can be written as $b^q_{d,d'}=\upsilon_{d,d'}b_q$. In other words, as a scaled version of the elements b_q which do not depend on the particular output functions $f_d(\mathbf{x})$. Using this form for $b^q_{d,d'}$, Equation (3) can be expressed as

$$\operatorname{cov}[f_d(\mathbf{x}),f_{d'}(\mathbf{x}')]=\sum_{q=1}^{Q}\upsilon_{d,d'}b_qk_q(\mathbf{x},\mathbf{x}')=\upsilon_{d,d'}\sum_{q=1}^{Q}b_qk_q(\mathbf{x},\mathbf{x}').$$

The covariance matrix for **f** takes the form

$$\mathbf{K}_{\mathbf{f},\mathbf{f}}=\boldsymbol{\Upsilon}\otimes\mathbf{K}, \tag{5}$$

where $\boldsymbol{\Upsilon}\in\Re^{D\times D}$, with entries $\upsilon_{d,d'}$, and $\mathbf{K}=\sum_{q=1}^{Q}b_q\mathbf{K}_q$ is an equivalent valid covariance function.

The intrinsic coregionalization model can also be seen as a linear model of coregionalization where we have $Q=1$. In such case, Equation (4) takes the form

$$\mathbf{K}_{\mathbf{f},\mathbf{f}}=\mathbf{A}_1\mathbf{A}_1^{\top}\otimes\mathbf{K}_1=\mathbf{B}_1\otimes\mathbf{K}_1, \tag{6}$$

where the coregionalization matrix $\mathbf{B}_1$ has elements $b^1_{d,d'}=\sum_{i=1}^{R_1}a^i_{d,1}a^i_{d',1}$. The value of R_1 determines the rank of the matrix $\mathbf{B}_1$.

As pointed out by Goovaerts (1997), the ICM is much more restrictive than the LMC since it assumes that each basic covariance $k_q(\mathbf{x},\mathbf{x}')$ contributes equally to the construction of the autocovariances and cross covariances for the outputs.

3. The term instantaneous mixing is taken from blind source separation. Of course, if the underlying processes are not temporal but spatial, instantaneous is not being used in its original sense. However, it allows us to distinguish this mixing from convolutional mixing.

2.1.2 Linear Model of Coregionalization in Machine Learning

Several of the approaches to multiple output learning in machine learning based on kernels can be seen as examples of the linear model of coregionalization.

Semiparametric latent factor model. The semiparametric latent factor model (SLFM) proposed by Teh et al. (2005) turns out to be a simplified version of Equation (4). In particular, if $R_q = 1$ (see Equation 1), we can rewrite Equation (4) as

$$\mathbf{K}_{\mathbf{f},\mathbf{f}} = \sum_{q=1}^{Q} \mathbf{a}_q \mathbf{a}_q^\top \otimes \mathbf{K}_q,$$

where $\mathbf{a}_q \in \Re^{D\times 1}$ with elements $a_{d,q}$. With some algebraic manipulations that exploit the properties of the Kronecker product[4] we can write

$$\mathbf{K}_{\mathbf{f},\mathbf{f}} = \sum_{q=1}^{Q} (\mathbf{a}_q \otimes \mathbf{I}_N)\mathbf{K}_q(\mathbf{a}_q^\top \otimes \mathbf{I}_N) = (\widetilde{\mathbf{A}} \otimes \mathbf{I}_N)\widetilde{\mathbf{K}}(\widetilde{\mathbf{A}}^\top \otimes \mathbf{I}_N),$$

where $\mathbf{I}_N$ is the N-dimensional identity matrix, $\widetilde{\mathbf{A}} \in \Re^{D\times Q}$ is a matrix with columns $\mathbf{a}_q$ and $\widetilde{\mathbf{K}} \in \Re^{QN\times QN}$ is a block diagonal matrix with blocks given by $\mathbf{K}_q$.

The functions $u_q(\mathbf{x})$ are considered to be latent factors and the semiparametric name comes from the fact that it is combining a nonparametric model, this is a Gaussian process, with a parametric linear mixing of the functions $u_q(\mathbf{x})$. The kernel for each basic process q, $k_q(\mathbf{x},\mathbf{x}')$, is assumed to be of Gaussian type with a different length scale per input dimension. For computational speed up the informative vector machine (IVM) is employed (Lawrence et al., 2003).

Multi-task Gaussian processes. The intrinsic coregionalization model has been employed in Bonilla et al. (2008) for multitask learning. We refer to this approach as multi-task Gaussian processes (MTGP). The covariance matrix is expressed as $\mathbf{K}_{\bar{\mathbf{f}}(\mathbf{x}),\bar{\mathbf{f}}(\mathbf{x}')} = K^f \otimes k(\mathbf{x},\mathbf{x}')$, with $\bar{\mathbf{f}}(\mathbf{x}) = [f_1(\mathbf{x}),\ldots,f_D(\mathbf{x})]^\top$, K^f being constrained positive semi-definite and $k(\mathbf{x},\mathbf{x}')$ a covariance function over inputs. It can be noticed that this expression has is equal to the one in (5), when it is evaluated for $\mathbf{x},\mathbf{x}' \in \mathbf{X}$. In Bonilla et al. (2008), K^f (equal to $\boldsymbol{\Upsilon}$ in Equation 5 or $\mathbf{B}_1$ in Equation 6) expresses the correlation between tasks or inter-task dependencies and it is represented through a probabilistic principal component analysis (PPCA) model. In turn, the spectral factorization in the PPCA model is replaced by an incomplete Cholesky decomposition to keep numerical stability, so that $K^f \approx \widetilde{\mathbf{L}}\widetilde{\mathbf{L}}^\top$, where $\widetilde{\mathbf{L}} \in \Re^{D\times R_1}$. An application of MTGP for obtaining the inverse dynamics of a robotic manipulator was presented in Chai et al. (2009).

It can be shown that if the outputs are considered to be noise-free, prediction using the intrinsic coregionalization model under an isotopic data case is equivalent to independent prediction over each output (Helterbrand and Cressie, 1994). This circumstance is also known as autokrigeability (Wackernagel, 2003) and it can also be seen as the cancellation of inter-task transfer (Bonilla et al., 2008).

Multi-output Gaussian processes. The intrinsic coregionalization model has been also used in Osborne et al. (2008). Matrix $\boldsymbol{\Upsilon}$ in Expression (5) is assumed to be of the spherical parametrisation kind, $\boldsymbol{\Upsilon} = \mathrm{diag}(\mathbf{e})\mathbf{S}^\top\mathbf{S}\,\mathrm{diag}(\mathbf{e})$, where $\mathbf{e}$ gives a description for the length scale of each output variable and $\mathbf{S}$ is an upper triangular matrix whose i-th column is associated with particular spherical

4. See Brookes (2005) for a nice overview.

coordinates of points in $\Re^i$ (for details see Osborne and Roberts, 2007, Section 3.4). Function $k(\mathbf{x},\mathbf{x}')$ is represented through a Mátern kernel, where different parametrisations of the covariance allow the expression of periodic and non-periodic terms. Sparsification for this model is obtained using an IVM style approach.

Multi-task kernels in regularization theory. Kernels for multiple outputs have also been studied in the context of regularization theory. The approach is based mainly on the definition of kernels for multitask learning provided in Evgeniou and Pontil (2004) and Evgeniou et al. (2005), derived based on the theory of kernels for vector-valued functions. Let $\mathcal{D} = \{1,\dots,D\}$. According to Evgeniou et al. (2005), the following lemma can be used to construct multitask kernels,

Lemma 1 *If G is a kernel on $\mathcal{T}\times\mathcal{T}$ and, for every $d\in\mathcal{D}$ there are prescribed mappings $\Phi_d : \mathcal{X}\to\mathcal{T}$ such that*

$$k_{d,d'}(\mathbf{x},\mathbf{x}') = k((\mathbf{x},d),(\mathbf{x}',d')) = G(\Phi_d(\mathbf{x}),\Phi_{d'}(\mathbf{x}')), \quad \mathbf{x},\mathbf{x}'\in\Re^p,\ d,d'\in\mathcal{D},$$

then $k(\cdot)$ is a multitask or multioutput kernel.

A linear multitask kernel can be obtained if we set $\mathcal{T}=\Re^m$, $\Phi_d(\mathbf{x}) = \mathbf{C}_d\mathbf{x}$ with $\Phi_d\in\Re^m$ and $G:\Re^m\times\Re^m\to\Re$ as the polynomial kernel $G(\mathbf{z},\mathbf{z}') = (\mathbf{z}^\top\mathbf{z}')^n$ with $n=1$, leading to $k_{d,d'}(\mathbf{x},\mathbf{x}') = \mathbf{x}^\top\mathbf{C}_d^\top\mathbf{C}_{d'}\mathbf{x}'$. The lemma above can be seen as the result of applying kernel properties to the mapping $\Phi_d(\mathbf{x})$ (see Genton, 2001, p. 2). Notice that this corresponds to a generalization of the semiparametric latent factor model where each output is expressed through its own basic process acting over the linear transformation $\mathbf{C}_d\mathbf{x}$, this is, $u_d(\Phi_d(\mathbf{x})) = u_d(\mathbf{C}_d\mathbf{x})$. In general, it can be obtained from $f_d(\mathbf{x}) = \sum_{q=1}^{D} a_{d,q}u_q(\Phi_q(\mathbf{x}))$, where $a_{d,q}=1$ if $d=q$ or zero, otherwise.

A more detailed analysis of the LMC and more connections with other methods in statistics and machine learning can be found in Álvarez et al. (2011b).

3. Convolution Processes for Multiple Outputs

The approaches introduced above all involve some form of instantaneous mixing of a series of independent processes to construct correlated processes. Instantaneous mixing has some limitations. If we wanted to model two output processes in such a way that one process was a blurred version of the other, we cannot achieve this through instantaneous mixing. We can achieve blurring through convolving a base process with a smoothing kernel. If the base process is a Gaussian process, it turns out that the convolved process is also a Gaussian process. We can therefore exploit convolutions to construct covariance functions (Barry and Ver Hoef, 1996; Ver Hoef and Barry, 1998; Higdon, 1998, 2002). A recent review of several extensions of this approach for the single output case is presented in Calder and Cressie (2007). Applications include the construction of nonstationary covariances (Higdon, 1998; Higdon et al., 1998; Fuentes, 2002a,b; Paciorek and Schervish, 2004) and spatiotemporal covariances (Wikle et al., 1998; Wikle, 2002, 2003).

Ver Hoef and Barry (1998) first, and Higdon (2002) later, suggested using convolutions to construct multiple output covariance functions. The approach was introduced to the machine learning community by Boyle and Frean (2005). Consider again a set of D functions $\{f_d(\mathbf{x})\}_{d=1}^D$. Now each function could be expressed through a convolution integral between a smoothing kernel, $\{G_d(\mathbf{x})\}_{d=1}^D$, and a latent function $u(\mathbf{x})$,

$$f_d(\mathbf{x}) = \int_{\mathcal{X}} G_d(\mathbf{x}-\mathbf{z})u(\mathbf{z})\mathrm{d}\mathbf{z}. \tag{7}$$

More generally, and in a similar way to the linear model of coregionalization, we can consider the influence of more than one latent function, $u_q^i(\mathbf{z})$, with $q = 1, \dots, Q$ and $i = 1, \dots, R_q$ to obtain

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} \int_{\mathcal{X}} G_{d,q}^i(\mathbf{x} - \mathbf{z}) u_q^i(\mathbf{z}) \mathrm{d}\mathbf{z}.$$

As in the LMC, there are Q groups of functions, each member of the group has the same covariance $k_q(\mathbf{x}, \mathbf{x}')$, but is sampled independently. Under the same independence assumptions used in the LMC, the covariance between $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x}')$ follows

$$\operatorname{cov}\left[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')\right] = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} \int_{\mathcal{X}} G_{d,q}^i(\mathbf{x} - \mathbf{z}) \int_{\mathcal{X}} G_{d',q}^i(\mathbf{x}' - \mathbf{z}') k_q(\mathbf{z}, \mathbf{z}') \mathrm{d}\mathbf{z}' \mathrm{d}\mathbf{z}. \tag{8}$$

Specifying $G_{d,q}^i(\mathbf{x} - \mathbf{z})$ and $k_q(\mathbf{z}, \mathbf{z}')$ in (8), the covariance for the outputs $f_d(\mathbf{x})$ can be constructed indirectly. Note that if the smoothing kernels are taken to be the Dirac delta function such that,

$$G_{d,q}^i(\mathbf{x} - \mathbf{z}) = a_{d,q}^i \delta(\mathbf{x} - \mathbf{z}),$$

where $\delta(\cdot)$ is the Dirac delta function, the double integral is easily solved and the linear model of coregionalization is recovered. This matches to the concept of *instantaneous mixing* we introduced to describe the LMC. In a convolutional process the mixing is more general, for example the latent process could be smoothed for one output, but not smoothed for another allowing correlated output functions of different length scales.

The traditional approach to convolution processes in statistics and signal processing is to assume that the latent functions $u_q(\mathbf{z})$ are independent white Gaussian noise processes, $k_q(\mathbf{z}, \mathbf{z}') = \sigma_q^2 \delta(\mathbf{z} - \mathbf{z}')$. This allows us to simplify (8) as

$$\operatorname{cov}\left[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')\right] = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} \sigma_q^2 \int_{\mathcal{X}} G_{d,q}^i(\mathbf{x} - \mathbf{z}) G_{d',q}^i(\mathbf{x}' - \mathbf{z}) \mathrm{d}\mathbf{z}.$$

In general, though, we can consider any type of latent process, for example, we could assume GPs for the latent functions with general covariances $k_q(\mathbf{z}, \mathbf{z}')$.

As well as this covariance across outputs, the covariance between the latent function, $u_q^i(\mathbf{z})$, and any given output, $f_d(\mathbf{x})$, can be computed,

$$\operatorname{cov}\left[f_d(\mathbf{x}), u_q^i(\mathbf{z})\right] = \int_{\mathcal{X}} G_{d,q}^i(\mathbf{x} - \mathbf{z}') k_q(\mathbf{z}', \mathbf{z}) \mathrm{d}\mathbf{z}'. \tag{9}$$

Additionally, we can corrupt each of the outputs of the convolutions with an independent process (which could also include a noise term), $w_d(\mathbf{x})$, to obtain

$$y_d(\mathbf{x}) = f_d(\mathbf{x}) + w_d(\mathbf{x}). \tag{10}$$

The covariance between two different outputs $y_d(\mathbf{x})$ and $y_{d'}(\mathbf{x}')$ is then recovered as

$$\operatorname{cov}\left[y_d(\mathbf{x}), y_{d'}(\mathbf{x}')\right] = \operatorname{cov}\left[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')\right] + \operatorname{cov}\left[w_d(\mathbf{x}), w_{d'}(\mathbf{x}')\right] \delta_{d,d'},$$

where $\delta_{d,d'}$ is the Kronecker delta function.[5]

As mentioned before, Ver Hoef and Barry (1998) and Higdon (2002) proposed the direct use of convolution processes for constructing multiple output Gaussian processes. Lawrence et al. (2007) arrive at a similar construction from solving a physical model: a first order differential equation (see also Gao et al., 2008). This idea of using physical models to inspire multiple output systems has been further extended in Álvarez et al. (2009) who give examples using the heat equation and a second order system. A different approach using Kalman Filtering ideas has been proposed in Calder (2003, 2007). Calder proposed a model that incorporates dynamical systems ideas to the process convolution formalism. Essentially, the latent processes are of two types: random walks and independent cyclic second-order autoregressions. With this formulation, it is possible to construct a multivariate output process using convolutions over these latent processes. Particular relationships between outputs and latent processes are specified using a special transformation matrix ensuring that the outputs are invariant under invertible linear transformations of the underlying factor processes (this matrix is similar in spirit to the sensitivity matrix of Lawrence et al. (2007) and it is given a particular form so that not all latent processes affect the whole set of outputs).

Bayesian kernel methods. The convolution process is closely related to the Bayesian kernel method (Pillai et al., 2007; Liang et al., 2009) for constructing reproducible kernel Hilbert spaces (RKHS), assigning priors to signed measures and mapping these measures through integral operators. In particular, define the following space of functions,

$$\mathcal{F} = \left\{ f \middle| f(x) = \int_{\mathcal{X}} G(x,z)\gamma(\mathrm{d}z),\ \gamma \in \Gamma \right\},$$

for some space $\Gamma \subseteq \mathcal{B}(\mathcal{X})$ of signed Borel measures. In Pillai et al. (2007, Proposition 1), the authors show that for $\Gamma = \mathcal{B}(\mathcal{X})$, the space of all signed Borel measures, $\mathcal{F}$ corresponds to a RKHS. Examples of these measures that appear in the form of stochastic processes include Gaussian processes, Dirichlet processes and Lévy processes. This framework can be extended for the multiple output case, expressing the outputs as

$$f_d(x) = \int_{\mathcal{X}} G_d(x,z)\gamma(\mathrm{d}z).$$

The analysis of the mathematical properties of such spaces of functions is beyond the scope of this paper and is postponed for future work.

Other connections of the convolution process approach with methods in statistics and machine learning are further explored in Álvarez et al. (2011b).

A general purpose convolution kernel for multiple outputs. A simple general purpose kernel for multiple outputs based on the convolution integral can be constructed assuming that the kernel smoothing function, $G_{d,q}(\mathbf{x})$, and the covariance for the latent function, $k_q(\mathbf{x},\mathbf{x}')$, follow both a Gaussian form. A similar construction using a Gaussian form for $G(\mathbf{x})$ and a white noise process for $u(\mathbf{x})$ has been used in Paciorek and Schervish (2004) to propose a nonstationary covariance function in single output regression. It has also been used in Boyle and Frean (2005) as an example of constructing dependent Gaussian processes.

The kernel smoothing function is given as

$$G_{d,q}(\mathbf{x}) = S_{d,q}\mathcal{N}(\mathbf{x}|\mathbf{0},\mathbf{P}_d^{-1}),$$

5. We have slightly abused of the delta notation to indicate the Kronecker delta for discrete arguments and the Dirac function for continuous arguments. The particular meaning should be understood from the context.

where $S_{d,q}$ is a variance coefficient that depends both on the output d and the latent function q and $\mathbf{P}_d$ is the precision matrix associated to the particular output d. The covariance function for the latent process is expressed as

$$k_q(\mathbf{x},\mathbf{x}') = \mathcal{N}(\mathbf{x}-\mathbf{x}'|\mathbf{0},\boldsymbol{\Lambda}_q^{-1}),$$

with $\boldsymbol{\Lambda}_q$ the precision matrix of the latent function q.

Expressions for the kernels are obtained applying systematically the identity for the product of two Gaussian distributions. Let $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\mathbf{P}^{-1})$ denote a Gaussian for $\mathbf{x}$, then

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1,\mathbf{P}_1^{-1})\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_2,\mathbf{P}_2^{-1}) = \mathcal{N}(\boldsymbol{\mu}_1|\boldsymbol{\mu}_2,\mathbf{P}_1^{-1}+\mathbf{P}_2^{-1})\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c,\mathbf{P}_c^{-1}), \tag{11}$$

where $\boldsymbol{\mu}_c = (\mathbf{P}_1+\mathbf{P}_2)^{-1}(\mathbf{P}_1\boldsymbol{\mu}_1+\mathbf{P}_2\boldsymbol{\mu}_2)$ and $\mathbf{P}_c^{-1} = (\mathbf{P}_1+\mathbf{P}_2)^{-1}$. For all integrals we assume that $\mathcal{X} = \Re^p$. Using these forms for $G_{d,q}(\mathbf{x})$ and $k_q(\mathbf{x},\mathbf{x}')$, expression (8) (with $R_q = 1$) can be written as

$$k_{f_d,f_{d'}}(\mathbf{x},\mathbf{x}') = \sum_{q=1}^{Q} S_{d,q}S_{d',q}\int_{\mathcal{X}}\mathcal{N}(\mathbf{x}-\mathbf{z}|\mathbf{0},\mathbf{P}_d^{-1})\int_{\mathcal{X}}\mathcal{N}(\mathbf{x}'-\mathbf{z}'|\mathbf{0},\mathbf{P}_{d'}^{-1})\mathcal{N}(\mathbf{z}-\mathbf{z}'|\mathbf{0},\boldsymbol{\Lambda}_q^{-1})\mathrm{d}\mathbf{z}'\mathrm{d}\mathbf{z}.$$

Since the Gaussian covariance is stationary, we can write it as $\mathcal{N}(\mathbf{x}-\mathbf{x}'|\mathbf{0},\mathbf{P}^{-1}) = \mathcal{N}(\mathbf{x}'-\mathbf{x}|\mathbf{0},\mathbf{P}^{-1}) = \mathcal{N}(\mathbf{x}|\mathbf{x}',\mathbf{P}^{-1}) = \mathcal{N}(\mathbf{x}'|\mathbf{x},\mathbf{P}^{-1})$. Using the identity in Equation (11) twice, we get

$$k_{f_d,f_{d'}}(\mathbf{x},\mathbf{x}') = \sum_{q=1}^{Q} S_{d,q}S_{d',q}\mathcal{N}(\mathbf{x}-\mathbf{x}'|\mathbf{0},\mathbf{P}_d^{-1}+\mathbf{P}_{d'}^{-1}+\boldsymbol{\Lambda}_q^{-1}). \tag{12}$$

For a high value of the input dimension, p, the term $1/[(2\pi)^{p/2}|\mathbf{P}_d^{-1}+\mathbf{P}_{d'}^{-1}+\boldsymbol{\Lambda}_q^{-1}|^{1/2}]$ in each of the Gaussian's normalization terms will dominate, making values go quickly to zero. We can fix this problem, by scaling the outputs using the factors $1/[(2\pi)^{p/4}|2\mathbf{P}_d^{-1}+\boldsymbol{\Lambda}_q^{-1}|^{1/4}]$ and $1/[(2\pi)^{p/4}|2\mathbf{P}_{d'}^{-1}+\boldsymbol{\Lambda}_q^{-1}|^{1/4}]$. Each of these scaling factors correspond to the standard deviation associated to $k_{f_d,f_d}(\mathbf{x},\mathbf{x})$ and $k_{f_{d'},f_{d'}}(\mathbf{x},\mathbf{x})$.

Equally for the covariance $\operatorname{cov}[f_d(\mathbf{x}),u_q(\mathbf{x}')]$ in Equation (9), we obtain

$$k_{f_d,u_q}(\mathbf{x},\mathbf{x}') = S_{d,q}\mathcal{N}(\mathbf{x}-\mathbf{x}'|\mathbf{0},\mathbf{P}_d^{-1}+\boldsymbol{\Lambda}_q^{-1}).$$

Again, this covariance must be standardized when working in higher dimensions.

4. Hyperparameter Learning

Given the convolution formalism, we can construct a full GP over the set of outputs. The likelihood of the model is given by

$$p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}|\mathbf{0},\mathbf{K}_{\mathbf{f},\mathbf{f}}+\boldsymbol{\Sigma}), \tag{13}$$

where $\mathbf{y} = \left[\mathbf{y}_1^\top,\ldots,\mathbf{y}_D^\top\right]^\top$ is the set of output functions with $\mathbf{y}_d = [y_d(\mathbf{x}_1),\ldots,y_d(\mathbf{x}_N)]^\top$; $\mathbf{K}_{\mathbf{f},\mathbf{f}} \in \Re^{DN\times DN}$ is the covariance matrix arising from the convolution. It expresses the covariance of each data point at every other output and data point and its elements are given by $\operatorname{cov}[f_d(\mathbf{x}),f_{d'}(\mathbf{x}')]$ in (8). The term $\boldsymbol{\Sigma}$ represents the covariance associated with the independent processes in (10), $w_d(\mathbf{x})$. It could contain structure, or alternatively could simply represent noise that is independent across the data points. The vector $\boldsymbol{\theta}$ refers to the hyperparameters of the model. For exposition we will focus on the isotopic case (although our implementations allow heterotopic modeling), so we have a matrix $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ which is the common set of training input vectors at which the covariance is evaluated.

The predictive distribution for a new set of input vectors $\mathbf{X}_*$ is (Rasmussen and Williams, 2006)

$$p(\mathbf{y}_*|\mathbf{y}, \mathbf{X}, \mathbf{X}_*, \boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{y}_*|\mathbf{K}_{\mathbf{f}_*,\mathbf{f}}(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \boldsymbol{\Sigma})^{-1}\mathbf{y}, \mathbf{K}_{\mathbf{f}_*,\mathbf{f}_*} - \mathbf{K}_{\mathbf{f}_*,\mathbf{f}}(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \boldsymbol{\Sigma})^{-1}\mathbf{K}_{\mathbf{f},\mathbf{f}_*} + \boldsymbol{\Sigma}_*\right),$$

where we have used $\mathbf{K}_{\mathbf{f}_*,\mathbf{f}_*}$ as a compact notation to indicate when the covariance matrix is evaluated at the inputs $\mathbf{X}_*$, with a similar notation for $\mathbf{K}_{\mathbf{f}_*,\mathbf{f}}$. Learning from the log-likelihood involves the computation of the inverse of $\mathbf{K}_{\mathbf{f},\mathbf{f}} + \boldsymbol{\Sigma}$ giving the problematic complexity of $O(N^3D^3)$. Once the parameters have been learned, prediction is $O(ND)$ for the predictive mean and $O(N^2D^2)$ for the predictive variance.

As we have mentioned before, the main focus of this paper is to present some efficient approximations for the multiple output convolved Gaussian Process. Given the methods presented before, we now show an application that benefits from the non-instantaneous mixing element brought by the convolution process framework.

Comparison between instantaneous mixing and non-instantaneous mixing for regression in genes expression data. Microarray studies have made the simultaneous measurement of mRNA from thousands of genes practical. Transcription is governed by the presence or absence of transcription factor (TF) proteins that act as switches to turn on and off the expression of the genes. Most of these methods are based on assuming that there is an instantaneous linear relationship between the gene expression and the protein concentration. We compare the performance of the intrinsic coregionalization model (Section 2.1.1) and the convolved GPs for two independent time series or replicas of 12 time points collected hourly throughout Drosophila embryogenesis in wild-type embryos (Tomancak et al., 2002). For preprocessing the data, we follow Honkela et al. (2010). We concentrate on a particular transcription factor protein, namely `twi`, and the genes associated with it. The information about the network connections is obtained from the ChIP-chip experiments. This particular TF is key regulator of mesoderm and muscle development in Drosophila (Zinzen et al., 2009).

After preprocessing the data, we end up with a data set of 1621 genes with expression data for $N = 12$ time points. It is believed that this set of genes are regulated by at least the `twi` transcription factor. For each one of these genes, we have access to 2 replicas. We randomly select $D = 50$ genes from replica 1 for training a full multiple output GP model based on either the LMC framework or the convolved GP framework. The corresponding 50 genes of replica 2 are used for testing and results are presented in terms of the standardized mean square error (SMSE) and the mean standardized log loss (MSLL) as defined in Rasmussen and Williams (2006).[6] The parameters of both the LMC and the convolved GPs are found through the maximization of the marginal likelihood in Equation (13). We repeated the experiment 10 times using a different set of 50 genes each time. We also repeated the experiment selecting the 50 genes for training from replica 2 and the corresponding 50 genes of replica 1 for testing.

6. The definitions for the SMSE and the MSLL we have used here are slightly different from the ones provided in Rasmussen and Williams (2006). Instead of comparing against a Gaussian with a global mean and variance computed from all the outputs in the training data, we compare against a Gaussian with local means and local variances computed from the training data associated to each output.

We are interested in a reduced representation of the data so we assume that $Q = 1$ and $R_q = 1$, for the LMC and the convolved multiple output GP in Equations (2) and (8), respectively. For the LMC model, we follow Bonilla et al. (2008) and assume an incomplete Cholesky decomposition for $\mathbf{B}_1 = \widetilde{L}\widetilde{L}^\top$, where $\widetilde{L} \in \Re^{50\times1}$ and as the basic covariance $k_q(\mathbf{x},\mathbf{x}')$ we assume the squared exponential covariance function (p. 83, Rasmussen and Williams, 2006). For the convolved multiple output GP we employ the covariance described in Section 3, Equation (12), with the appropriate scaling factors.

Train set	Test set	Method	Average SMSE	Average MSLL
Replica 1	Replica 2	LMC	0.6069 ± 0.0294	-0.2687 ± 0.0594
		CMOC	0.4859 ± 0.0387	-0.3617 ± 0.0511
Replica 2	Replica 1	LMC	0.6194 ± 0.0447	-0.2360 ± 0.0696
		CMOC	0.4615 ± 0.0626	-0.3811 ± 0.0748

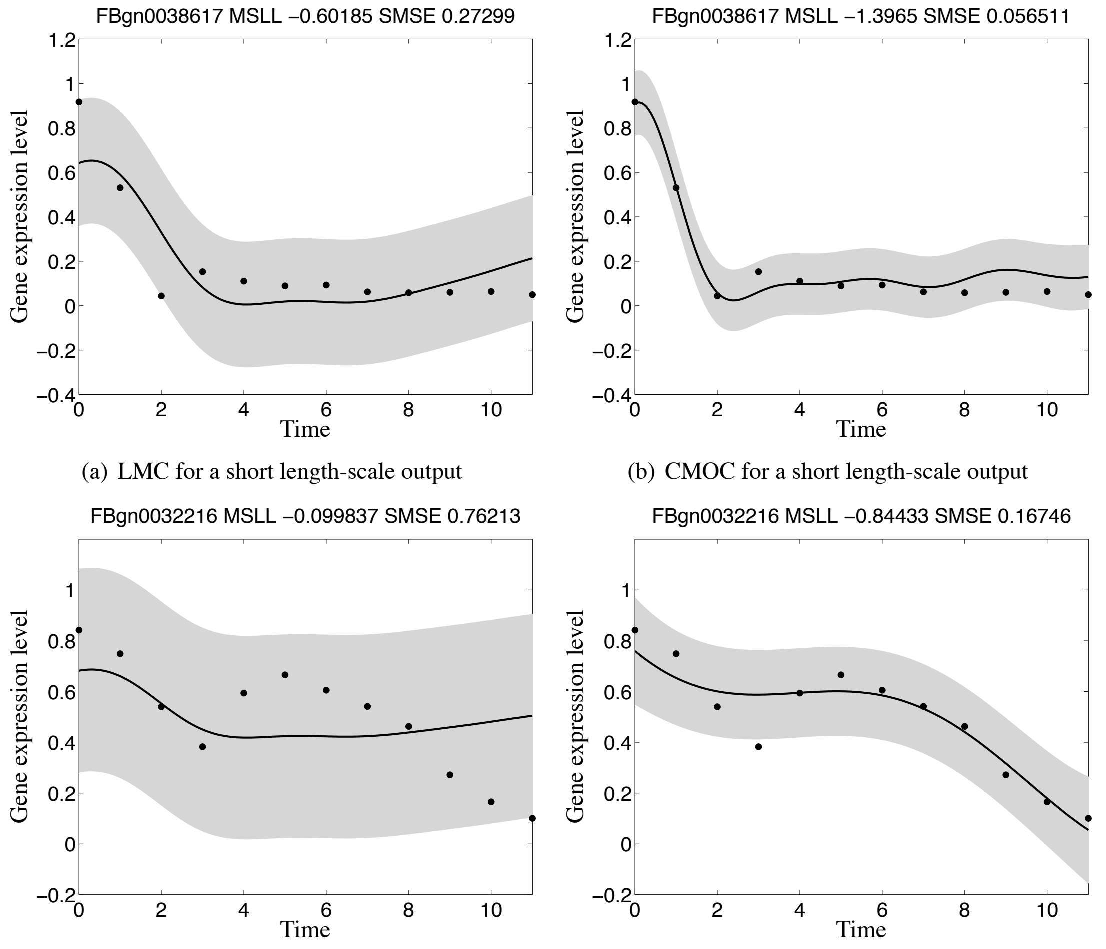
Table 1: Standardized mean square error (SMSE) and mean standardized log loss (MSLL) for the gene expression data for 50 outputs. CMOC stands for convolved multiple output covariance. The experiment was repeated ten times with a different set of 50 genes each time. Table includes the value of one standard deviation over the ten repetitions. More negative values of MSLL indicate better models.

Table 1 shows the results of both methods over the test set for the two different replicas. It can be seen that the convolved multiple output covariance (appearing as CMOC in the table), outperforms the LMC covariance both in terms of SMSE and MSLL.

Figure 1 shows the prediction made over the test set (replica 2 in this case) by the two models for two particular genes, namely FBgn0038617 (Figure 1, first row) and FBgn0032216 (Figure 1, second row). The black dots in the figures represent the gene expression data of the particular genes. Figures 1(a) and 1(c) show the response of the LMC and Figures 1(b) and 1(d) show the response of the convolved multiple output covariance. It can be noticed from the data that the two genes differ in their responses to the action of the transcription factor, that is, while gene FBgn0038617 has a rapid decay around time 2 and becomes relatively constant for the rest of the time interval, gene FBgn0032216 has a smoother response within the time frame. The linear model of coregionalization is driven by a latent function with a length-scale that is shared across the outputs. Notice from Figures 1(a) and 1(c) that the length-scale for both responses is the same. On the other hand, due-to the non-instantaneous mixing of the latent function, the convolved multiple output framework, allows the description of each output using its own length-scale, which gives an added flexibility for describing the data.

Table 2 (first four rows) shows the performances of both models for the genes of Figure 1. CMOC outperforms the linear model of coregionalization for both genes in terms of SMSE and MSLL.

A similar analysis can be made for Figures 2(a), 2(b), 2(c) and 2(d). In this case, the test set is replica 1 and we have chosen two different genes, FBgn0010531 and FBgn0004907 with a similar behavior. Table 2 (last four rows) also highlights the performances of both models for the genes of Figure 2. Again, CMOC outperforms the linear model of coregionalization for both genes and in terms of SMSE and MSLL.



(a) LMC for a short length-scale output (b) CMOC for a short length-scale output

(c) LMC for a long length-scale output (d) CMOC for a long length-scale output

Figure 1: Predictive mean and variance for genes FBgn0038617 (first row) and FBgn0032216 (second row) using the linear model of coregionalization in Figures 1(a) and 1(c) and the convolved multiple-output covariance in Figures 1(b) and 1(d), with $Q = 1$ and $R_q = 1$. The training data comes from replica 1 and the testing data from replica 2. The solid line corresponds to the predictive mean, the shaded region corresponds to 2 standard deviations of the prediction. Performances in terms of SMSE and MSLL are given in the title of each figure and appear also in Table 2. The adjectives “short” and “long” given to the length-scales in the captions of each figure, must be understood like relative to each other.

Having said this, we can argue that the performance of the LMC model can be improved by either increasing the value of Q or the value R_q, or both. For the intrinsic coregionalization model, we would fix the value of $Q = 1$ and increase the value of R_1. Effectively, we would be increasing the rank of the coregionalization matrix $\mathbf{B}_1$, meaning that more latent functions sampled from the same covariance function are being used to explain the data. In a extreme case in which each output has its own length scale, this translates into equating the number of latent functions to the number

Test replica	Test genes	Method	SMSE	MSLL
Replica 2	FBgn0038617	LMC	0.2729	-0.6018
		CMOC	0.0565	-1.3965
	FBgn0032216	LMC	0.7621	-0.0998
		CMOC	0.1674	-0.8443
Replica 1	FBgn0010531	LMC	0.2572	-0.5699
		CMOC	0.0446	-1.3434
	FBgn0004907	LMC	0.4984	-0.3069
		CMOC	0.0971	-1.0841

Table 2: Standardized mean square error (SMSE) and mean standardized log loss (MSLL) for the genes in Figures 1 and 2 for LMC and CMOC. Genes FBgn0038617 and FBgn0010531 have a shorter length-scale when compared to genes FBgn0032216 and FBgn0004907.

of outputs, or in other words assuming a full rank for the matrix $\mathbf{B}_1$. This leads to the need of estimating the matrix $\mathbf{B}_1 \in \Re^{D\times D}$, that might be problematic if D is high. For the semiparametric latent factor model, we would fix the value of $R_q = 1$ and increase Q, the number of latent functions sampled from Q different GPs. Again, in the extreme case of each output having its own length-scale, we might need to estimate a matrix $\widetilde{\mathbf{A}} \in \Re^{D\times D}$, which could be problematic for a high value of outputs. In a more general case, we could also combine values of $Q > 1$ and $R_q > 1$. We would need then, to find values of Q and R_q that fit the different outputs with different length scales.

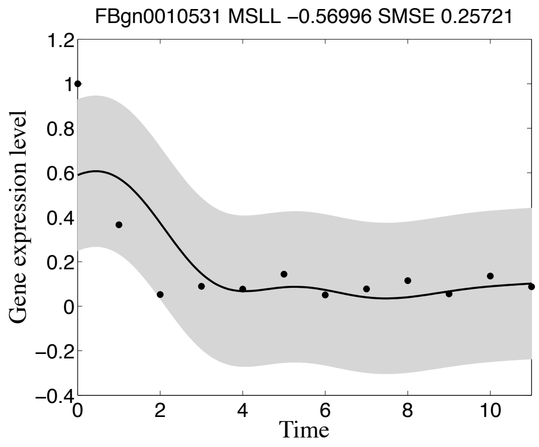
In practice though, we will see in the experimental section, that both the linear model of coregionalization and the convolved multiple output GPs can perform equally well in some data sets. However, the convolved covariance could offer an explanation of the data through a simpler model or converge to the LMC, if needed.

5. Efficient Approximations for Convolutional Processes

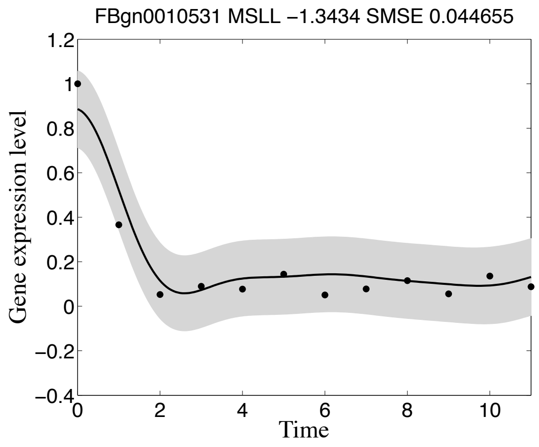
Assuming that the double integral in Equation (8) is tractable, the principle challenge for the convolutional framework is computing the inverse of the covariance matrix associated with the outputs. For D outputs, each having N data points, the inverse has computational complexity $O(D^3N^3)$ and associated storage of $O(D^2N^2)$. We show how through making specific conditional independence assumptions, inspired by the model structure (Álvarez and Lawrence, 2009), we arrive at a efficient approximation similar in form to the partially independent training conditional model (PITC, see Quiñonero-Candela and Rasmussen, 2005). The relationship with PITC then inspires us to make further conditional independence assumptions.

5.1 Latent Functions as Conditional Means

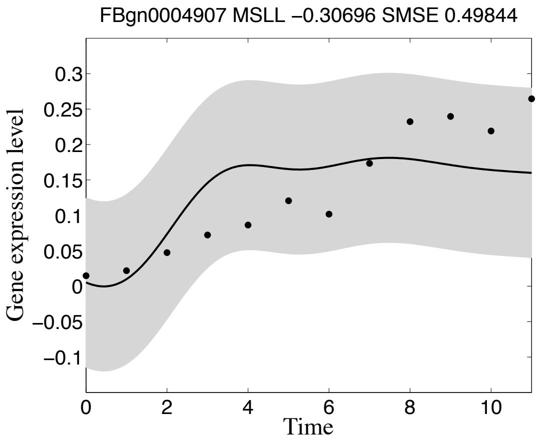
For notational simplicity, we restrict the analysis of the approximations to one latent function $u(\mathbf{x})$. The key to all approximations is based on the form we assume for the latent functions. From the perspective of a generative model, Equation (7) can be interpreted as follows: first we draw a sample from the Gaussian process prior $p(u(\mathbf{z}))$ and then solve the integral for each of the outputs $f_d(\mathbf{x})$ involved. Uncertainty about $u(\mathbf{z})$ is also propagated through the convolution transform.



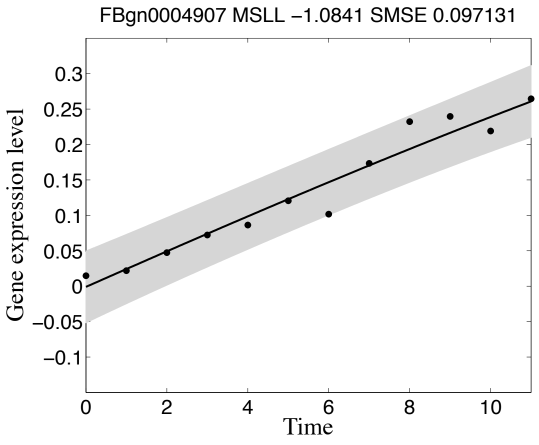
(a) LMC for a short length-scale output



(b) CMOC for a short length-scale output



(c) LMC for a long length-scale output



(d) CMOC for a long length-scale output

Figure 2: Predictive mean and variance for genes FBgn0010531 (first row) and FBgn0004907 (second row) using the linear model of coregionalization in Figures 2(a) and 2(c), and the convolved multiple-output covariance in Figures 2(b) and 2(d), with $Q = 1$ and $R_q = 1$. The difference with Figure 1 is that now the training data comes from replica 2 while the testing data comes from replica 1. The solid line corresponds to the predictive mean, the shaded region corresponds to 2 standard deviations of the prediction. Performances in terms of SMSE and MSLL are given in the title of each figure.

For the set of approximations, instead of drawing a sample from $u(\mathbf{z})$, we first draw a sample from a finite representation of $u(\mathbf{z})$, $\mathbf{u}(\mathbf{Z}) = [u(\mathbf{z}_1), \ldots, u(\mathbf{z}_K)]^\top$, where $\mathbf{Z} = \{\mathbf{z}_k\}_{k=1}^K$ is the set of input vectors at which $u(\mathbf{z})$ is evaluated. Due to the properties of a Gaussian process, $p(\mathbf{u}(\mathbf{Z}))$ follows a multivariate Gaussian distribution. Conditioning on $\mathbf{u}(\mathbf{Z})$, we next sample from the conditional prior $p(u(\mathbf{z})|\mathbf{u}(\mathbf{Z}))$ and use this function to solve the convolution integral for each $f_d(\mathbf{x})$.[7] Under

7. For simplicity in the notation, we just write $\mathbf{u}$ to refer to $\mathbf{u}(\mathbf{Z})$.

this generative approach, we can approximate each function $f_d(\mathbf{x})$ using

$$f_d(\mathbf{x}) \approx \int_{\mathcal{X}} G_d(\mathbf{x}-\mathbf{z})\,\mathrm{E}\left[u(\mathbf{z})|\mathbf{u}\right]\mathrm{d}\mathbf{z}. \tag{14}$$

Replacing $u(\mathbf{z})$ for $\mathrm{E}\left[u(\mathbf{z})|\mathbf{u}\right]$ is a reasonable approximation as long as $u(\mathbf{z})$ is a smooth function so that the infinite dimensional object $u(\mathbf{z})$ can be summarized by $\mathbf{u}$. Figure 3 shows a cartoon example of the quality of the approximations for two outputs as the size of the set $\mathbf{Z}$ increases. The first column represents the conditional prior $p\left(u(\mathbf{z})|\mathbf{u}\right)$ for a particular choice of $u(\mathbf{z})$. The second and third columns represent the outputs $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ obtained when using Equation (14).

Using expression (14), the likelihood function for $\mathbf{f}$ follows

$$p(\mathbf{f}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{f}|\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u},\mathbf{K}_{\mathbf{f},\mathbf{f}}-\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top}\right), \tag{15}$$

where $\mathbf{K}_{\mathbf{u},\mathbf{u}}$ is the covariance matrix between the samples from the latent function $\mathbf{u}(\mathbf{Z})$, with elements given by $k_{u,u}(\mathbf{z},\mathbf{z}')$ and $\mathbf{K}_{\mathbf{f},\mathbf{u}} = \mathbf{K}_{\mathbf{u},\mathbf{f}}^{\top}$ is the cross-covariance matrix between the latent function $u(\mathbf{z})$ and the outputs $f_d(\mathbf{x})$, with elements $\operatorname{cov}\left[f_d(\mathbf{x}),u(\mathbf{z})\right]$ in (9).

Given the set of points $\mathbf{u}$, we can have different assumptions about the uncertainty of the outputs in the likelihood term. For example, we could assume that the outputs are independent or uncorrelated, keeping only the uncertainty involved for each output in the likelihood term. Another approximation assumes that the outputs are deterministic, this is $\mathbf{K}_{\mathbf{f},\mathbf{f}} = \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top}$. The only uncertainty left would be due to the prior $p(\mathbf{u})$. Next, we present different approximations of the covariance of the likelihood that lead to a reduction in computational complexity.

5.1.1 PARTIAL INDEPENDENCE

We assume that the individual outputs in $\mathbf{f}$ are independent given the latent function $\mathbf{u}$, leading to the following expression for the likelihood

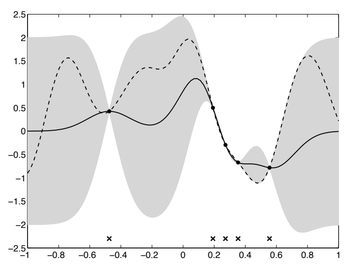
$$p(\mathbf{f}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \prod_{d=1}^{D} p(\mathbf{f}_d|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \prod_{d=1}^{D}\mathcal{N}\left(\mathbf{f}|\mathbf{K}_{\mathbf{f}_d,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u},\mathbf{K}_{\mathbf{f}_d,\mathbf{f}_d}-\mathbf{K}_{\mathbf{f}_d,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}_d}\right).$$

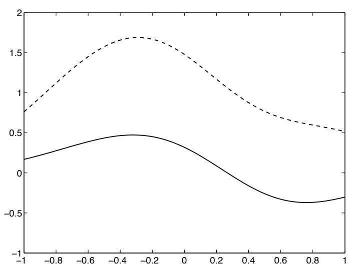
We rewrite this product of multivariate Gaussians as a single Gaussian with a block diagonal covariance matrix, including the uncertainty about the independent processes

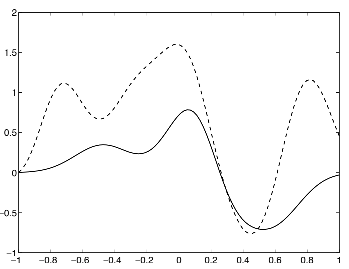
$$p(\mathbf{y}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u},\mathbf{D}+\boldsymbol{\Sigma}\right) \tag{16}$$

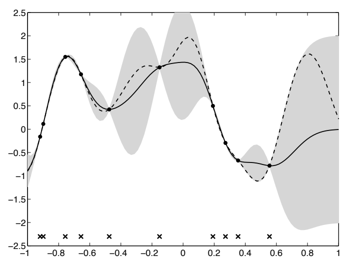
where $\mathbf{D} = \operatorname{blockdiag}\left[\mathbf{K}_{\mathbf{f},\mathbf{f}}-\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top}\right]$, and we have used the notation $\operatorname{blockdiag}\left[\mathbf{G}\right]$ to indicate that the block associated with each output of the matrix $\mathbf{G}$ should be retained, but all other elements should be set to zero. We can also write this as $\mathbf{D} = \left[\mathbf{K}_{\mathbf{f},\mathbf{f}}-\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right]\odot\mathbf{M}$ where $\odot$ is the Hadamard product and $\mathbf{M} = \mathbf{I}_D\otimes\mathbf{1}_N$, $\mathbf{1}_N$ being the $N\times N$ matrix of ones. We now marginalize the values of the samples from the latent function by using its process prior, this means $p(\mathbf{u}|\mathbf{Z}) = \mathcal{N}(\mathbf{u}|\mathbf{0},\mathbf{K}_{\mathbf{u},\mathbf{u}})$. This leads to the following marginal likelihood,

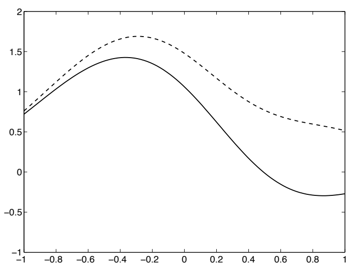
$$p(\mathbf{y}|\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta})p(\mathbf{u}|\mathbf{Z})\mathrm{d}\mathbf{u} = \mathcal{N}\left(\mathbf{y}|\mathbf{0},\mathbf{D}+\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}+\boldsymbol{\Sigma}\right). \tag{17}$$

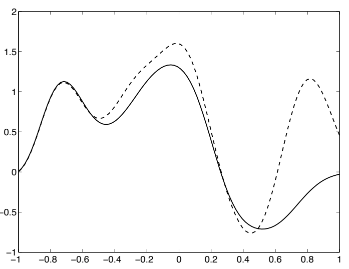

(a) Conditional prior for $K = 5$

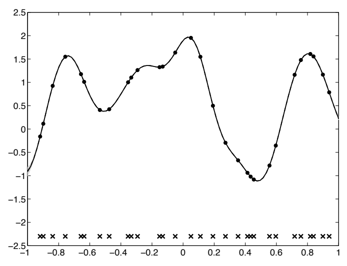

(b) Output one for $K = 5$

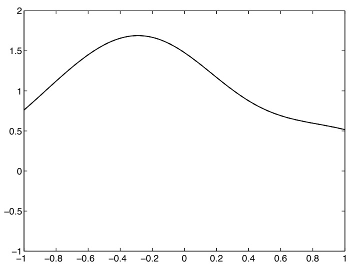

(c) Output two for $K = 5$


(d) Conditional prior for $K = 10$


(e) Output one for $K = 10$


(f) Output two for $K = 10$


(g) Conditional prior for $K = 30$


(h) Output one for $K = 30$

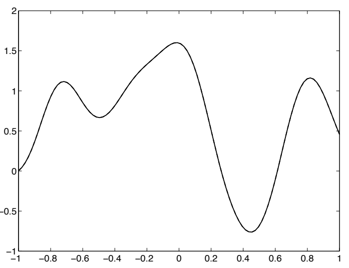

(i) Output two for $K = 30$

Figure 3: Conditional prior and two outputs for different values of K. The first column, Figures 3(a), 3(d) and 3(g), shows the mean and confidence intervals of the conditional prior distribution using one input function and two output functions. The dashed line represents one sample from the prior. Conditioning over a few points of this sample, shown as black dots, the conditional mean and conditional covariance are computed. The solid line represents the conditional mean and the shaded region corresponds to 2 standard deviations away from the mean. The second column, 3(b), 3(e) and 3(h), shows the solution to Equation (7) for output one using the sample from the prior (dashed line) and the conditional mean (solid line), for different values of K. The third column, 3(c), 3(f) and 3(i), shows the solution to Equation (7) for output two, again for different values of K.

Notice that, compared to (13), the full covariance matrix $\mathbf{K}_{\mathbf{f},\mathbf{f}}$ has been replaced by the low rank covariance $\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}$ in all entries except in the diagonal blocks corresponding to $\mathbf{K}_{\mathbf{f}_d,\mathbf{f}_d}$. Depending on our choice of K, the inverse of the low rank approximation to the covariance is either dominated by a $O(DN^3)$ term or a $O(K^2DN)$ term. Storage of the matrix is $O(N^2D)+O(NDK)$.

Note that if we set $K = N$ these reduce to $O(N^3D)$ and $O(N^2D)$ respectively. Rather neatly this matches the computational complexity of modeling the data with D independent Gaussian processes across the outputs.

The functional form of (17) is almost identical to that of the partially independent training conditional (PITC) approximation (Quiñonero-Candela and Rasmussen, 2005) or the partially independent conditional (PIC) approximation (Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2007), with the samples we retain from the latent function providing the same role as the *inducing values* in the PITC or PIC.[8] This is perhaps not surprising given that the PI(T)C approximations are also derived by making conditional independence assumptions. A key difference is that in PI(T)C it is not obvious which variables should be grouped together when making these conditional independence assumptions; here it is clear from the structure of the model that each of the outputs should be grouped separately.

5.1.2 FULL INDEPENDENCE

We can be inspired by the analogy of our approach to the PI(T)C approximation and consider a more radical factorization of the likelihood term. In the fully independent training conditional (FITC) approximation or the fully independent conditional (FIC) approximation (Snelson and Ghahramani, 2006, 2007), a factorization across the data points is assumed. For us that would lead to the following expression for the conditional distribution of the output functions given the inducing variables,

$$p(\mathbf{f}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \prod_{d=1}^{D}\prod_{n=1}^{N} p(f_{n,d}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}),$$

which can be expressed through (16) with $\mathbf{D} = \operatorname{diag}\left[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top}\right] = \left[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top}\right] \odot \mathbf{M}$, with $\mathbf{M} = \mathbf{I}_D \otimes \mathbf{I}_N$ or simply $\mathbf{M} = \mathbf{I}_{DN}$. The marginal likelihood, including the uncertainty about the independent processes, is given by Equation (17) with the diagonal form for $\mathbf{D}$. Training with this approximated likelihood reduces computational complexity to $O(K^2DN)$ and the associated storage to $O(KDN)$.

5.1.3 DETERMINISTIC LIKELIHOOD

In Quiñonero-Candela and Rasmussen (2005), the relationship between the projected process approximation (Csató and Opper, 2001; Seeger et al., 2003) and the FI(T)C and PI(T)C approximations is elucidated. They show that if, given the set of values $\mathbf{u}$, the outputs are assumed to be deterministic, the likelihood term of Equation (15) can be simplified as

$$p(\mathbf{f}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{f}|\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u},\mathbf{0}\right).$$

Marginalizing with respect to the latent function using $p(\mathbf{u}|\mathbf{Z}) = \mathcal{N}(\mathbf{u}|\mathbf{0},\mathbf{K}_{\mathbf{u},\mathbf{u}})$ and including the uncertainty about the independent processes, we obtain the marginal likelihood as

$$p(\mathbf{y}|\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta})p(\mathbf{u}|\mathbf{Z})\mathrm{d}\mathbf{u} = \mathcal{N}\left(\mathbf{y}|\mathbf{0},\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top} + \boldsymbol{\Sigma}\right).$$

8. We refer to both PITC and PIC by PI(T)C.

In other words, we can approximate the full covariance $\mathbf{K}_{\mathbf{f},\mathbf{f}}$ using the low rank approximation $\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top}$. Using this new marginal likelihood to estimate the parameters $\boldsymbol{\theta}$ reduces computational complexity to $O(K^2DN)$. The approximation obtained has similarities with the projected latent variables (PLV) method also known as the projected process approximation (PPA) or the deterministic training conditional (DTC) approximation (Csató and Opper, 2001; Seeger et al., 2003; Quiñonero-Candela and Rasmussen, 2005; Rasmussen and Williams, 2006).

5.1.4 Additional Independence Assumptions

As mentioned before, we can consider different conditional independence assumptions for the likelihood term. One further assumption that is worth mentioning considers conditional independencies across data points and dependence across outputs. This would lead to the following likelihood term

$$p(\mathbf{f}|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\bar{\mathbf{f}}_n|\mathbf{u},\mathbf{Z},\mathbf{X},\boldsymbol{\theta}),$$

where $\bar{\mathbf{f}}_n = [f_1(\mathbf{x}_n), f_2(\mathbf{x}_n), \dots, f_D(\mathbf{x}_n)]^{\top}$. We can use again Equation (16) to express the likelihood. In this case, though, the matrix $\mathbf{D}$ is a partitioned matrix with blocks $\mathbf{D}_{d,d'} \in \Re^{N\times N}$ and each block $\mathbf{D}_{d,d'}$ would be given as $\mathbf{D}_{d,d'} = \operatorname{diag}\left[\mathbf{K}_{\mathbf{f}_d,\mathbf{f}_{d'}} - \mathbf{K}_{\mathbf{f}_d,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}_{d'}}\right]$. For cases in which $D > N$, that is, the number of outputs is greater than the number of data points, this approximation may be more accurate than the one obtained with the partial independence assumption. For cases where $D < N$ it may be less accurate, but faster to compute.[9]

5.2 Posterior and Predictive Distributions

Combining the likelihood term for each approximation with $p(\mathbf{u}|\mathbf{Z})$ using Bayes' theorem, the posterior distribution over $\mathbf{u}$ is obtained as

$$p(\mathbf{u}|\mathbf{y},\mathbf{X},\mathbf{Z},\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{u}|\mathbf{K}_{\mathbf{u},\mathbf{u}}\mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}(\mathbf{D}+\boldsymbol{\Sigma})^{-1}\mathbf{y}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{u}}\right), \tag{18}$$

where $\mathbf{A} = \mathbf{K}_{\mathbf{u},\mathbf{u}} + \mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top}(\mathbf{D}+\boldsymbol{\Sigma})^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}$ and $\mathbf{D}$ follows a particular form according to the different approximations: for partial independence it equals $\mathbf{D} = \operatorname{blockdiag}\left[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right]$; for full independence it is $\mathbf{D} = \operatorname{diag}\left[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right]$ and for the deterministic likelihood, $\mathbf{D} = \mathbf{0}$.

For computing the predictive distribution we have two options, either use the posterior for $\mathbf{u}$ and the approximated likelihoods or the posterior for $\mathbf{u}$ and the likelihood of Equation (15), that corresponds to the likelihood of the model without any approximations. The difference between both options is reflected in the covariance for the predictive distribution. Quiñonero-Candela and Rasmussen (2005) proposed a taxonomy of different approximations according to the type of likelihood used for the predictive distribution, in the context of single output Gaussian processes.

In this paper, we opt for the posterior for $\mathbf{u}$ and the likelihood of the model without any approximations. If we choose the exact likelihood term in Equation (15) (including the noise term), the

9. Notice that if we work with the block diagonal matrices $\mathbf{D}_{d,d'}$, we would need to invert the full matrix $\mathbf{D}$. However, since the blocks $\mathbf{D}_{d,d'}$ are diagonal matrices themselves, the inversion can be done efficiently using, for example, a block Cholesky decomposition. Furthermore, we would be restricted to work with isotopic input spaces. Alternatively, we could rearrange the elements of the matrix $\mathbf{D}$ so that the blocks of the main diagonal are the covariances associated with the vectors $\bar{\mathbf{f}}_n$.

predictive distribution is expressed through the integration of the likelihood term evaluated at $\mathbf{X}_*$, with (18), giving

$$p(\mathbf{y}_*|\mathbf{y},\mathbf{X},\mathbf{X}_*,\mathbf{Z},\boldsymbol{\theta}) = \int p(\mathbf{y}_*|\mathbf{u},\mathbf{Z},\mathbf{X}_*,\boldsymbol{\theta})p(\mathbf{u}|\mathbf{y},\mathbf{X},\mathbf{Z},\boldsymbol{\theta})\mathrm{d}\mathbf{u} = \mathcal{N}\left(\mathbf{y}_*|\boldsymbol{\mu}_{\mathbf{y}_*},\mathbf{K}_{\mathbf{y}_*,\mathbf{y}_*}\right),$$

where

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{y}_*} &= \mathbf{K}_{\mathbf{f}_*,\mathbf{u}}\mathbf{A}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}^{\top}(\mathbf{D}+\boldsymbol{\Sigma})^{-1}\mathbf{y},\\ \mathbf{K}_{\mathbf{y}_*,\mathbf{y}_*} &= \mathbf{K}_{\mathbf{f}_*,\mathbf{f}_*} - \mathbf{K}_{\mathbf{f}_*,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{f}_*,\mathbf{u}}^{\top} + \mathbf{K}_{\mathbf{f}_*,\mathbf{u}}\mathbf{A}^{-1}\mathbf{K}_{\mathbf{f}_*,\mathbf{u}}^{\top} + \boldsymbol{\Sigma}_*.\end{aligned}$$

For the single output case, the assumption of the deterministic likelihood is equivalent to the deterministic training conditional (DTC) approximation, the full independence approximation leads to the fully independent training conditional (FITC) approximation (Quiñonero-Candela and Rasmussen, 2005) and the partial independence leads to the partially independent training conditional (PITC) approximation (Quiñonero-Candela and Rasmussen, 2005). The similarities of our approximations for multioutput GPs with respect to approximations presented in Quiñonero-Candela and Rasmussen (2005) for single output GPs are such, that we find it convenient to follow the same terminology and also refer to our approximations as DTC, FITC and PITC approximations for multioutput Gaussian processes.

5.3 Discussion: Model Selection in Approximated Models

The marginal likelihood approximation for the PITC, FITC and DTC variants is a function of both the hyperparameters of the covariance function and the location of the inducing variables. For estimation purposes, there seems to be a consensus in the GP community that hyperparameters for the covariance function can be obtained by maximization of the marginal likelihood. For selecting the inducing variables, though, there are different alternatives that can in principle be used. Simpler methods include fixing the inducing variables to be the same set of input data points or grouping the input data using a clustering method like K-means and then use the K resulting vectors as inducing variables. More sophisticated alternatives consider that the set of inducing variables must be restricted to be a subset of the input data (Csató and Opper, 2001; Williams and Seeger, 2001). This set of methods require a criteria for choosing the optimal subset of the training points (Smola and Bartlett, 2001; Seeger et al., 2003). Such approximations are truly sparse in the sense that only few data points are needed at the end for making predictions. Recently, Snelson and Ghahramani (2006) suggested using the marginal likelihood not only for the optimization of the hyperparameters in the covariance function, but also for the optimization of the location of these inducing variables. Although, using such procedure to find the optimal location of the inducing inputs might look in principle like an overwhelming optimization problem (inducing points usually appear non-linearly in the covariance function), in practice it has been shown that performances close to the full GP model can be obtained in a fraction of the time that it takes to train the full model. In that respect, the inducing points that are finally found are optimal in the same optimality sense that the hyperparameters of the covariance function.

Essentially, it would be possible to use any of the methods just mentioned above together with the multiple-output GP regression models presented in Sections 2.1, 2.1.2 and 3. In this paper, though, we follow Snelson and Ghahramani (2006) and optimize the locations of the inducing variables using the approximated marginal likelihoods and leave the comparison between the different model selection methods for inducing variables for future work.

In appendix A we include the derivatives of the marginal likelihood wrt the matrices $\mathbf{K}_{\mathbf{f},\mathbf{f}}, \mathbf{K}_{\mathbf{u},\mathbf{f}}$ and $\mathbf{K}_{\mathbf{u},\mathbf{u}}$.

6. Experimental Evaluation

In this section we present results of applying the approximations in exam score prediction, pollutant metal prediction and the prediction of gene expression behavior in a gene-network. When possible, we first compare the convolved multiple output GP method against the intrinsic model of coregionalization and the semiparametric latent factor model. Then, we compare the different approximations in terms of accuracy and training times. First, though, we illustrate the performance of the approximation methods in a toy example.[10]

6.1 A Toy Example

For the toy experiment, we employ the kernel constructed as an example in Section 3. The toy problem consists of $D = 4$ outputs, one latent function, $Q = 1$ and $R_q = 1$ and one input dimension. The training data was sampled from the full GP with the following parameters, $S_{1,1} = S_{2,1} = 1$, $S_{3,1} = S_{4,1} = 5$, $P_{1,1} = P_{2,1} = 50$, $P_{3,1} = 300, P_{4,1} = 200$ for the outputs and $\Lambda_1 = 100$ for the latent function. For the independent processes, $w_d(\mathbf{x})$, we simply added white noise separately to each output so we have variances $\sigma_1^2 = \sigma_2^2 = 0.0125$, $\sigma_3^2 = 1.2$ and $\sigma_4^2 = 1$. We generate $N = 500$ observation points for each output and use 200 observation points (per output) for training the full and the approximated multiple output GP and the remaining 300 observation points for testing. We repeated the same experiment setup ten times and compute the standardized mean square error and the mean standardized log loss. For the approximations we use $K = 30$ inducing inputs. We sought the kernel parameters and the positions of the inducing inputs through maximizing the marginal likelihood using a scaled conjugate gradient algorithm. Initially the inducing inputs are equally spaced between the interval $[-1, 1]$.

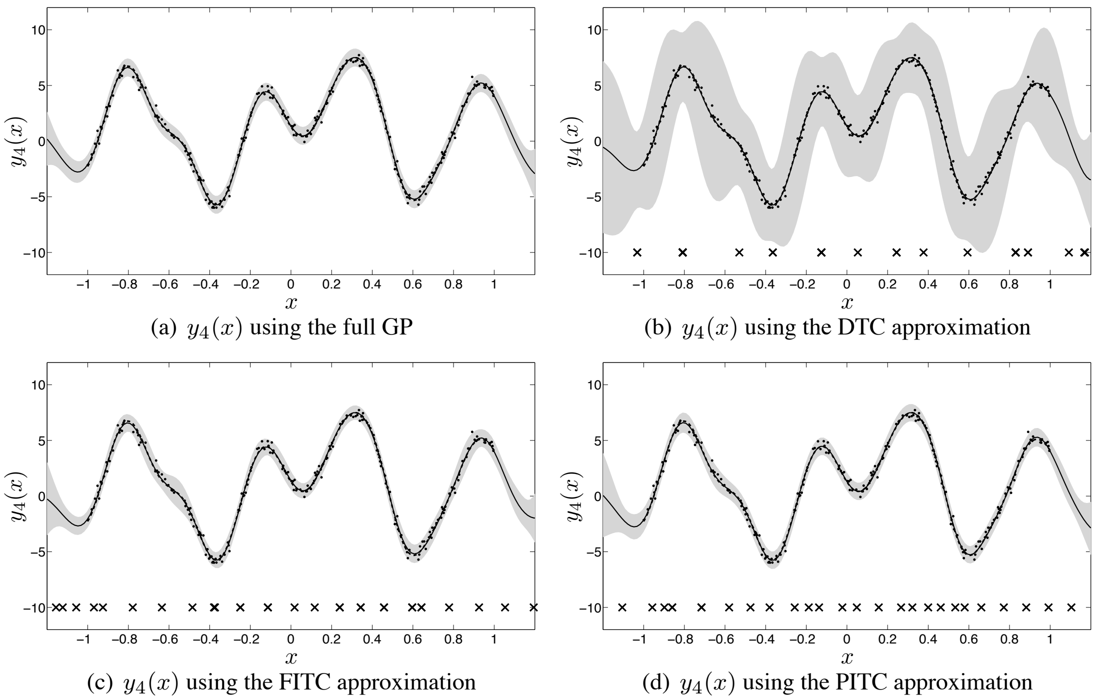
Figure 4 shows the training result of one of the ten repetitions. The predictions shown correspond to the full GP in Figure 4(a), the DTC approximation in Figure 4(b), the FITC approximation in Figure 4(c) and the PITC approximation in Figure 4(d).

Tables 3 and 4 show the average prediction results over the test set. Table 3 shows that the SMSE of the approximations is similar to the one obtained with the full GP. However, there are important differences in the values of the MSLL shown in Table 4. DTC offers the worst performance. It gets better for FITC and PITC since they offer a more precise approximation to the full covariance.

The training times for iteration of each model are 1.97 secs for the full GP, 0.20 secs for DTC, 0.41 for FITC and 0.59 for the PITC, on average.

As we have mentioned before, one important feature of multiple output prediction is that we can exploit correlations between outputs to predict missing observations. We used a simple example to illustrate this point. We removed a portion of one output between $[-0.8, 0]$ from the training data in the experiment before (as shown in Figure 5) and train the different models to predict the behavior of $y_4(x)$ for the missing information. The predictions shown correspond to the full GP in Figure 5(a), an independent GP in Figure 5(b), the DTC approximation in Figure 5(c), the FITC approximation in

10. Code to run all simulations in this section is available at `http://staffwww.dcs.shef.ac.uk/people/N.Lawrence/multigp/`.



(a) $y_4(x)$ using the full GP

(b) $y_4(x)$ using the DTC approximation

(c) $y_4(x)$ using the FITC approximation

(d) $y_4(x)$ using the PITC approximation

Figure 4: Predictive mean and variance using the full multi-output GP and the approximations for output 4. The solid line corresponds to the predictive mean, the shaded region corresponds to 2 standard deviations of the prediction. The dashed line corresponds to the ground truth signal, that is, the sample from the full GP model without noise. In these plots the predictive mean overlaps almost exactly with the ground truth. The dots are the noisy training points. The crosses in Figures 4(b), 4(c) and 4(d) correspond to the locations of the inducing inputs after convergence. Notice that the DTC approximation in Figure 4(b) captures the predictive mean correctly, but fails in reproducing the correct predictive variance.

Method	SMSE $y_1(x)$	SMSE $y_2(x)$	SMSE $y_3(x)$	SMSE $y_4(x)$
Full GP	1.06 ± 0.08	0.99 ± 0.06	1.10 ± 0.09	1.05 ± 0.09
DTC	1.06 ± 0.08	0.99 ± 0.06	1.12 ± 0.09	1.05 ± 0.09
FITC	1.06 ± 0.08	0.99 ± 0.06	1.10 ± 0.08	1.05 ± 0.08
PITC	1.06 ± 0.08	0.99 ± 0.06	1.10 ± 0.09	1.05 ± 0.09

Table 3: Standardized mean square error (SMSE) for the toy problem over the test set. All numbers are to be multiplied by 10^{-2}. The experiment was repeated ten times. Table includes the value of one standard deviation over the ten repetitions.

Figure 5(d) and the PITC approximation in Figure 5(e). The training of the approximation methods is done in the same way than in the experiment before.

Method	MSLL $y_1(x)$	MSLL $y_2(x)$	MSLL $y_3(x)$	MSLL $y_4(x)$
Full GP	-2.27 ± 0.04	-2.30 ± 0.03	-2.25 ± 0.04	-2.27 ± 0.05
DTC	-0.98 ± 0.18	-0.98 ± 0.18	-1.25 ± 0.16	-1.25 ± 0.16
FITC	-2.26 ± 0.04	-2.29 ± 0.03	-2.16 ± 0.04	-2.23 ± 0.05
PITC	-2.27 ± 0.04	-2.30 ± 0.03	-2.23 ± 0.04	-2.26 ± 0.05

Table 4: Mean standardized log loss (MSLL) for the toy problem over the test set. More negative values of MSLL indicate better models. The experiment was repeated ten times. Table includes the value of one standard deviation over the ten repetitions.

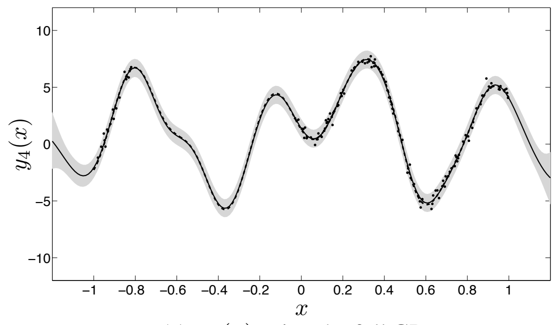
Due to the strong dependencies between the signals, our model is able to capture the correlations and predicts accurately the missing information.

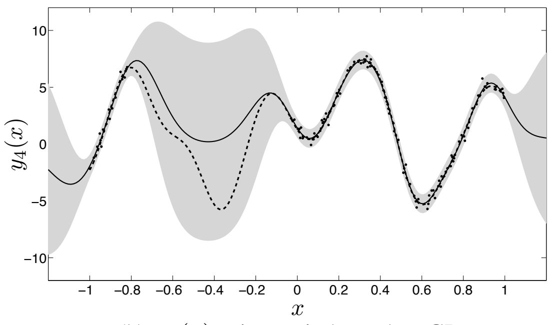
6.2 Exam Score Prediction

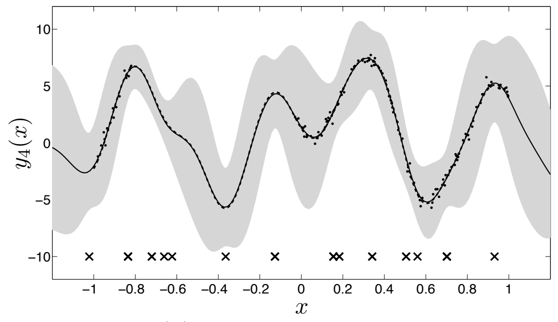
In the first experiment with real data that we consider, the goal is to predict the exam score obtained by a particular student belonging to a particular school. The data comes from the Inner London Education Authority (ILEA).[11] It consists of examination records from 139 secondary schools in years 1985, 1986 and 1987. It is a random 50% sample with 15362 students. The input space consists of four features related to each student (year in which each student took the exam, gender, performance in a verbal reasoning (VR) test[12] and ethnic group) and four features related to each school (percentage of students eligible for free school meals, percentage of students in VR band one, school gender and school denomination). From the multiple output point of view, each school represents one output and the exam score of each student a particular instantiation of that output or $D = 139$.

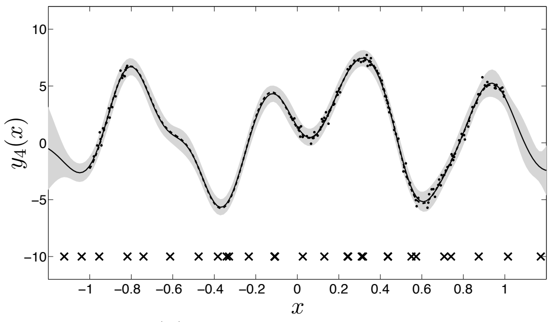
We follow the same preprocessing steps employed in Bonilla et al. (2008). The only features used are the student-dependent ones, which are categorial variables. Each of them is transformed to a binary representation. For example, the possible values that the variable year of the exam can take are 1985, 1986 or 1987 and are represented as 100, 010 or 001. The transformation is also applied to the variables gender (two binary variables), VR band (four binary variables) and ethnic group (eleven binary variables), ending up with an input space with 20 dimensions. The categorial nature of the data restricts the input space to $N = 202$ unique input feature vectors. However, two students represented by the same input vector $\mathbf{x}$, and belonging both to the same school, d, can obtain different exam scores. To reduce this noise in the data, we take the mean of the observations that, within a school, share the same input vector and use a simple heteroskedastic noise model in which the variance for each of these means is divided by the number of observations used to compute it.[13] The performance measure employed is the percentage of explained variance defined as the total variance of the data minus the sum-squared error on the test set as a percentage of the total data variance. It can be seen as the percentage version of the coefficient of determination between the

11. This data is available at `http://www.cmm.bristol.ac.uk/learning-training/multilevel-m-support/datasets.shtml`.
12. Performance in the verbal reasoning test was divided in three bands. Band 1 corresponds to the highest 25%, band 2 corresponds to the next 50% and band 3 the bottom 25% (Nuttall et al., 1989; Goldstein, 1991).
13. Different noise models can be used. However, we employed this one so that we can compare directly to the results presented in Bonilla et al. (2008).


(a) $y_4(x)$ using the full GP


(b) $y_4(x)$ using an independent GP


(c) $y_4(x)$ using the DTC approximation


(d) $y_4(x)$ using the FITC approximation

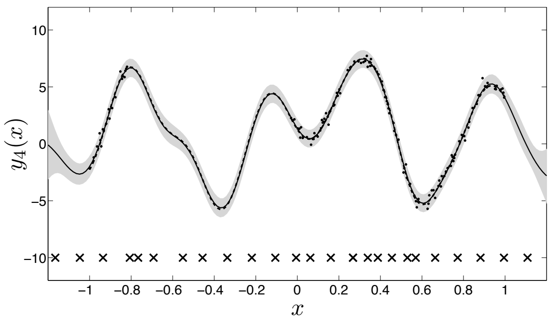

(e) $y_4(x)$ using the PITC approximation

Figure 5: Predictive mean and variance using the full multi-output GP, the approximations and an independent GP for output 4 with a range of missing observations in the interval $[-0.8, 0.0]$. The solid line corresponds to the mean predictive, the shaded region corresponds to 2 standard deviations away from the mean and the dash line is the actual value of the signal without noise. The dots are the noisy training points. The crosses in Figures 5(c), 5(d) and 5(e) correspond to the locations of the inducing inputs after convergence.

test targets and the predictions. The performance measure is computed for ten repetitions with 75% of the data in the training set and 25% of the data in the testing set.

We first compare different methods without including the efficient approximations. These methods are independent GPs, multi-task GPs (Bonilla et al., 2008), the intrinsic coregionalization model, the semiparametric latent factor model and convolved multiple output GPs. Results are

Method	Explained variance (%)
Independent GPs (Bonilla et al., 2008)	31.12 ± 1.33
Multi-task GP (Nyström, $R_1 = 2$) (Bonilla et al., 2008)	36.16 ± 0.99
Intrinsic coregionalization model ($R_1 = 1$)	52.54 ± 2.46
Intrinsic coregionalization model ($R_1 = 2$)	51.94 ± 1.84
Intrinsic coregionalization model ($R_1 = 5$)	45.31 ± 1.63
Semiparametric latent factor model ($Q = 2$)	51.82 ± 1.93
Semiparametric latent factor model ($Q = 5$)	44.87 ± 1.15
Convolved Multiple Outputs GPs ($Q = 1$, $R_q = 1$)	$\mathbf{53.84 \pm 2.01}$

Table 5: Average percentage of explained variance and standard deviation for the exam score prediction on the ILEA data set computed over 10 repetitions. The independent GP result and the multi-task GP result were taken from Bonilla et al. (2008). The value of R_1 in the multi-task GP and in the intrinsic coregionalization model indicates the rank of the matrix $\mathbf{B}_1$ in Equation (6). The value of Q in the semiparametric latent factor model indicates the number of latent functions. The value of R_q in the convolved multiple output GP refers to the number of latent functions that share the same number of parameters (see Equation 8). Refer to the text for more details.

presented in Table 5. The results for the independent GPs and the multi-task GPs were taken from Bonilla et al. (2008). The multi-task GP result uses a matrix $\mathbf{B}_1$ with rank $R_1 = 2$. For the intrinsic model of coregionalization, we use an incomplete Cholesky decomposition $\mathbf{B}_1 = \widetilde{L}\widetilde{L}^\top$, and include results for different values of the rank R_1. The basic covariance $k_q(\mathbf{x}, \mathbf{x}')$ in the ICM is assumed to follow a Gaussian form. For the semiparametric latent factor model, all the latent functions use covariance functions with Gaussian forms. For SLFM, we include results for different values of the number of latent functions ($Q = 2$ and $Q = 5$). Note that SLFM with $Q = 1$ is equivalent to ICM with $R_1 = 1$. For the convolved multiple output covariance result, the kernel employed was introduced in Section 3. For all the models we estimate the parameters maximizing the likelihood through scaled conjugate gradient and run the optimization algorithm for a maximum of 1000 iterations. Table 5 shows that all methods outperform the independent GPs. Even though multi-task GPs with $R_1 = 2$ and ICM with $R_1 = 2$ are equivalent methods, the difference of results might be explained because the multi-task GP method uses a Nyström approximation for the matrix $\mathbf{K}_1$ in Equation (6). Results for ICM with $R_1 = 1$, SLFM with $Q = 2$ and the convolved covariance are similar within the standard deviations. The convolved GP was able to recover the best performance using only one latent function ($Q = 1$). This data set was also employed to evaluate the performance of the multitask kernels in Evgeniou and Pontil (2004). The best result presented in this work was 34.37 ± 0.3. However, due to the averaging of the observations that we employed here, it is not fair to compare directly against those results.

We present next the results of using the efficient approximations for the exam school prediction example. In Figure 6, we have included the results of Table 5 alongside the results of using DTC, FITC and PITC for 5, 20 and 50 inducing points. The initial positions of the inducing points are selected using the *k-means* algorithm with the training data points as inputs to the algorithm. The positions of these points are optimized in a scaled conjugate gradient procedure together with the parameters of the model. We notice that using the approximations we obtain similar performances

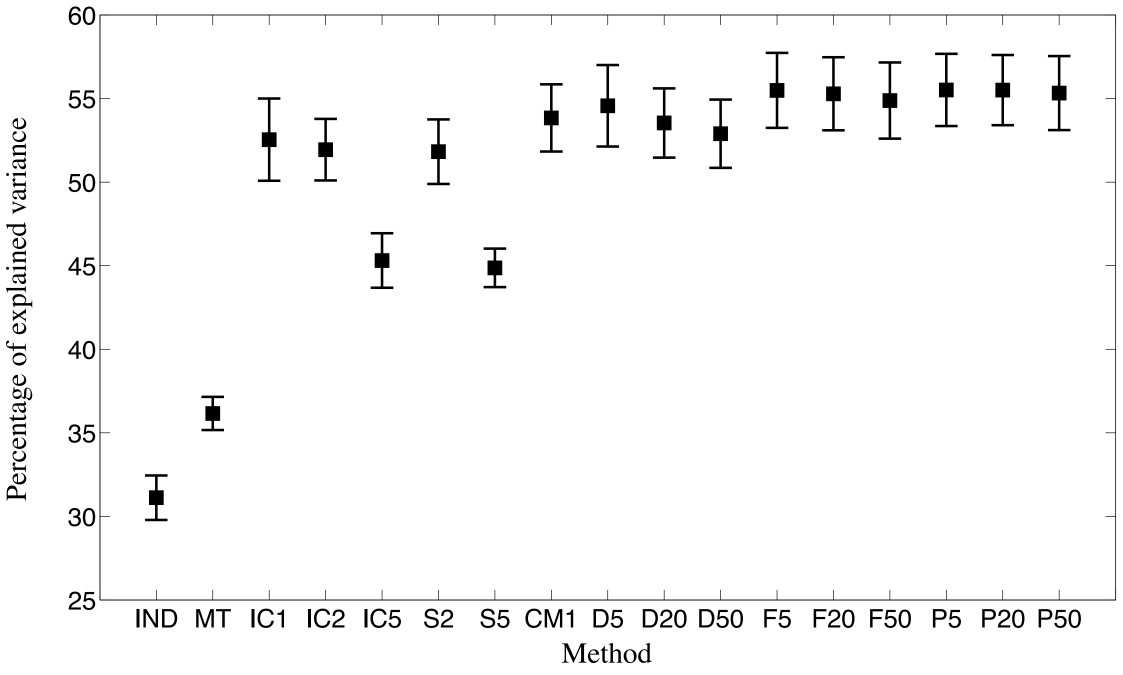


Figure 6: Mean and standard deviation of the percentage of explained variance for exam score prediction results on the ILEA data set. The experiment was repeated ten times. In the bottom of the figure, IND stands for independent GPs, MT stands for multi-task GPs, ICR_1 stands for intrinsic coregionalization model with rank R_1, SQ stands for semiparametric latent factor model with Q latent functions, CM1 stands for convolved multiple output covariance with $Q = 1$ and $R_q = 1$ and DK, FK, PK stands for DTC, FITC and PITC with K inducing points, respectively. The independent GPs and multi-task GPs results were obtained from Bonilla et al. (2008).

to the full models with as few as 5 inducing points. FITC and PITC slightly outperform the DTC method, although results are within the standard deviation.

Table 6 shows the training times for the different methods.[14] Clearly, the efficient approximations are faster than the full methods. This is particularly true when comparing the training times per iteration (second column). The approximations were run over 1000 iterations, but the results for 100 iterations were pretty much the same. For the ICM and SLFM results, definitely more than 100 iterations were needed. With 1000 iterations DTC with 5 inducing points offers a speed up factor of 24 times over the ICM with $R_1 = 1$ and a speed up factor of 137 over the full convolved multiple output method.[15] On the other hand, with 1000 iterations, PITC with 50 inducing points offers a speed up of 9.8 over ICM with $R_1 = 1$ and a speed up of 55 over the full convolved GP method.

14. All experiments with real data were run in workstations with 2.59 GHz, AMD Opteron's and up to 16 GHz of RAM. Only one processor was used on each run.

15. The speed up factor is computed as the relation between the slower method and the faster method, using the training times of the third column in Table 6.

Method	Time per iter. (secs)	Training time (secs)
ICM ($R_1 = 1$)	83.60	16889
ICM ($R_1 = 2$)	85.61	47650
ICM ($R_1 = 5$)	88.02	64535
SLFM ($Q = 2$)	97.00	58564
SLFM ($Q = 5$)	130.23	130234
CMOGP ($Q = 1, R_q = 1$)	95.55	95510
DTC 5 ($Q = 1, R_q = 1$)	0.69	694
DTC 20 ($Q = 1, R_q = 1$)	0.80	804
DTC 50 ($Q = 1, R_q = 1$)	1.04	1046
FITC 5 ($Q = 1, R_q = 1$)	0.94	947
FITC 20 ($Q = 1, R_q = 1$)	1.02	1026
FITC 50 ($Q = 1, R_q = 1$)	1.27	1270
PITC 5 ($Q = 1, R_q = 1$)	1.13	1132
PITC 20 ($Q = 1, R_q = 1$)	1.24	1248
PITC 50 ($Q = 1, R_q = 1$)	1.71	1718

Table 6: Training times for the exam score prediction example. In the table, CMOGP stands for convolved multiple outputs GP. The first column indicates the training time per iteration of each method while the second column indicates the total training time. All the numbers presented are average results over the ten repetitions.

As mentioned before, the approximations reach similar performances using 100 iterations, increasing the speed up factors by ten.

To summarize this example, we have shown that the convolved multiple output GP offers a similar performance to the ICM and SLFM methods. We also showed that the efficient approximations can offer similar performances to the full methods and by a fraction of their training times. Moreover, this example involved a relatively high-input high-output dimensional data set, for which the convolved covariance has not been used before in the literature.

6.3 Heavy Metals in the Swiss Jura

The second example with real data that we consider is the prediction of the concentration of several metal pollutants in a region of the Swiss Jura. This is a relatively low-input low-output dimensional data set that we use to illustrate the ability of the PITC approximation to reach the performance of the full GP if the enough amount of inducing points is used. The data consist of measurements of concentrations of several heavy metals collected in the topsoil of a 14.5 km^2 region of the Swiss Jura. The data is divided into a prediction set (259 locations) and a validation set (100 locations).[16] In a typical situation, referred to as undersampled or heterotopic case, a few expensive measurements of the attribute of interest are supplemented by more abundant data on correlated attributes that are cheaper to sample. We follow the experiment described in Goovaerts (1997, p. 248, 249) in which a *primary variable* (cadmium) at prediction locations in conjunction with some *secondary variables* (nickel and zinc) at prediction and validation locations, are employed to predict the con-

16. This data is available at `http://www.ai-geostats.org/`.

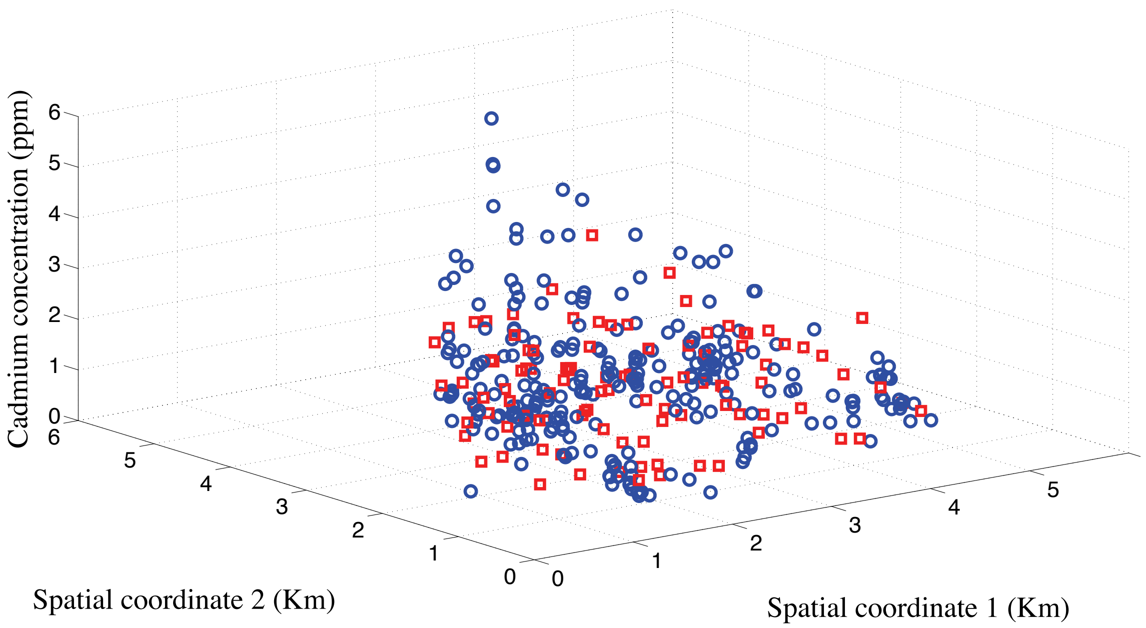


Figure 7: Cadmium concentration for the Swiss Jura example. The blue circles refer to the prediction set (training data for cadmium) and the red squares are the concentrations for the validation set (testing data for cadmium).

centration of the primary variable at validation locations. Figure 7 shows the cadmium concentration for the particular set of input locations of the prediction set (blue circles) and the particular set of input locations of the validation set (red squares). As in the exam score prediction example, we first compare the performances of the full GP methods and later we introduce the performances of the approximations. We compare results of independent GPs, ordinary cokriging, the intrinsic coregionalization model, the semiparametric latent factor model and the convolved multiple output covariance. For independent GPs we use Gaussian covariances with different length-scales for each input dimension. Before describing the particular setup for the other methods appearing in Table 7, we first say a few lines about the cokriging method. The interested reader can find details in several geostatistics books (see Cressie, 1993; Goovaerts, 1997; Wackernagel, 2003).

Cokriging is the generalization of kriging to multiple outputs. It is an unbiased linear predictor that minimizes the error variance between the data and the predicted values. Different cokriging methods assume that each output can be decomposed as a sum of a residual component with zero mean and non-zero covariance function and a trend component. The difference between the cokriging estimators is based on the assumed model for the trend component. While in simple cokriging the mean is assumed to be constant and known, in ordinary cokriging it is assumed to be constant, but unknown, leading to a different set of equations for the predictor. Whichever cokriging method is used implies using the values of the covariance for the residual component in the equations for the prediction, making explicit the need for a positive semidefinite covariance function. In the geostatistics literature, the usual practice is to use the linear model of coregionalization to construct a valid covariance function for the residual component and then use any of the cokriging estimators

Method	Average Mean absolute error
Independent GPs	0.5739 ± 0.0003
Ordinary cokriging (p. 248, 249 Goovaerts, 1997)	0.51
Intrinsic coregionalization model ($R_1 = 2$)	0.4608 ± 0.0025
Semiparametric latent factor model ($Q = 2$)	0.4578 ± 0.0025
Convolved Multiple Outputs GPs ($Q = 2$, $R_q = 1$)	$\mathbf{0.4552 \pm 0.0013}$

Table 7: Average mean absolute error and standard deviation for predicting the concentration of metal cadmium with the full dependent GP model and different forms for the covariance function. The result for ordinary cokriging was obtained from Goovaerts (p. 248, 249 1997) and it is explained in the text. For the intrinsic coregionalization model and the semiparametric latent factor model we use a Gaussian covariance with different length-scales along each input dimension. For the convolved multiple output covariance, we use the covariance described in Section 3. See the text for more details.

for making predictions. A common algorithm to fit the linear model of coregionalization minimizes some error measure between a sample or experimental covariance matrix obtained from the data and the particular matrix obtained from the form chosen for the linear model of coregionalization (Goulard and Voltz, 1992).

Let us go back to the results shown in Table 7. The result that appears as ordinary cokriging was obtained with the ordinary cokriging predictor and a LMC with $Q = 3$ and $R_q = 3$ (p. 119 Goovaerts, 1997). Two of the basic covariances $k_q(\mathbf{x}, \mathbf{x}')$ have a particular polynomial form, while the other corresponds to a bias term.[17] For the prediction stage, only the closest 16 data locations in the primary and secondary variables are employed. Also in Table 7, we present results using the intrinsic coregionalization with a rank two ($R_1 = 2$) for $\mathbf{B}_1$, the semiparametric latent factor model with two latent functions ($Q = 2$) and the convolved multiple output covariance with two latent functions ($Q = 2$ and $R_q = 1$). The choice of either $R_1 = 2$ or $Q = 2$ for the methods was due to the cokriging setup for which two polynomial-type covariances were used. The basic covariances for ICM and SLFM have a Gaussian form with different length scales in each input dimension. For the CMOC, we employ the covariance from Section 3. Parameters for independent GPs, ICM, SLFM and CMOC are learned maximizing the marginal likelihood in Equation (13), using a scaled conjugate gradient procedure. We run the optimization algorithm for up to 200 iterations. Since the prediction and location sets are fixed, we repeat the experiment ten times changing the initial values of the parameters.

Table 7 shows that all methods, including ordinary cokriging, outperform independent GPs. ICM, SLFM and CMOC outperform cokriging. Results for SLFM and CMOC are similar, although CMOC outperformed ICM in every trial of the ten repetitions. The better performance for the SLFM and the CMOC over the ICM would indicate the need for a second latent function with different parameters to the first one. Using a non-instantaneous approach may slightly increase the performance. However, results overlap within one standard deviation.

17. In fact, the linear model of coregionalization employed is constructed using variograms as basic tools that account for the dependencies in the input space. Variograms and covariance functions are related tools used in the geostatistics literature to describe dependencies between variables. A precise definition of the concept of variogram is out of the scope of this paper.

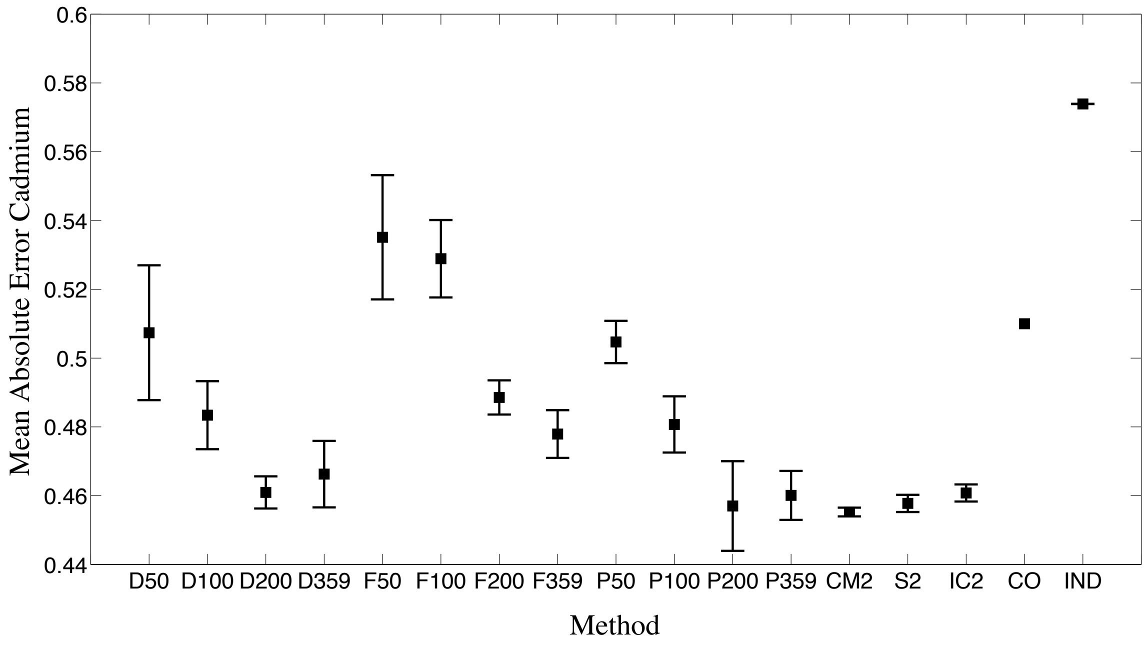


Figure 8: Average mean absolute error and standard deviation for prediction of the pollutant metal cadmium. The experiment was repeated ten times. In the bottom of the figure DK, FK, PK stands for DTC, FITC and PITC with K inducing values, CM2 stands for convolved multiple output covariance with $Q = 2$ and $R_q = 1$, S2 stands for semiparametric latent factor model with $Q = 2$ latent functions, IC2 stands for intrinsic coregionalization model with rank $R_1 = 2$, CO stands for the cokriging method explained in the text and IND stands for independent GPs.

We next include the performances for the efficient approximations. For the results of the approximations, a *k-means* procedure is employed first to find the initial locations of the inducing values and then these locations are optimized in the same optimization procedure used for the parameters. Each experiment is repeated ten times changing the initial value of the parameters. Figure 8 shows the results of prediction for cadmium for the different approximations with varying number of inducing points (this is, different values of K). We also include in the figure the results for the convolved multiple output GP (CM2), semiparametric latent factor model (S2), intrinsic coregionalization model (IC2), ordinary cokriging (CO) and independent GPs (IND).

Notice that DTC and PITC outperform cokriging and independent GPs for any value of K. Also for $K = 200$ and $K = 359$, DTC and PITC reach the performance of the full GP methods, either in average (for $K = 200$) or within one standard deviation (for $K = 359$). $K = 200$ might be a considerable amount of inducing points when compared to the total amount of input training data (359 for nickel and zinc and 259 for cadmium). The need of that amount of inducing points could be explained due to the high variability of the data: mean values for the concentration of pollutant metals are 1.30, 20.01 and 75.88 for cadmium, nickel and zinc, while standard deviations are 0.91,

Method	Time per iter. (secs)	Training time (secs)
ICM	3.84	507
SLFM	4.14	792
CMOGP	4.47	784
DTC 50	0.28	20
DTC 100	0.80	64
DTC 200	1.95	185
DTC 359	4.24	551
FITC 50	0.81	69
FITC 100	1.14	159
FITC 200	2.12	244
FITC 359	5.76	691
PITC 50	1.78	268
PITC 100	2.46	320
PITC 200	4.06	385
PITC 359	7.94	1191

Table 8: Training times for the prediction of the cadmium pollutant metal. In the table, CMOGP stands for convolved multiple outputs GP. The first column indicates the training time per iteration of each method and the second column indicates the total training time. All the numbers presented are average results over the ten repetitions.

8.09 and 30.81 giving coefficients of variation of 70.00%, 40.42% and 40.60%.[18] Variability in cadmium can be observed intuitively from Figure 7. Notice also that FITC outperforms cokriging and independent GPs for $K = 200$ and $K = 359$. The figure also shows that DTC outperforms FITC for all values of K. However, the measure of performance employed, the mean absolute error, does not take into account the predictive variance of the approximated GPs. Using as measures the standardized mean absolute error and the mean standardized log-likelihood, that take into account the predictive variance, FITC outperforms DTC: DTC in average has a MSLL of 0.4544 and a SMSE of 0.9594 while FITC in average has a MSLL of -0.0637 with a SMSE of 0.9102. PITC in average has a MSLL of -0.1226 and SMSE 0.7740. Averages were taken over the different values of K.

Finally, Table 8 shows the timing comparisons for the pollutant example. The training times for DTC with 200 inducing points and PITC with 200 inducing points, which are the first methods that reach the performance of the full GP, are less than any of the times of the full GP methods. For DTC with 200 inducing points, the speed up factor is about 2.74 when compared to ICM and 4.23 when compared to CMOGP. For PITC with 200 inducing points, the speed up factor is 1.31 when compared to ICM and 2.03 when compared to CMOGP. Notice also that all methods are less or equally expensive than the different full GP variants, except for PITC with 359 inducing variables. For this case, however, 4 out of the 10 repetitions reached the average performance in 100 iterations, given a total training time of approximately 794.12 secs., a time much closer to CMOGP and SLFM.

18. The coefficient of variation is defined as the standard deviation over the mean. It could be interpreted also as the inverse of the signal-to-noise ratio.

6.4 Regression Over Gene Expression Data

We now present a third example with real data. This time we only include the performances for the approximations. The goal is to do multiple output regression over gene expression data. The setup was described in Section 4. The difference with that example, is that instead of using $D = 50$ outputs, here we use $D = 1000$ outputs. We do multiple output regression using DTC, FITC and PITC fixing the number of inducing points to $K = 8$ equally spaced in the interval $[-0.5, 11.5]$. Since it is a 1-dimensional input data set, we do not optimize the location of the inducing points, but fix them to the equally spaced initial positions. As for the full GP model in example of Section 4, we make $Q = 1$ and $R_q = 1$. Again we use scaled conjugate gradient to find the parameters that maximize the marginal likelihood in each approximation. The optimization procedure runs for 100 iterations.

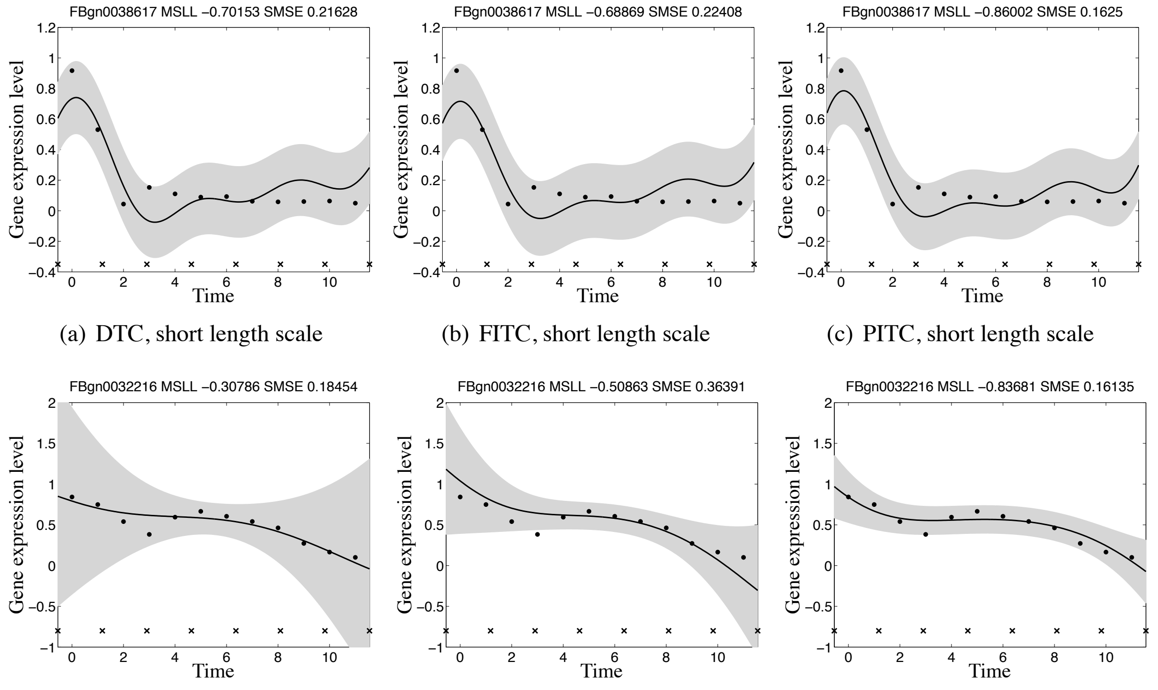
Train set	Test set	Method	Average SMSE	Average MSLL	Average TTPI
Replica 1	Replica 2	DTC	0.5421 ± 0.0085	-0.2493 ± 0.0183	2.04
		FITC	0.5469 ± 0.0125	-0.3124 ± 0.0200	2.31
		PITC	0.5537 ± 0.0136	-0.3162 ± 0.0206	2.59
Replica 2	Replica 1	DTC	0.5454 ± 0.0173	0.6499 ± 0.7961	2.10
		FITC	0.5565 ± 0.0425	-0.3024 ± 0.0294	2.32
		PITC	0.5713 ± 0.0794	-0.3128 ± 0.0138	2.58

Table 9: Standardized mean square error (SMSE), mean standardized log loss (MSLL) and training time per iteration (TTPI) for the gene expression data for 1000 outputs using the efficient approximations for the convolved multiple output GP. The experiment was repeated ten times with a different set of 1000 genes each time. Table includes the value of one standard deviation over the ten repetitions.

Table 9 shows the results of applying the approximations in terms of SMSE and MSLL (columns 4 and 5). DTC and FITC slightly outperforms PITC in terms of SMSE, but PITC outperforms both DTC and FITC in terms of MSLL. This pattern repeats itself when the training data comes from replica 1 or from replica 2.

In Figure 9 we show the performance of the approximations over the same two genes of Figure 1, these are FBgn0038617 and FBgn0032216. The non-instantaneous mixing effect of the model can still be observed. Performances for these particular genes are highlighted in Table 10. Notice that the performances are between the actual performances for the LMC and the CMOC appearing in Table 2. We include these figures only for illustrative purposes, since both experiments use a different number of outputs. Figures 1 and 2 were obtained as part of multiple output regression problem of $D = 50$ outputs, while Figures 9 and 10 were obtained in a multiple output regression problem with $D = 1000$ outputs.

In Figure 10, we replicate the same exercise for the genes FBgn0010531 and FBgn0004907, that also appeared in Figure 2. Performances for DTC, FITC and PITC are shown in Table 10 (last six rows), which compare favourably with the performances for the linear model of coregionalization in Table 2 and close to the performances for the CMOC. In average, PITC outperforms the other methods for the specific set of genes in both figures above.



(a) DTC, short length scale (b) FITC, short length scale (c) PITC, short length scale

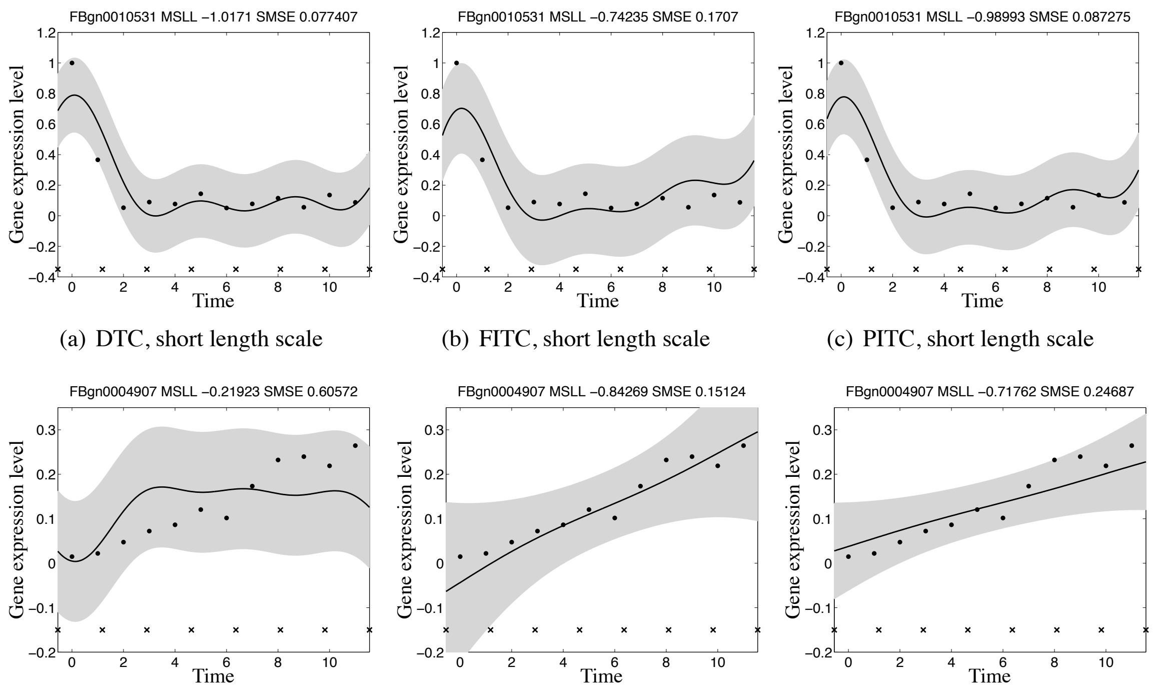
(d) DTC, long length scale (e) FITC, long length scale (f) PITC, long length scale

Figure 9: Predictive mean and variance for genes FBgn0038617 (first row) and FBgn0032216 (second row) using the different approximations. In the first column DTC in Figures 9(a) and 9(d), second column FITC in Figures 9(b) and 9(e), and in the third column PITC in Figures 9(c) and 9(f). The training data comes from replica 1 and the testing data from replica 2. The solid line corresponds to the predictive mean, the shaded region corresponds to 2 standard deviations of the prediction. Performances in terms of SMSE and MSLL are given in the title of each figure. The adjectives "short" and "long" given to the length-scales in the captions of each figure, must be understood like relative to each other. The crosses in the bottom of each figure indicate the positions of the inducing points, which remain fixed during the training procedure.

With respect to the training times, the Table 9 in the column 6 shows the average training time per iteration (average TTPI) for each approximation. To have an idea of the saving times, one iteration of the full GP model for the same 1000 genes would take around 4595.3 seconds. This gives a speed up factor of 1780, approximately.

7. Conclusions

In this paper we first presented a review of different alternatives for multiple output regression grouped under a similar framework known as the linear model of coregionalization. Then we illustrated how the linear model of coregionalization can be interpreted as an instantaneous mixing of latent functions, in contrast to a convolved multiple output framework, where the mixing



(a) DTC, short length scale (b) FITC, short length scale (c) PITC, short length scale

(d) DTC, long length scale (e) FITC, long length scale (f) PITC, long length scale

Figure 10: Predictive mean and variance for genes FBgn0010531 (first row) and FBgn0004907 (second row) using the different approximations. In the first column DTC in Figures 10(a) and 10(d), second column FITC in Figures 10(b) and 10(e), and in the third column PITC in Figures 10(c) and 10(f). The training data comes now from replica 2 and the testing data from replica 1. The solid line corresponds to the predictive mean, the shaded region corresponds to 2 standard deviations of the prediction. Performances in terms of SMSE and MSLL are given in the title of each figure. The crosses in the bottom of each figure indicate the positions of the inducing points, which remain fixed during the training procedure.

is not necessarily instantaneous. Experimental results showed that in systems with a presence of some dynamics (for example, the gene expression data set), having this additional element of non-instantaneous mixing can lead to simpler explanations of the data. While, in systems for which the dynamics is not so obvious (for example, the exam score prediction data set), the benefit of using the non-instantaneous mixing was less noticeable.

We have also presented different efficient approximations for multiple output GPs, in the context of convolution processes. Using these approximations we can capture the correlated information among outputs while reducing the amount of computational load for prediction and optimization purposes. The computational complexity for the DTC and the FITC approximations is $O(NDK^2)$. The reduction in computational complexity for the PITC approximation is from $O(N^3D^3)$ to $O(N^3D)$. This matches the computational complexity for modeling with independent GPs. However, as we have seen, the predictive power of independent GPs is lower. Also, since

Test replica	Test genes	Method	SMSE	MSLL
Replica 2	FBgn0038617	DTC	0.2162	-0.7015
		FITC	0.2240	-0.6886
		PITC	0.1625	-0.8600
	FBgn0032216	DTC	0.1845	-0.3078
		FITC	0.3639	-0.5086
		PITC	0.1613	-0.8368
Replica 1	FBgn0010531	DTC	0.0774	-1.0171
		FITC	0.1707	-0.7423
		PITC	0.0872	-0.9899
	FBgn0004907	DTC	0.6057	-0.2192
		FITC	0.1512	-0.8426
		PITC	0.2468	-0.7176

Table 10: Standardized mean square error (SMSE) and mean standardized log loss (MSLL) for the genes in Figures 9 and 10 for DTC, FITC and PITC with $K = 8$. Genes FBgn0038617 and FBgn0010531 have a shorter length-scale when compared to genes FBgn0032216 and FBgn0004907.

PITC makes a better approximation of the likelihood, the variance of the results is usually lower and approaches closely to the performance of the full GP, when compared to DTC and FITC. As a byproduct of seeing the linear model of coregionalization as a particular case of the convolved GPs, we can extend all the approximations to work under the linear model of coregionalization regime.

With an appropriate selection of the kernel smoothing function we have an indirect way to generate different forms for the covariance function in the multiple output setup. We showed an example with Gaussian kernels, for which a suitable standardization of the kernels can be made, leading to competitive results in high-dimensional input regression problems, as seen in the school exam score prediction problem. The authors are not aware of other work in which this convolution process framework has been applied in problems with high input dimensions.

As shown with the Swiss Jura experiment, we might need a considerable amount of inducing points compared to the amount of training data, when doing regression over very noisy outputs. This agrees to some extent with our intuition in Section 5, where we conditioned the validity of the approximations to the smoothness of the latent functions. However, even for this case, we can obtain the same performances in a fraction of the time that takes to train a full GP. Moreover, the approximations allow multiple output regression over a large amount of outputs, in scenarios where training a full GP become extremely expensive. We showed an example of this type with the multiple output regression over the gene expression data.

Linear dynamical systems responses can be expressed as a convolution between the impulse response of the system with some input function. This convolution approach is an equivalent way of representing the behavior of the system through a linear differential equation. For systems involving high amounts of coupled differential equations (Álvarez et al., 2009; Álvarez et al., 2011a; Honkela et al., 2010), the approach presented here is a reasonable way of obtaining approximate solutions and incorporating prior domain knowledge to the model.

Recently, Titsias (2009) highlighted how optimizing inducing variables can be problematic as they introduce many hyperparameters in the likelihood term. Titsias (2009) proposed a variational method with an associated lower bound where inducing variables are *variational parameters*. Following the ideas presented here, we can combine easily the method of Titsias (2009) and propose a lower bound for the multiple output case. We have followed a first attempt in that direction and some results have been presented in Álvarez et al. (2010).

Acknowledgments

The authors would like to thank Edwin Bonilla for his valuable feedback with respect to the exam score prediction example. The work has benefited greatly from discussions with David Luengo, Michalis Titsias, and Magnus Rattray. We also thank to three anonymous reviewers for their helpful comments. The authors are very grateful for support from a Google Research Award "Mechanistically Inspired Convolution Processes for Learning" and the EPSRC Grant No EP/F005687/1 "Gaussian Processes for Systems Identification with Applications in Systems Biology". MA also acknowledges the support from the Overseas Research Student Award Scheme (ORSAS), from the School of Computer Science of the University of Manchester and from the Universidad Tecnológica de Pereira, Colombia.

Appendix A. Derivatives for the Approximations

In this appendix, we present the derivatives needed to apply the gradient methods in the optimization routines. We present the first order derivatives of the log-likelihood with respect to $\mathbf{K}_{\mathbf{f},\mathbf{f}}$, $\mathbf{K}_{\mathbf{u},\mathbf{f}}$ and $\mathbf{K}_{\mathbf{u},\mathbf{u}}$. These derivatives can be combined with the derivatives of $\mathbf{K}_{\mathbf{f},\mathbf{f}}$, $\mathbf{K}_{\mathbf{u},\mathbf{f}}$ and $\mathbf{K}_{\mathbf{u},\mathbf{u}}$ with respect to $\boldsymbol{\theta}$ and employ these expressions in a gradient-like optimization procedure.

We follow the notation of Brookes (2005) obtaining similar results to Lawrence (2007). This notation allows us to apply the chain rule for matrix derivation in a straight-forward manner. Let's define $\mathbf{G{:}} = \operatorname{vec}\mathbf{G}$, where vec is the vectorization operator over the matrix $\mathbf{G}$. For a function $\mathcal{L}$ the equivalence between $\frac{\partial\mathcal{L}}{\partial\mathbf{G}}$ and $\frac{\partial\mathcal{L}}{\partial\mathbf{G{:}}}$ is given through $\frac{\partial\mathcal{L}}{\partial\mathbf{G{:}}} = \left(\left(\frac{\partial\mathcal{L}}{\partial\mathbf{G}}\right){:}\right)^{\top}$. The obtain the hyperparameters, we maximize the following log-likelihood function,

$$\mathcal{L}(\mathbf{Z},\boldsymbol{\theta}) \propto -\frac{1}{2}\log|\mathbf{D}+\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}| - \frac{1}{2}\operatorname{trace}\left[\left(\mathbf{D}+\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)^{-1}\mathbf{y}\mathbf{y}^{\top}\right] \qquad (19)$$

where we have redefined $\mathbf{D}$ as $\mathbf{D} = \left[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right]\odot\mathbf{M}+\boldsymbol{\Sigma}$, to keep a simpler notation. Using the matrix inversion lemma and its equivalent form for determinants, expression (19) can be written as

$$\begin{aligned}\mathcal{L}(\mathbf{Z},\boldsymbol{\theta}) \propto & \frac{1}{2}\log|\mathbf{K}_{\mathbf{u},\mathbf{u}}| - \frac{1}{2}\log|\mathbf{A}| - \frac{1}{2}\log|\mathbf{D}| - \frac{1}{2}\operatorname{trace}\left[\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^{\top}\right] \\ & + \frac{1}{2}\operatorname{trace}\left[\mathbf{D}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^{\top}\right].\end{aligned}$$

We can find $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}$ and $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}}$ applying the chain rule to $\mathcal{L}$ obtaining expressions for $\frac{\partial\mathcal{L}}{\partial\mathbf{K}_{\mathbf{f},\mathbf{f}}}$, $\frac{\partial\mathcal{L}}{\partial\mathbf{K}_{\mathbf{f},\mathbf{u}}}$ and $\frac{\partial\mathcal{L}}{\partial\mathbf{K}_{\mathbf{u},\mathbf{u}}}$ and combining those with the derivatives of the covariances wrt $\boldsymbol{\theta}$ and $\mathbf{Z}$,

$$\frac{\partial\mathcal{L}}{\partial\mathbf{G{:}}} = \frac{\partial\mathcal{L}_{\mathbf{A}}}{\partial\mathbf{A{:}}}\frac{\partial\mathbf{A{:}}}{\partial\mathbf{D{:}}}\frac{\partial\mathbf{D{:}}}{\partial\mathbf{G{:}}} + \frac{\partial\mathcal{L}_{\mathbf{D}}}{\partial\mathbf{D{:}}}\frac{\partial\mathbf{D{:}}}{\partial\mathbf{G{:}}} + \left[\frac{\partial\mathcal{L}_{\mathbf{A}}}{\partial\mathbf{A{:}}}\frac{\partial\mathbf{A{:}}}{\partial\mathbf{G{:}}} + \frac{\partial\mathcal{L}_{\mathbf{G}}}{\partial\mathbf{G{:}}}\right]\delta_{GK}, \qquad (20)$$

where the subindex in $\mathcal{L}_{\mathbf{E}}$ stands for those terms of $\mathcal{L}$ which depend on $\mathbf{E}$, $\mathbf{G}$ is either $\mathbf{K}_{\mathbf{f},\mathbf{f}}$, $\mathbf{K}_{\mathbf{u},\mathbf{f}}$ or $\mathbf{K}_{\mathbf{u},\mathbf{u}}$ and δ_{GK} is zero if $\mathbf{G}$ is equal to $\mathbf{K}_{\mathbf{f},\mathbf{f}}$ and one in other case. Next we present expressions for each partial derivative

$$\frac{\partial \mathcal{L}_{\mathbf{A}}}{\partial \mathbf{A}:} = -\frac{1}{2}(\mathbf{C}:)^\top, \quad \frac{\partial \mathbf{A}:}{\partial \mathbf{D}:} = -\left(\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\otimes\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\right), \quad \frac{\partial \mathcal{L}_{\mathbf{D}}}{\partial \mathbf{D}:} = -\frac{1}{2}\left(\left(\mathbf{D}^{-1}\mathbf{H}\mathbf{D}^{-1}\right):\right)^\top$$

$$\frac{\partial \mathbf{D}:}{\partial \mathbf{K}_{\mathbf{f},\mathbf{f}}:} = \operatorname{diag}(\mathbf{M}:), \quad \frac{\partial \mathbf{D}:}{\partial \mathbf{K}_{\mathbf{u},\mathbf{f}}:} = -\operatorname{diag}(\mathbf{M}:)\left[\left(\mathbf{I}\otimes\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\right) + \left(\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\otimes\mathbf{I}\right)\mathbf{T}_{\mathbf{D}}\right],$$

$$\frac{\partial \mathbf{D}:}{\partial \mathbf{K}_{\mathbf{u},\mathbf{u}}:} = \operatorname{diag}(\mathbf{M}:)\left(\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\otimes\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\right), \frac{\partial \mathbf{A}:}{\partial \mathbf{K}_{\mathbf{u},\mathbf{f}}:} = \left(\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\otimes\mathbf{I}\right) + \left(\mathbf{I}\otimes\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\right)\mathbf{T}_{\mathbf{A}}$$

$$\frac{\partial \mathbf{A}:}{\partial \mathbf{K}_{\mathbf{u},\mathbf{u}}:} = \mathbf{I}, \quad \frac{\partial \mathcal{L}_{\mathbf{K}_{\mathbf{u},\mathbf{f}}}}{\partial \mathbf{K}_{\mathbf{u},\mathbf{f}}:} = \left(\left(\mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{D}^{-1}\right):\right)^\top, \quad \frac{\partial \mathcal{L}_{\mathbf{K}_{\mathbf{u},\mathbf{u}}}}{\partial \mathbf{K}_{\mathbf{u},\mathbf{u}}:} = \frac{1}{2}\left(\left(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\right):\right)^\top,$$

where $\mathbf{C} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{D}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{A}^{-1}$, $\mathbf{T}_{\mathbf{D}}$ and $\mathbf{T}_{\mathbf{A}}$ are *vectorized transpose matrices* (see, e.g., Brookes, 2005) and $\mathbf{H} = \mathbf{D} - \mathbf{y}\mathbf{y}^\top + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top + \left(\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top\right)^\top$. We can replace the above expressions in (20) to find the corresponding derivatives, so

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{f},\mathbf{f}}:} &= \frac{1}{2}\left[\left((\mathbf{C}):\right)^\top\left(\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\otimes\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\right) - \frac{1}{2}\left(\left(\mathbf{D}^{-1}\mathbf{H}\mathbf{D}^{-1}\right):\right)^\top\right]\operatorname{diag}(\mathbf{M}:) &(21)\\ &= -\frac{1}{2}\left(\left(\mathbf{D}^{-1}\mathbf{J}\mathbf{D}^{-1}\right):\right)^\top\operatorname{diag}(\mathbf{M}:) = -\frac{1}{2}\left(\operatorname{diag}(\mathbf{M}:)\left(\mathbf{D}^{-1}\mathbf{J}\mathbf{D}^{-1}\right):\right)^\top &(22)\\ &= -\frac{1}{2}\left(\left(\mathbf{D}^{-1}\mathbf{J}\mathbf{D}^{-1}\odot\mathbf{M}\right):\right)^\top = -\frac{1}{2}(\mathbf{Q}:)^\top &(23)\end{aligned}$$

or simply

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{f},\mathbf{f}}} = -\frac{1}{2}\mathbf{Q},$$

where $\mathbf{J} = \mathbf{H} - \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{C}\mathbf{K}_{\mathbf{u},\mathbf{f}}$ and $\mathbf{Q} = \left(\mathbf{D}^{-1}\mathbf{J}\mathbf{D}^{-1}\odot\mathbf{M}\right)$. We have used the property $(\mathbf{B}:)^\top(\mathbf{F}\otimes\mathbf{P}) = \left(\left(\mathbf{P}^\top\mathbf{B}\mathbf{F}\right):\right)^\top$ in (21) and the property $\operatorname{diag}(\mathbf{B}:)\mathbf{F}: = (\mathbf{B}\odot\mathbf{F}):$, to go from (22) to (23). We also have

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{u},\mathbf{f}}:} &= \frac{1}{2}(\mathbf{Q}:)^\top\left[\left(\mathbf{I}\otimes\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\right) + \left(\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\otimes\mathbf{I}\right)\mathbf{T}_{\mathbf{D}}\right] - \frac{1}{2}(\mathbf{C}:)^\top\\ &\quad\left[\left(\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\otimes\mathbf{I}\right) + \left(\mathbf{I}\otimes\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\right)\mathbf{T}_{\mathbf{A}}\right] + \left(\left(\mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{D}^{-1}\right):\right)^\top &(24)\\ &= \left(\left(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{Q} - \mathbf{C}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1} + \mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{D}^{-1}\right):\right)^\top\end{aligned}$$

or simply

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{u},\mathbf{f}}} = \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{Q} - \mathbf{C}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1} + \mathbf{A}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{D}^{-1}\mathbf{y}\mathbf{y}^\top\mathbf{D}^{-1},$$

where in (24), $(\mathbf{Q}:)^\top(\mathbf{F}\otimes\mathbf{I})\mathbf{T}_{\mathbf{D}} = (\mathbf{Q}:)^\top\mathbf{T}_{\mathbf{D}}(\mathbf{I}\otimes\mathbf{F}) = \left(\mathbf{T}_{\mathbf{D}}^\top\mathbf{Q}:\right)^\top(\mathbf{I}\otimes\mathbf{F}) = (\mathbf{Q}:)^\top(\mathbf{I}\otimes\mathbf{F})$. A similar analysis is formulated for the term involving $\mathbf{T}_{\mathbf{A}}$. Finally, results for $\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{u},\mathbf{f}}}$ and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}$ are obtained as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{u},\mathbf{u}}} = -\frac{1}{2}\left(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} - \mathbf{C} - \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\mathbf{Q}\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\right), \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} = -\frac{1}{2}\mathbf{Q}.$$

References

Mauricio A. Álvarez and Neil D. Lawrence. Sparse convolved Gaussian processes for multi-output regression. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 57–64. MIT Press, Cambridge, MA, 2009.

Mauricio A. Álvarez, David Luengo, and Neil D. Lawrence. Latent Force Models. In David van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 9–16. JMLR W&CP 5, Clearwater Beach, Florida, 16-18 April 2009.

Mauricio A. Álvarez, David Luengo, Michalis K. Titsias, and Neil D. Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 25–32. JMLR W&CP 9, Chia Laguna, Sardinia, Italy, 13-15 May 2010.

Mauricio A. Álvarez, Jan Peters, Bernhard Schölkopf, and Neil D. Lawrence. Switched latent force models for movement segmentation. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 55–63. MIT Press, Cambridge, MA, 2011a.

Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: a review, 2011b. Universidad Tecnológica de Pereira, Massachusetts Institute of Technology and University of Sheffield. In preparation.

Bart Bakker and Tom Heskes. Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

Ronald Paul Barry and Jay M. Ver Hoef. Blackbox kriging: spatial prediction without specifying variogram models. *Journal of Agricultural, Biological and Environmental Statistics*, 1(3):297–322, 1996.

Edwin V. Bonilla, Kian Ming Chai, and Christopher K. I. Williams. Multi-task Gaussian process prediction. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 153–160. MIT Press, Cambridge, MA, 2008.

Phillip Boyle and Marcus Frean. Dependent Gaussian processes. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 217–224. MIT Press, Cambridge, MA, 2005.

Michael Brookes. The matrix reference manual. Available on-line., 2005. `http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html`.

Catherine A. Calder. *Exploring Latent Structure in Spatial Temporal Processes Using Process Convolutions*. PhD thesis, Institute of Statistics and Decision Sciences, Duke University, Durham, NC, USA, 2003.

Catherine A. Calder. Dynamic factor process convolution models for multivariate space-time data with application to air quality assessment. *Environmental and Ecological Statistics*, 14(3):229–247, 2007.

Catherine A. Calder and Noel Cressie. Some topics in convolution-based spatial modeling. In *Proceedings of the 56th Session of the International Statistics Institute*, August 2007.

Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

Kian Ming A. Chai, Christopher K. I. Williams, Stefan Klanke, and Sethu Vijayakumar. Multi-task Gaussian process learning of robot inverse dynamics. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 265–272. MIT Press, Cambridge, MA, 2009.

Noel A. C. Cressie. *Statistics for Spatial Data*. John Wiley & Sons (Revised edition), USA, 1993.

Lehel Csató and Manfred Opper. Sparse representation for Gaussian process models. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 444–450. MIT Press, Cambridge, MA, 2001.

Theodoros Evgeniou and Massimiliano Pontil. Regularized Multi-task Learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, 2004.

Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

Montserrat Fuentes. Interpolation of nonstationary air pollution processes: a spatial spectral approach. *Statistical Modelling*, 2:281–298, 2002a.

Montserrat Fuentes. Spectral methods for nonstationary spatial processes. *Biometrika*, 89(1):197–210, 2002b.

Pei Gao, Antti Honkela, Magnus Rattray, and Neil D. Lawrence. Gaussian process modelling of latent chemical species: Applications to inferring transcription factor activities. *Bioinformatics*, 24:i70–i75, 2008. doi: 10.1093/bioinformatics/btn278.

Marc G. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

Harvey Goldstein. Multilevel modelling of survey data. *The Statistician*, 40(2):235–244, 1991.

Pierre Goovaerts. *Geostatistics For Natural Resources Evaluation*. Oxford University Press, USA, 1997.

Michel Goulard and Marc Voltz. Linear coregionalization model: Tools for estimation and choice of cross-variogram matrix. *Mathematical Geology*, 24(3):269–286, 1992.

Jeffrey D. Helterbrand and Noel Cressie. Universal cokriging under intrinsic coregionalization. *Mathematical Geology*, 26(2):205–226, 1994.

Tom Heskes. Empirical Bayes for learning to learn. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning 17*, pages 367–374. Morgan Kaufmann, San Francisco, CA, June 29-July 2 2000.

David M. Higdon. A process-convolution approach to modeling temperatures in the north atlantic ocean. *Journal of Ecological and Environmental Statistics*, 5:173–190, 1998.

David M. Higdon. Space and space-time modelling using process convolutions. In C. Anderson, V. Barnett, P. Chatwin, and A. El-Shaarawi, editors, *Quantitative Methods for Current Environmental Issues*, pages 37–56. Springer-Verlag, 2002.

David M. Higdon, Jenise Swall, and John Kern. Non-stationary spatial modeling. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 6*, pages 761–768. Oxford University Press, 1998.

Antti Honkela, Charles Girardot, E. Hilary Gustafson, Ya-Hsin Liu, Eileen E. M. Furlong, Neil D. Lawrence, and Magnus Rattray. Model-based method for transcription factor target identification with limited data. *Proc. Natl. Acad. Sci.*, 107(17):7793–7798, 2010.

Andre G. Journel and Charles J. Huijbregts. *Mining Geostatistics*. Academic Press, London, 1978. ISBN 0-12391-050-1.

Neil D. Lawrence. Learning for larger datasets with the Gaussian process latent variable model. In Marina Meila and Xiaotong Shen, editors, *AISTATS 11*, pages 243–250. Omnipress, San Juan, Puerto Rico, 21-24 March 2007.

Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Sue Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 625–632. MIT Press, Cambridge, MA, 2003.

Neil D. Lawrence, Guido Sanguinetti, and Magnus Rattray. Modelling transcriptional regulation using Gaussian processes. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19*, pages 785–792. MIT Press, Cambridge, MA, 2007.

Feng Liang, Kai Mao, Ming Liao, Sayan Mukherjee, and Mike West. Non-parametric Bayesian kernel models. Department of Statistical Science, Duke University, Discussion Paper 07-10. (Submitted for publication), 2009.

Desmond L. Nuttall, Harvey Goldstein, Robert Prosser, and Jon Rasbash. Differential school effectiveness. *International Journal of Educational Research*, 13(7):769–776, 1989.

Michael A. Osborne and Stephen J. Roberts. Gaussian processes for prediction. Technical report, Department of Engineering Science, University of Oxford, 2007.

Michael A. Osborne, Alex Rogers, Sarvapali D. Ramchurn, Stephen J. Roberts, and Nicholas R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN 2008)*, 2008.

Christopher J. Paciorek and Mark J. Schervish. Nonstationary covariance functions for Gaussian process regression. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Natesh S. Pillai, Qiang Wu, Feng Liang, Sayan Mukherjee, and Robert L. Wolpert. Characterizing the function space for Bayesian kernel models. *Journal of Machine Learning Research*, 8:1769–1797, 2007.

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

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. ISBN 0-262-18253-X.

Matthias Seeger, Christopher K. I. Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In Christopher M. Bishop and Brendan J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, FL, 3–6 Jan 2003.

Alexander J. Smola and Peter L. Bartlett. Sparse greedy Gaussian process regression. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, Cambridge, MA, 2001.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Yair Weiss, Bernhard Schölkopf, and John C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, Cambridge, MA, 2006.

Edward Snelson and Zoubin Ghahramani. Local and global sparse Gaussian process approximations. In Marina Meila and Xiaotong Shen, editors, *AISTATS 11*, pages 524–531, San Juan, Puerto Rico, 21-24 March 2007. Omnipress.

Yee Whye Teh, Matthias Seeger, and Michael I. Jordan. Semiparametric latent factor models. In Robert G. Cowell and Zoubin Ghahramani, editors, *AISTATS 10*, pages 333–340. Society for Artificial Intelligence and Statistics, Barbados, 6-8 January 2005.

Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 567–574. JMLR W&CP 5, Clearwater Beach, Florida, 16-18 April 2009.

Pavel Tomancak, Amy Beaton, Richard Weiszmann, Elaine Kwan, ShengQiang Shu, Suzanna E Lewis, Stephen Richards, Michael Ashburner, Volker Hartenstein, Susan E Celniker, and Gerald M Rubin. Systematic determination of patterns of gene expression during drosophila embryogenesis. *Genome Biology*, 3(12):research0088.1–0088.14, 2002.

Jay M. Ver Hoef and Ronald Paul Barry. Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Plannig and Inference*, 69:275–294, 1998.

Hans Wackernagel. *Multivariate Geostatistics*. Springer-Verlag Heidelberg New York, 2003.

Christopher K. Wikle. A kernel-based spectral model for non-Gaussian spatio-temporal processes. *Statistical Modelling*, 2:299–314, 2002.

Christopher K. Wikle. Hierarchical Bayesian models for predicting the spread of ecological processes. *Ecology*, 84(6):1382–1394, 2003.

Christopher K. Wikle, L. Mark Berliner, and Noel Cressie. Hierarchical Bayesian space-time models. *Environmental and Ecological Statistics*, 5:117–154, 1998.

Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, Cambridge, MA, 2001.

Ya Xue, Xuejun Liao, and Lawrence Carin. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

Robert P. Zinzen, Charles Girardot, Julien Gagneur, Martina Braun, and Eileen E. M. Furlong. Combinatorial binding predicts spatio-temporal cis-regulatory activity. *Nature*, 462:65–70, 2009.

Journal of Machine Learning Research 12 (2011) 1501-1536 Submitted 10/10; Revised 2/11; Published 5/11

Learning from Partial Labels

Timothee Cour TIMOTHEE.COUR@GMAIL.COM
NEC Laboratories America
10080 N Wolfe Rd # Sw3350
Cupertino, CA 95014, USA

Benjamin Sapp BENSAPP@CIS.UPENN.EDU
Ben Taskar TASKAR@SEAS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
3330 Walnut Street
Philadelphia, PA 19107, USA

Editor: Yoav Freund

Abstract

We address the problem of partially-labeled multiclass classification, where instead of a single label per instance, the algorithm is given a candidate set of labels, only one of which is correct. Our setting is motivated by a common scenario in many image and video collections, where only partial access to labels is available. The goal is to learn a classifier that can disambiguate the partially-labeled training instances, and generalize to unseen data. We define an intuitive property of the data distribution that sharply characterizes the ability to learn in this setting and show that effective learning is possible even when all the data is only partially labeled. Exploiting this property of the data, we propose a convex learning formulation based on minimization of a loss function appropriate for the partial label setting. We analyze the conditions under which our loss function is asymptotically consistent, as well as its generalization and transductive performance. We apply our framework to identifying faces culled from web news sources and to naming characters in TV series and movies; in particular, we annotated and experimented on a very large video data set and achieve 6% error for character naming on 16 episodes of the TV series *Lost*.

Keywords: weakly supervised learning, multiclass classification, convex learning, generalization bounds, names and faces

1. Introduction

We consider a weakly-supervised multiclass classification setting where each instance is partially labeled: instead of a single label per instance, the algorithm is given a candidate set of labels, only one of which is correct. A typical example arises in photographs containing several faces per image and a caption that only specifies who is in the picture but not which name matches which face. In this setting each face is ambiguously labeled with the set of names extracted from the caption, see Figure 1 (bottom). Photograph collections with captions have motivated much recent interest in weakly annotated images and videos (Duygulu et al., 2002; Barnard et al., 2003; Berg et al., 2004; Gallagher and Chen, 2007). Another motivating example is shown in Figure 1 (top), which shows a setting where we can obtain plentiful but weakly labeled data: videos and screenplays. Using a screenplay, we can tell who is in a given scene, but for every detected face in the scene, the person's

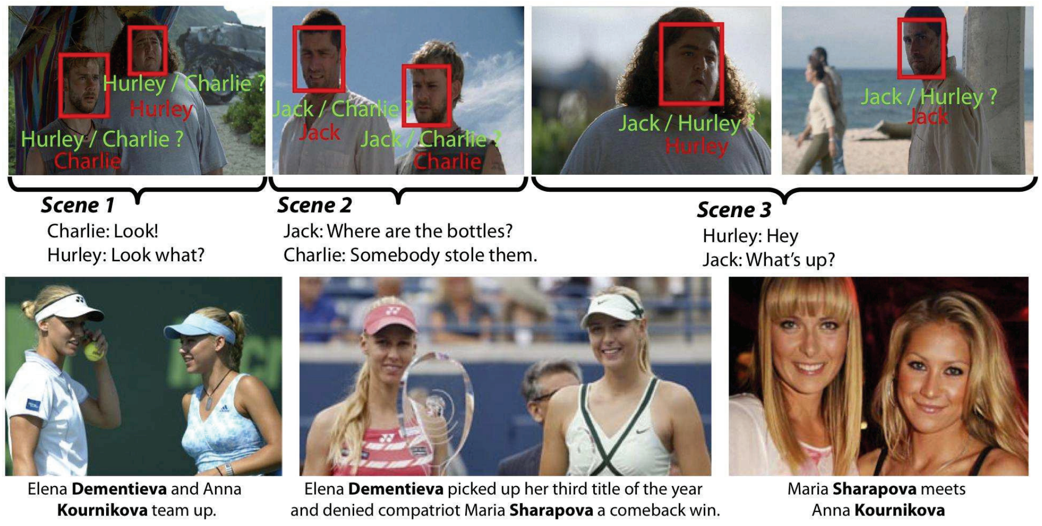

Hurley / Charlie ?
Hurley
Hurley / Charlie ?
Charlie
Jack / Charlie ?
Jack
Jack / Charlie ?
Charlie
Jack / Hurley ?
Hurley
Jack / Hurley ?
Jack
Scene 1
Charlie: Look!
Hurley: Look what?
Scene 2
Jack: Where are the bottles?
Charlie: Somebody stole them.
Scene 3
Hurley: Hey
Jack: What's up?
Elena Dementieva and Anna Kournikova team up.
Elena Dementieva picked up her third title of the year and denied compatriot Maria Sharapova a comeback win.
Maria Sharapova meets Anna Kournikova

Figure 1: Two examples of partial labeling scenarios for naming faces. **Top:** using a screenplay, we can tell who is in a movie scene, but for every face in the corresponding images, the person's identity is ambiguous (green labels). **Bottom:** images in photograph collections and webpages are often tagged ambiguously with several potential names in the caption or nearby text. In both cases, our goal is to learn a model from ambiguously labeled examples so as to disambiguate the training labels and also generalize to unseen examples.

identity is ambiguous: each face is partially labeled with the set of characters appearing at some point in the scene (Satoh et al., 1999; Everingham et al., 2006; Ramanan et al., 2007). The goal in each case is to learn a person classifier that can not only disambiguate the labels of the training faces, but also generalize to unseen data. Learning accurate models for face and object recognition from such imprecisely annotated images and videos can improve the performance of many applications, including image retrieval and video summarization.

This partially labeled setting is situated between fully supervised and fully unsupervised learning, but is qualitatively different from the semi-supervised setting where both labeled and unlabeled data are available. There have been several papers that addressed this partially labeled (also called ambiguously labeled) problem. Many formulations use the expectation-maximization-like algorithms to estimate the model parameters and "fill-in" the labels (Côme et al., 2008; Ambroise et al., 2001; Vannoorenberghe and Smets, 2005; Jin and Ghahramani, 2002). Most methods involve either non-convex objectives or procedural, iterative reassignment schemes which come without any guarantees of achieving global optima of the objective or classification accuracy. To the best of our knowledge, there has not been theoretical analysis of conditions under which proposed approaches are guaranteed to learn accurate classifiers. The contributions of this paper are:

- We show theoretically that effective learning is possible under reasonable distributional assumptions even when all the data is partially labeled, leading to useful upper and lower bounds on the true error.

- We propose a convex learning formulation based on this analysis by extending general multiclass loss functions to handle partial labels.

- We apply our convex learning formulation to the task of identifying faces culled from web news sources, and to naming characters in TV series. We experiment on a large data set consisting of 100 hours of video, and in particular achieve 6% (resp. 13%) error for character naming across 8 (resp. 32) labels on 16 episodes of *Lost*, consistently outperforming several strong baselines.

- We contribute the *Annotated Faces on TV data set*, which contains about 3,000 cropped faces extracted from 8 episodes of the TV show *Lost* (one face per track). Each face is registered and annotated with a groundtruth label (there are 40 different characters). We also include a subset of those faces with the partial label set automatically extracted from the screenplay.

- We provide the *Convex Learning from Partial Labels Toolbox*, an open-source matlab and C++ implementation of our approach as well as the baseline approach discussed in the paper. The code includes scripts to illustrate the process on Faces in the Wild Data Set (Huang et al., 2007a) and our Annotated Faces on TV data set.

The paper is organized as follows.[1] We review related work and relevant learning scenarios in Section 2. We pose the partially labeled learning problem as minimization of an ambiguous loss in Section 3, and establish upper and lower bounds between the (unobserved) true loss and the (observed) ambiguous loss in terms of a critical distributional property we call ambiguity degree. We propose the novel *Convex Learning from Partial Labels* (CLPL) formulation in Section 4, and show it offers a tighter approximation to the ambiguous loss, compared to a straightforward formulation. We derive generalization bounds for the inductive setting, and in Section 5 also provide bounds for the transductive setting. In addition, we provide reasonable sufficient conditions that will guarantee a consistent labeling in a simple case. We show how to solve proposed CLPL optimization problems by reducing them to more standard supervised optimization problems in Section 6, and provide several concrete algorithms that can be adapted to our setting, such as support vector machines and boosting. We then proceed to a series of controlled experiments in Section 7, comparing CLPL to several baselines on different data sets. We also apply our framework to a naming task in TV series, where screenplay and closed captions provide ambiguous labels. The code and data used in the paper can be found at: `http://www.vision.grasp.upenn.edu/video`.

2. Related Work

We review here the related work for learning under several forms of weak supervision, as well concrete applications.

2.1 Weakly Supervised Learning

To put the partially-labeled learning problem into perspective, it is useful to lay out several related learning scenarios (see Figure 2), ranging from fully supervised (supervised and multi-label learning), to weakly-supervised (semi-supervised, multi-instance, partially-labeled), to unsupervised.

- In **semi-supervised** learning (Zhu and Goldberg, 2009; Chapelle et al., 2006), the learner has access to a set of labeled examples as well as a set of unlabeled examples.

1. A preliminary version of this work appeared in Cour et al. (2009). Sections 4.2 to 6 present new material, and Sections 7 and 8 contain additional experiments, data sets and comparisons.

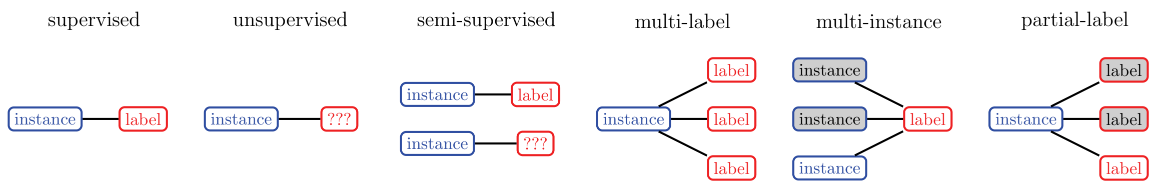


Figure 2: Range of supervision in classification. Training may be: **supervised** (a label is given for each instance), **unsupervised** (no label is given for any instance), **semi-supervised** (labels are given for some instances), **multi-label** (each instance can have multiple labels), **multi-instance** (a label is given for a group of instances where at least one instance in the group has the label), or **partially-labeled** (for each instance, several possible labels are given, only one of which is correct).

- In **multi-label** learning (Boutell et al., 2004; Tsoumakas et al., 2010), each example is assigned multiple labels, all of which can be true.

- In **multi-instance** learning (Dietterich et al., 1997; Andrews and Hofmann, 2004; Viola et al., 2006), examples are not individually labeled but grouped into sets which either contain at least one positive example, or only negative examples. A special case considers the easier scenario where **label proportions** in each bag are known (Kuck and de Freitas, 2005), allowing one to compute convergence bounds on the estimation error of the correct labels (Quadrianto et al., 2009).

- Finally, in our setting of **partially labeled** learning, also called ambiguously labeled learning, each example again is supplied with multiple labels, *only one of which is correct*. A formal definition is given in Section 3.

Clearly, these settings can be combined, for example with multi-instance multi-label learning (MIML) (Zhou and Zhang, 2007), where training instances are associated with not only multiple instances but also multiple labels. Another combination of interest appears in a recent paper building on our previous work (Cour et al., 2009) that addresses the case where sets of instances are ambiguously labeled with candidate labeling sets (Luo and Orabona, 2010).

2.2 Learning From Partially-labeled or Ambiguous Data

There have been several papers that addressed the ambiguous label problem. A number of these use the expectation-maximization algorithm (EM) to estimate the model parameters and the true label (Côme et al., 2008; Ambroise et al., 2001; Vannoorenberghe and Smets, 2005; Jin and Ghahramani, 2002). For example Jin and Ghahramani (2002) use an EM-like algorithm with a discriminative log-linear model to disambiguate correct labels from incorrect ones. Grandvalet and Bengio (2004) add a minimum entropy term to the set of possible label distributions, with a non-convex objective as in the case of (Jin and Ghahramani, 2002). Hullermeier and Beringer (2006) propose several non-parametric, instance-based algorithms for ambiguous learning based on greedy heuristics. These papers only report results on synthetically-created ambiguous labels for data sets such as the UCI repository. Also, the algorithms proposed rely on iterative non-convex learning.

2.3 Images and Captions

A related multi-class setting is common for images with captions: for example, a photograph of a beach with a palm tree and a boat, where object locations are not specified. Duygulu et al. (2002) and Barnard et al. (2003) show that such partial supervision can be sufficient to learn to identify the object locations. The key observation is that while text and images are separately ambiguous, jointly they complement each other. The text, for instance, does not mention obvious appearance properties, but the frequent co-occurrence of a word with a visual element could be an indication of association between the word and a region in the image. Of course, words in the text without correspondences in the image and parts of the image not described in the text are virtually inevitable. The problem of naming image regions can be posed as translation from one language to another. Barnard et al. (2003) address it using a multi-modal extension to mixture of latent Dirichlet allocations.

2.4 Names and Faces

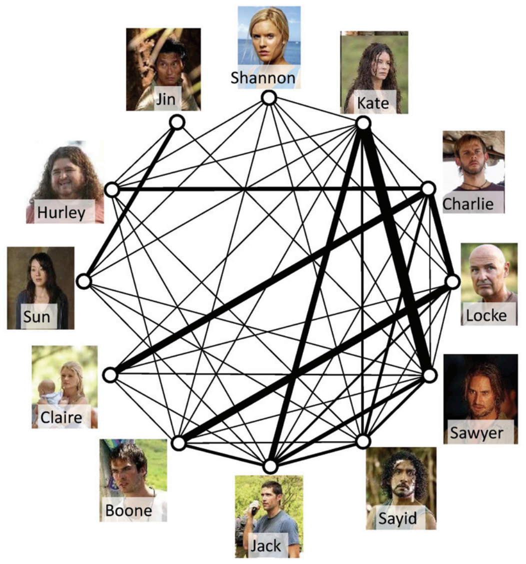
The specific problem of naming faces in images and videos using text sources has been addressed in several works (Satoh et al., 1999; Berg et al., 2004; Gallagher and Chen, 2007; Everingham et al., 2006). There is a vast literature on fully supervised face recognition, which is out of the scope of this paper. Approaches relevant to ours include Berg et al. (2004), which aims at clustering face images obtained by detecting faces from images with captions. Since the name of the depicted people typically appears in the caption, the resulting set of images is ambiguously labeled if more than one name appears in the caption. Moreover, in some cases the correct name may not be included in the set of potential labels for a face. The problem can be solved by using unambiguous images to estimate discriminant coordinates for the entire data set. The images are clustered in this space and the process is iterated. Gallagher and Chen (2007) address the similar problem of retrieval from consumer photo collections, in which several people appear in each image which is labeled with their names. Instead of estimating a prior probability for each individual, the algorithm estimates a prior for groups using the ambiguous labels. Unlike Berg et al. (2004), the method of Gallagher and Chen (2007) does not handle erroneous names in the captions.

2.5 People in Video

In work on video, a wide range of cues has been used to automatically obtain supervised data, including: captions or transcripts (Everingham et al., 2006; Cour et al., 2008; Laptev et al., 2008), sound (Satoh et al., 1999) to obtain the transcript, or clustering based on clothing, face and hair color within scenes to group instances (Ramanan et al., 2007). Most of the methods involve either procedural, iterative reassignment schemes or non-convex optimization.

3. Formulation

In the standard supervised multiclass setting, we have labeled examples $S = \{(x_i, y_i)_{i=1}^m\}$ from an unknown distribution $P(X,Y)$ where $X \in \mathcal{X}$ is the input and $Y \in \{1,\dots,L\}$ is the class label. In the partially supervised setting we investigate, instead of an unambiguous single label per instance we have a set of labels, one of which is the correct label for the instance. We will denote $\mathbf{y}_i = \{y_i\} \cup \mathbf{z}_i$ as the ambiguity set actually observed by the learning algorithm, where $\mathbf{z}_i \subseteq \{1,\dots,L\} \setminus \{y_i\}$ is a set of additional labels, and y_i the latent groundtruth label which we would like to recover. Throughout the paper, we will use boldface to denote sets and uppercase to denote random variables



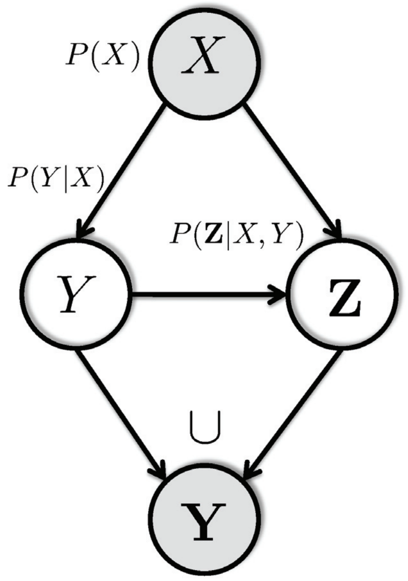


Figure 3: **Left**: Co-occurrence graph of the top characters across 16 episodes of *Lost*. Edge thickness corresponds to the co-occurrence frequency of characters. **Right**: The model of the data generation process: $(X,\mathbf{Y})$ are observed, $(Y,\mathbf{Z})$ are hidden, with $\mathbf{Y} = Y \cup \mathbf{Z}$.

with corresponding lowercase values of random variables. We suppose $X, Y, \mathbf{Z}$ are distributed according to an (unknown) distribution $P(X,Y,\mathbf{Z}) = P(X)P(Y \mid X)P(\mathbf{Z} \mid X,Y)$ (see Figure 3, right), of which we only observe samples of the form $S = \{(x_i, \mathbf{y}_i)_{i=1}^m\} = \{(x_i, \{y_i\} \cup \mathbf{z}_i)_{i=1}^m\}$. (In case X is continuous, $P(X)$ is a density with respect to some underlying measure μ on $\mathcal{X}$, but we will simply refer to the joint $P(X,Y,\mathbf{Z})$ as a distribution.) With the above definitions, $y_i \in \mathbf{y}_i, \mathbf{z}_i \subset \mathbf{y}_i, y_i \notin \mathbf{z}_i$ and $Y \in \mathbf{Y}, \mathbf{Z} \subset \mathbf{Y}, Y \notin \mathbf{Z}$.

Clearly, our setup generalizes the standard semi-supervised setting where some examples are labeled and some are unlabeled: an example is labeled when the corresponding ambiguity set $\mathbf{y}_i$ is a singleton, and unlabeled when $\mathbf{y}_i$ includes all the labels. However, we do not explicitly consider the semi-supervised setting this paper, and our analysis below provides essentially vacuous bounds for the semi-supervised case. Instead, we consider the middle-ground, where all examples are partially labeled as described in our motivating examples and analyze assumptions under which learning can be guaranteed to succeed.

In order to learn from ambiguous data, we must make some assumptions about the distribution $P(\mathbf{Z} \mid X,Y)$. Consider a very simple ambiguity pattern that makes accurate learning impossible: $L = 3$, $|\mathbf{z}_i| = 1$ and label 1 is present in every set $\mathbf{y}_i$, for all i. Then we cannot distinguish between the case where 1 is the true label of every example, and the case where it is not a label of any example. More generally, if two labels always co-occur when present in $\mathbf{y}$, we cannot tell them apart. In order to disallow this case, below we will make an assumption on the distribution $P(\mathbf{Z} \mid X,Y)$ that ensures some diversity in the ambiguity set. This assumption is often satisfied in practice. For example, consider our initial motivation of naming characters in TV shows, where the ambiguity set for any given detected face in a scene is given by the set of characters occurring at some point in that scene. In Figure 3 (left), we show the co-occurrence graph of characters in a season of the TV show *Lost*,

Symbol	Meaning
x, X	observed input value/variable: $x, X \in \mathcal{X}$
y, Y	hidden label value/variable: $y, Y \in \{1, \dots, L\}$
$\mathbf{z}, \mathbf{Z}$	hidden additional label set/variable: $\mathbf{z}, \mathbf{Z} \subseteq \{1, \dots, L\}$
$\mathbf{y}, \mathbf{Y}$	observed label set/variable: $\mathbf{y} = \{y\} \cup \mathbf{z}, \mathbf{Y} = \{Y\} \cup \mathbf{Z}$
$h(x), h(X)$	multiclass classifier mapping $h : \mathcal{X} \mapsto \{1, \dots, L\}$
$\mathcal{L}(h(x), y), \mathcal{L}_A(h(x), \mathbf{y})$	standard and partial $0/1$ loss

Table 1: Summary of notation used.

where the thickness of the edges corresponds to the number of times characters share a scene. This suggests that for most characters, ambiguity sets are diverse and we can expect that the ambiguity degree is small. A more quantitative diagram will be given in Figure 11 (left).

Many formulations of fully-supervised multiclass learning have been proposed based on minimization of convex upper bounds on risk, usually, the expected $0/1$ loss (Zhang, 2004):

$$0/1 \textbf{ loss:} \quad \mathcal{L}(h(x), y) = \mathbb{1}(h(x) \neq y),$$

where $h(x) : \mathcal{X} \mapsto \{1, \dots, L\}$ is a multiclass classifier.

We cannot evaluate the $0/1$ loss using our partially labeled training data. We define a surrogate loss which we can evaluate, and we call ambiguous or partial $0/1$ loss (where A stands for ambiguous):

$$\textbf{Partial } 0/1 \textbf{ loss:} \quad \mathcal{L}_A(h(x), \mathbf{y}) = \mathbb{1}(h(x) \notin \mathbf{y}).$$

3.1 Connection Between Partial and Standard $0/1$ Losses

An obvious observation is that the partial loss is an underestimate of the true loss. However, in the ambiguous learning setting we would like to minimize the true $0/1$ loss, with access only to the partial loss. Therefore we need a way to upper-bound the $0/1$ loss using the partial loss. We first introduce a measure of hardness of learning under ambiguous supervision, which we define as ambiguity degree ε of a distribution $P(X, Y, \mathbf{Z})$:

$$\textbf{Ambiguity degree:} \quad \varepsilon = \sup_{x,y,z: P(x,y)>0, z \in \{1,\dots,L\}} P(z \in \mathbf{Z} \mid X = x, Y = y). \tag{1}$$

In words, ε corresponds to the maximum probability of an extra label z co-occurring with a true label y, over all labels and inputs. Let us consider several extreme cases: When $\varepsilon = 0$, $\mathbf{Z} = \emptyset$ with probability one, and we are back to the standard supervised learning case, with no ambiguity. When $\varepsilon = 1$, some extra label always co-occurs with a true label y on an example x and we cannot tell them apart: no learning is possible for this example. For a fixed ambiguity set size C (i.e., $P(|\mathbf{Z}| = C) = 1$), the smallest possible ambiguity degree is $\varepsilon = C/(L-1)$, achieved for the case where $P(\mathbf{Z} \mid X, Y)$ is uniform over subsets of size C, for which we have $P(z \in \mathbf{Z} \mid X, Y) = C/(L-1)$ for all $z \in \{1, \dots, L\} \backslash \{y\}$. Intuitively, the best case scenario for ambiguous learning corresponds to a distribution with high conditional entropy for $P(\mathbf{Z} \mid X, Y)$.

The following proposition shows we can bound the (unobserved) $0/1$ loss by the (observed) partial loss, allowing us to approximately minimize the standard loss with access only to the partial one. The tightness of the approximation directly relates to the ambiguity degree.

Proposition 1 (Partial loss bound via ambiguity degree ε**)** *For any classifier h and distribution* $P(X,Y,\mathbf{Z})$*, with* $\mathbf{Y} = X \cup \mathbf{Z}$ *and ambiguity degree* ε*:*

$$\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y})] \leq \mathbb{E}_P[\mathcal{L}(h(X),Y)] \leq \frac{1}{1-\varepsilon}\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y})],$$

with the convention $1/0 = +\infty$*. These bounds are tight, and for the second one, for any (rational)* ε*, we can find a number of labels L, a distribution P and classifier h such that equality holds.*

Proof. All proofs appear in Appendix B. ■

3.2 Robustness to Outliers

One potential issue with Proposition 1 is that unlikely (outlier) pairs x,y (with vanishing $P(x,y)$) might force ε to be close to 1, making the bound very loose. We show we can refine the notion of ambiguity degree ε by excluding such pairs.

Definition 2 (ε,δ)***-ambiguous distribution.*** *A distribution* $P(X,Y,\mathbf{Z})$ *is* (ε,δ)*-ambiguous if there exists a subset G of the support of* $P(X,Y)$*,* $G \subseteq \mathcal{X} \times \{1,\dots,L\}$ *with probability mass at least* $1-\delta$*, that is,* $\int_{(x,y)\in G} P(X=x,Y=y)d\mu(x,y) \geq 1-\delta$*, integrated with respect to the appropriate underlying measure* μ *on* $\mathcal{X} \times \{1,\dots,L\}$*, for which*

$$\sup_{(x,y)\in G, z\in\{1,\dots,L\}} P(z \in \mathbf{Z} \mid X=x, Y=y) \leq \varepsilon.$$

Note that in the extreme case $\varepsilon = 0$ corresponds to standard semi-supervised learning, where $1-\delta$-proportion of examples are unambiguously labeled, and δ are (potentially) fully unlabeled. Even though we can accommodate it, semi-supervised learning is not our focus in this paper and our bounds are not well suited for this case.

This definition allows us to bound the $0/1$ loss even in the case when some unlikely set of pairs x,y with probability $\leq \delta$ would make the ambiguity degree large. Suppose we mix an initial distribution with small ambiguity degree, with an outlier distribution with large overall ambiguity degree. The following proposition shows that the bound degrades only by an additive amount, which can be interpreted as a form of robustness to outliers.

Proposition 3 (Partial loss bound via (ε,δ) **)** *For any classifier h and* (ε,δ)*-ambiguous* $P(\mathbf{Z} \mid X,Y)$*,*

$$\mathbb{E}_P[\mathcal{L}(h(X),Y)] \leq \frac{1}{1-\varepsilon}\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y})] + \delta.$$

A visualization of the bounds in Proposition 1 and Proposition 3 is shown in Figure 4.

3.3 Label-specific Recall Bounds

In the types of data from video experiments, we observe that certain subsets of labels are harder to disambiguate than others. We can further tighten our bounds between ambiguous loss and standard

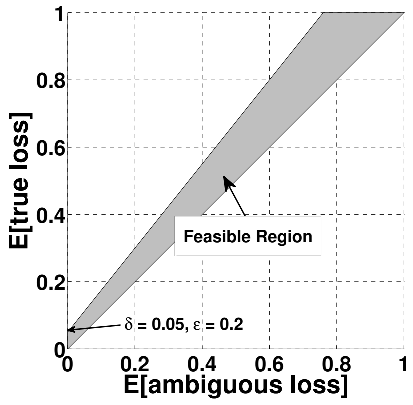


Figure 4: Feasible region for expected ambiguous and true loss, for $\varepsilon = 0.2, \delta = 0.05$.

$0/1$ loss if we consider label-specific information. We define the *label-specific ambiguity degree* ε_a of a distribution (with $a \in \{1,\dots,L\}$) as:

$$\varepsilon_a = \sup_{x,z:P(X=x,Y=a)>0;z\in\{1,\dots,L\}} P(z \in \mathbf{Z} \mid X = x, Y = a).$$

We can show a label-specific analog of Proposition 1:

Proposition 4 (Label-specific partial loss bound) *For any classifier h and distribution $P(X,Y,\mathbf{Z})$ with* label-specific ambiguity degree ε_a *,*

$$\mathbb{E}_P[\mathcal{L}(h(X),Y) \mid Y = a] \leq \frac{1}{1-\varepsilon_a}\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y}) \mid Y = a],$$

where we see that ε_a bounds per-class recall.

These bounds give a strong connection between ambiguous loss and real loss when ε is small. This assumption allows us to approximately minimize the expected real loss by minimizing (an upper bound on) the ambiguous loss, as we propose in the following section.

4. A Convex Learning Formulation

We have not assumed any specific form for our classifier $h(x)$ above. We now focus on a particular family of classifiers, which assigns a score $g_a(x)$ to each label a for a given input x and select the highest scoring label:

$$h(x) = \arg\max_{a\in 1..L} g_a(x).$$

We assume that ties are broken arbitrarily, for example, by selecting the label with smallest index a. We define the vector $g(x) = [g_1(x) \dots g_L(x)]^\top$, with each component $g_a : \mathcal{X} \mapsto \mathbb{R}$ in a function class $\mathcal{G}$. Below, we use a multi-linear function class $\mathcal{G}$ by assuming a feature mapping $\mathbf{f}(x) : \mathcal{X} \mapsto \mathbb{R}^d$ from inputs to d real-valued features and let $g_a(x) = \mathbf{w}_a \cdot \mathbf{f}(x)$, where $\mathbf{w}_a \in \mathbb{R}^d$ is a weight vector for each class, bounded by some norm: $||\mathbf{w}_a||_p \leq B$ for $p = 1,2$.

We build our learning formulation on a simple and general multiclass scheme, frequently used for the fully supervised setting (Crammer and Singer, 2002; Rifkin and Klautau, 2004; Zhang, 2004; Tewari and Bartlett, 2005), that combines convex binary losses $\psi(\cdot) : \mathbb{R} \mapsto \mathbb{R}_+$ on individual components of g to create a multiclass loss. For example, we can use hinge, exponential or logistic loss. In particular, we assume a type of one-against-all scheme for the supervised case:

$$\mathcal{L}_\psi(g(x),y) = \psi(g_y(x)) + \sum_{a \neq y} \psi(-g_a(x)).$$

A classifier $h_g(x)$ is selected by minimizing the empirical loss $\mathcal{L}_\psi$ on the sample $S = \{x_i, y_i\}_{i=1}^m$ (called empirical ψ-risk) over the function class $\mathcal{G}$:

$$\inf_{g \in \mathcal{G}} \mathbb{E}_S[\mathcal{L}_\psi(g(X),Y)] = \inf_{g \in \mathcal{G}} \frac{1}{m} \sum_{i=1}^m \mathcal{L}_\psi(g(x_i),y_i).$$

For the fully supervised case, under appropriate assumptions, this form of the multiclass loss is infinite-sample consistent. This means that a minimizer $\hat{g}$ of ψ-risk achieves optimal $0/1$ risk $\inf_g \mathbb{E}_S[\mathcal{L}_\psi(g(X),Y)] = \inf_g \mathbb{E}_P[\mathcal{L}(g(X),Y)]$ as the number of samples m grows to infinity, provided that the function class $\mathcal{G}$ grows appropriately fast with m to be able to approximate any function from $\mathcal{X}$ to $\mathbb{R}$ and $\psi(u)$ satisfies the following conditions: (1) $\psi(u)$ is convex, (2) bounded below, (3) differentiable and (4) $\psi(u) < \psi(-u)$ when $u > 0$ (Theorem 9 in Zhang (2004)). These conditions are satisfied, for example, for the exponential, logistic and squared hinge loss $\max(0, 1-u)^2$. Below, we construct a loss function for the partially labeled case and consider when the proposed loss is consistent.

4.1 Convex Loss for Partial Labels

In the partially labeled setting, instead of an unambiguous single label y per instance we have a set of labels Y, one of which is the correct label for the instance. We propose the following loss, which we call our *Convex Loss for Partial Labels* (CLPL):

$$\mathcal{L}_\psi(g(x),\mathbf{y}) = \psi\left(\frac{1}{|\mathbf{y}|} \sum_{a \in \mathbf{y}} g_a(x)\right) + \sum_{a \notin \mathbf{y}} \psi(-g_a(x)). \tag{2}$$

Note that if $\mathbf{y}$ is a singleton, the CLPL function reduces to the regular multiclass loss. Otherwise, CLPL will drive up the *average* of the scores of the labels in $\mathbf{y}$. If the score of the correct label is large enough, the other labels in the set do not need to be positive. This tendency alone does not guarantee that the correct label has the *highest* score. However, we show in Proposition 6 that $\mathcal{L}_\psi(g(x),\mathbf{y})$ upperbounds $\mathcal{L}_A(g(x),\mathbf{y})$ whenever $\psi(\cdot)$ is an upper bound on the $0/1$ loss.

Of course, minimizing an upperbound on the loss does not always lead to sensible algorithms. We show next that our loss function is consistent under certain assumptions and offers a tighter upperbound to the ambiguous loss compared to a more straightforward multi-label approach.

4.2 Consistency for Partial Labels

We derive conditions under which the minimizer of the CLPL in Equation 2 with partial labels achieves optimal $0/1$ risk: $\inf_{g \in \mathcal{G}} \mathbb{E}_S[\mathcal{L}_\psi(g(X),\mathbf{Y})] = \inf_{g \in \mathcal{G}} \mathbb{E}_P[\mathcal{L}(g(X),Y)]$ in the limit of infinite data and arbitrarily rich $\mathcal{G}$. Not surprisingly, our loss function is not consistent without making some additional assumptions on $P(\mathbf{Y} \mid X)$ beyond the assumptions for the fully supervised case. Note that the Bayes optimal classifier for $0/1$ loss satisfies the condition $h(x) \in \arg\max_a P(Y = a \mid X = x)$, and may not be unique. First, we require that $\arg\max_a P(Y = a \mid X = x) = \arg\max_a P(a \in \mathbf{Y} \mid X = x)$, since otherwise $\arg\max_a P(Y = a \mid X = x)$ cannot be determined by any algorithm from partial labels $\mathbf{Y}$ without additional information even with an infinite amount of data. Second, we require a simple dominance condition as detailed below and provide a counterexample when this condition does not hold. The dominance relation defined formally below states that when a is the most (or one of the most) likely label given x according to $P(\mathbf{Y} \mid X = x)$ and b is not, $\mathbf{c} \cup \{a\}$ has higher (or equal) probability than $\mathbf{c} \cup \{b\}$ for any set of other labels $\mathbf{c}$.

Proposition 5 (Partial label consistency) *Suppose the following conditions hold:*

- $\psi(\cdot)$ *is differentiable, convex, lower-bounded and non-increasing, with* $\psi'(0) < 0$.
- *When* $P(X = x) > 0$, $\arg\max_{a'} P(Y = a' \mid X = x) = \arg\max_{a'} P(a' \in \mathbf{Y} \mid X = x)$.
- *The following dominance relation holds:* $\forall a \in \arg\max_{a'} P(a' \in \mathbf{Y} \mid X = x)$, $\forall b \notin \arg\max_{a'} P(a' \in \mathbf{Y} \mid X = x)$, $\forall \mathbf{c} \subset \{1, \dots, L\} \backslash \{a, b\}$*:*
$$P(\mathbf{Y} = \mathbf{c} \cup \{a\} \mid X = x) \geq P(\mathbf{Y} = \mathbf{c} \cup \{b\} \mid X = x).$$

Then $\mathcal{L}_\psi(g(x), \mathbf{y})$ *is infinite-sample consistent:*
$$\inf_{g \in \mathcal{G}} \mathbb{E}_S[\mathcal{L}_\psi(g(X), \mathbf{Y})] = \inf_{g \in \mathcal{G}} \mathbb{E}_P[\mathcal{L}(g(X), Y)],$$
as $|S| = m \to \infty$ *and* $\mathcal{G} \to \mathbb{R}^L$ *. As a corollary, consistency is implied when ambiguity degree* $\varepsilon < 1$ *and* $P(Y \mid X)$ *is deterministic, that is,* $P(Y \mid X) = \mathbb{1}(Y = h(X))$ *for some* $h(\cdot)$.

If the dominance relation does not hold, we can find counter-examples where consistency fails. Consider a distribution with a single x with $P(x) > 0$, and let $L = 4$, $P(|\mathbf{Y}| = 2 \mid X = x) = 1$, ψ be the square-hinge loss, and $P(\mathbf{Y} \mid X = x)$ be such that:

		a			
	$250 \cdot P_{ab}$	1	2	3	4
b	1	0	29	44	**0**
	2	29	0	17	**26**
	3	44	17	0	9
	4	**0**	**26**	9	0
	$250 \cdot P_a$	73	72	70	35

Above, the abbreviations are $P_{ab} = P(\mathbf{Y} = \{a, b\} \mid X = x)$ and $P_a = \sum_b P_{ab}$, and the entries that do not satisfy the dominance relation are in bold. We can explicitly compute the minimizer of $\mathcal{L}_\psi$, which is $g = (\frac{1}{2}P_{ab} + \operatorname{diag}(2 - \frac{3}{2}P_a))^{-1}(3P_a - 2) \approx -\begin{bmatrix} 0.6572 & 0.6571 & 0.6736 & 0.8568 \end{bmatrix}$. It satisifes $\arg\max_a g_a = 2$ but $\arg\max_a \sum_b P_{ab} = 1$.

4.3 Comparison to Other Loss Functions

The "naive" partial loss, proposed by Jin and Ghahramani (2002), treats each example as having multiple correct labels, which implies the following loss function
$$\mathcal{L}_\psi^{naive}(g(x), \mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{a \in \mathbf{y}} \psi(g_a(x)) + \sum_{a \notin \mathbf{y}} \psi(-g_a(x)). \tag{3}$$

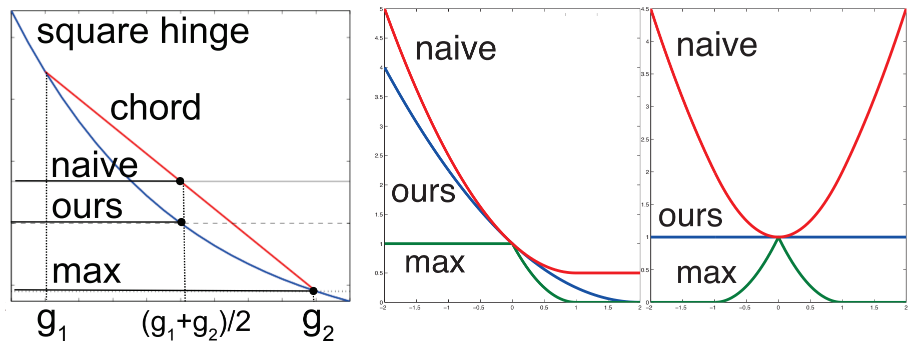


Figure 5: Our loss function in Equation 2 provides a tighter convex upperbound than the naive loss Equation 3 on the non-convex max-loss Equation 4. (**Left**) We show the square hinge ψ (blue) and a chord (red) touching two points g_1, g_2. The horizontal lines correspond to our loss $\psi(\frac{1}{2}(g_1+g_2))$ Equation 2, the max-loss $\psi(\max(g_1,g_2))$, and the naive loss $\frac{1}{2}(\psi(g_1)+\psi(g_2))$ (ignoring negative terms and assuming $\mathbf{y} = \{1,2\}$). (**Middle**) Corresponding losses as we vary $g_1 \in [-2,2]$ (with $g_2 = 0$). (**Right**) Same, with $g_2 = -g_1$.

One reason we expect our loss function to outperform the naive approach is that we obtain a tighter convex upper bound on $\mathcal{L}_A$. Let us also define

$$\mathcal{L}_\psi^{max}(g(x),\mathbf{y}) = \psi\left(\max_{a\in\mathbf{y}} g_a(x)\right) + \sum_{a\notin\mathbf{y}} \psi(-g_a(x)), \tag{4}$$

which is not convex, but is in some sense closer to the desired true loss. The following inequalities are verified for common losses ψ such as square hinge loss, exponential loss, and log loss with proper scaling:

Proposition 6 (Comparison between partial losses) *Under the usual conditions that ψ is a convex, decreasing upper bound of the step function, the following inequalities hold:*

$$2\mathcal{L}_A \le \mathcal{L}_\psi^{max} \le \mathcal{L}_\psi \le \mathcal{L}_\psi^{naive}.$$

The 2nd and 3rd bounds are tight, and the first one is tight provided $\psi(0) = 1$ *and* $\lim_{+\infty}\psi = 0$.

This shows that our CLPL $\mathcal{L}_\psi$ is a tighter approximation to $\mathcal{L}_A$ than $\mathcal{L}_\psi^{naive}$, as illustrated in Figure 5. To gain additional intuition as to why CLPL is better than the naive loss Equation 3: for an input x with ambiguous label set (a,b), CLPL only encourages the *average* of $g_a(x)$ and $g_b(x)$ to be large, allowing the correct score to be positive and the extraneous score to be negative (e.g., $g_a(x) = 2, g_b(x) = -1$). In contrast, the naive model encourages both $g_a(x)$ and $g_b(x)$ to be large.

4.4 Generalization Bounds

To derive concrete generalization bounds on multiclass error for CLPL we define our function class for g. We assume a feature mapping $\mathbf{f}(x) : \mathcal{X} \mapsto \mathbb{R}^d$ from inputs to d real-valued features and let $g_a(x) = \mathbf{w}_a \cdot \mathbf{f}(x)$, where $\mathbf{w}_a \in \mathbb{R}^d$ is a weight vector for each class, bounded by L_2 norm : $||\mathbf{w}_a||_2 \le B$.

We use $\psi(u) = \max(0, 1-u)^p$ (for example hinge loss with $p = 1$, squared hinge loss with $p = 2$). The corresponding margin-based loss is defined via a truncated, rescaled version of ψ:

$$\psi_\gamma(u) = \begin{cases} 1 & \text{if } u \leq 0, \\ (1-u/\gamma)^p & \text{if } 0 < u \leq \gamma, \\ 0 & \text{if } u > \gamma. \end{cases}$$

Proposition 7 (Generalization bound) *For any integer m and any $\eta \in (0,1)$, with probability at least $1-\eta$ over samples $S = \{(x_i, \mathbf{y}_i)\}_{i=1}^m$, for every g in $\mathcal{G}$:*

$$\mathbb{E}_P[\mathcal{L}_A(g(X), \mathbf{Y})] \leq \mathbb{E}_S[\mathcal{L}_{\psi_\gamma}(g(X), \mathbf{Y})] + \frac{4pBL^{5/2}}{c\gamma}\sqrt{\frac{\mathbb{E}_S[||\mathbf{f}(X)||^2]}{m}} + L\sqrt{\frac{8\log(2/\eta)}{m}}.$$

where c is an absolute constant from Lemma 12 in the appendix, $\mathbb{E}_S$ is the sample average and L is the number of labels.

The proof in the appendix uses definition 11 for Rademacher and Gaussian complexity, Lemma 12, Theorem 13 and Theorem 14 from Bartlett and Mendelson (2002), reproduced in the appendix and adapted to our notations for completeness. Using Proposition 7 and Proposition 1, we can derive the following bounds on the true expected 0/1 loss $\mathbb{E}_P[\mathcal{L}(g(X), Y)]$ from purely ambiguous data:

Proposition 8 (Generalization bounds on true loss) *For any distribution ε-ambiguous distribution P, integer m and $\eta \in (0,1)$, with probability at least $1-\eta$ over samples $S = \{(x_i, \mathbf{y}_i)\}_{i=1}^m$, for every $g \in \mathcal{G}$:*

$$\mathbb{E}_P[\mathcal{L}(g(X), Y)] \leq \frac{1}{1-\varepsilon}\left(\mathbb{E}_S[\mathcal{L}_{\psi_\gamma}(g(X), \mathbf{Y})] + \frac{4pBL^{5/2}}{c\gamma}\sqrt{\frac{\mathbb{E}_S[||\mathbf{f}(X)||^2]}{m}} + L\sqrt{\frac{8\log\frac{2}{\eta}}{m}}\right).$$

5. Transductive Analysis

We now turn to the analysis of our *Convex Loss for Partial Labels* (CLPL) in the transductive setting. We show guarantees on disambiguating the labels of instances under fairly reasonable assumptions.

Example 1 *Consider a data set S of two points, x, x', with label sets $\{1,2\}, \{1,3\}$, respectively and suppose that the total number of labels is 3. The objective function is given by:*

$$\psi(\frac{1}{2}(g_1(x) + g_2(x))) + \psi(-g_3(x)) + \psi(\frac{1}{2}(g_1(x') + g_3(x'))) + \psi(-g_2(x')).$$

Suppose the correct labels are $(1,1)$. It is clear that without further assumptions about x and x' we cannot assume that the minimizer of the loss above will predict the right label. However, if $\mathbf{f}(x)$ and $\mathbf{f}(x')$ are close, it should be intuitively clear that we should be able to deduce the label of the two examples is 1.

A natural question is under what conditions on the data will CLPL produce a labeling that is consistent with groundtruth. We provide an analysis under several assumptions.

5.1 Definitions

In the remainder of this section, we denote $y(x)$ (resp. $\mathbf{y}(x)$) as the true label (resp. ambiguous label set) of some $x \in S$, and $\mathbf{z}(x) = \mathbf{y}(x) \backslash \{y(x)\}$. $||\cdot||$ denotes an arbitrary norm, with $||\cdot||^*$ its dual norm. As above, ψ denotes a decreasing upper bound on the step function and g a classifier satisfying: $\forall a$, $||\mathbf{w}_a||^* \leq 1$ (we can easily generalize the remaining propositions to the case where g_a is 1-Lipschitz and $\mathbf{f}$ is the identity). For $x \in S$ and $\eta > 0$, we define $B_\eta(x)$ as the set of neighbors of x that have the same label as x:

$$B_\eta(x) = \{x' \in S \backslash \{x\} : ||f(x') - f(x)|| < \eta, y(x') = y(x)\}.$$

Lemma 9 *Let $x \in S$. If $\mathcal{L}_\psi(g(x), \mathbf{y}(x)) \leq \psi(\eta/2)$ and $\forall a \in \mathbf{z}(x), \exists x' \in B_\eta(x)$ such that $g_a(x') \leq -\eta/2$, then g predicts the correct label for x.*

In other words, g predicts the correct label for x when its loss is sufficiently small, and for each of its ambiguous labels a, we can find a neighbor with same label whose score $g_a(x')$ is small enough. Note that this does not make any assumption on the *nearest* neighbors of x.

Corollary 10 *Let $x \in S$. Suppose $\exists q \geq 0$, $x_1 ... x_q \in B_\eta(x)$ such that $\cap_{i=0..q} \mathbf{z}(x_i) = \emptyset$, $\max_{i=0..q} \mathcal{L}_\psi(g(x_i), \mathbf{y}(x_i)) \leq \psi(\eta/2)$ (with $x_0 := x$). Then g predicts the correct label for x.*

In other words, g predicts the correct label for x if we can find a set of neighbors of the same label with small enough loss, and without any common extra label. This simple condition often arises in our experiments.

6. Algorithms

Our formulation is quite flexible and we can derive many alternative algorithms depending on the choice of the binary loss $\psi(u)$, the regularization, and the optimization method. We can minimize Equation 2 using off-the-shelf binary classification solvers. To do this, we rewrite the two types of terms in Equation 2 as linear combinations of $m \cdot L$ feature vectors. We stack the parameters and features into one vector as follows below, so that $g_a(x) = \mathbf{w}_a \cdot \mathbf{f}(x) = \mathbf{w} \cdot \mathbf{f}(x, a)$:

$$\mathbf{w} = \begin{pmatrix} \mathbf{w}_1 \\ \dots \\ \mathbf{w}_L \end{pmatrix}; \;\; \mathbf{f}(x,a) = \begin{pmatrix} \mathbb{1}(a=1)\mathbf{f}(x) \\ \dots \\ \mathbb{1}(a=L)\mathbf{f}(x) \end{pmatrix}.$$

We also define $\mathbf{f}(x, \mathbf{y})$ to be the average feature vector of the labels in the set $\mathbf{y}$:

$$\mathbf{f}(x, \mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{a \in \mathbf{y}} \mathbf{f}(x, a).$$

With these definitions, we have:

$$\mathcal{L}_\psi(g(x), \mathbf{y}) = \psi(\mathbf{w} \cdot \mathbf{f}(x, \mathbf{y})) + \sum_{a \notin \mathbf{y}} \psi(-\mathbf{w} \cdot \mathbf{f}(x, a)).$$

Then to use a binary classification method to solve CLPL optimization, we simply transform the m partially labelled training examples $S = \{x_i, \mathbf{y}_i\}_{i=1}^m$ into m positive examples $S_+ = \{\mathbf{f}(x_i, \mathbf{y}_i)\}_{i=1}^m$

and $\sum_i L - |\mathbf{y}_i|$ negative examples $S_- = \{\mathbf{f}(x_i, a)\}_{i=1, a \notin \mathbf{y}_i}^m$. Note that the increase in dimension of the features by a factor of L does not significantly affect the running time of most methods since the vectors are sparse. We use the off-the-shelf implementation of binary SVM with squared hinge (Fan et al., 2008) in most of our experiments, where the objective is:

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_i \max(0, 1 - \mathbf{w}\cdot\mathbf{f}(x_i, \mathbf{y}_i))^2 + C\sum_{i, a\notin \mathbf{y}_i} \max(0, 1 + \mathbf{w}\cdot\mathbf{f}(x_i, a))^2.$$

Using hinge loss and L_1 regularization lead to a linear programming formulation, and using L_1 with exponential loss leads naturally to a boosting algorithm. We present (and experiment with) a boosting variant of the algorithm, allowing efficient feature selection, as described in Appendix A. We can also consider the case where the regularization is L_2 and $\mathbf{f}(x) : \mathcal{X} \mapsto \mathbb{R}^d$ is a nonlinear mapping to a high, possibly infinite dimensional space using kernels. In that case, it is simple to show that

$$\mathbf{w} = \sum_i \alpha_i \mathbf{f}(x_i, \mathbf{y}_i) - \sum_{i, a\notin \mathbf{y}_i} \alpha_{i,a}\mathbf{f}(x_i, a),$$

for some set of non-negative α's, where α_i corresponds to the positive example $\mathbf{f}(x_i, \mathbf{y}_i)$, and $\alpha_{i,a}$ corresponds to the negative example $\mathbf{f}(x_i, a)$, for $a \notin \mathbf{y}_i$. Letting $K(x, x') = \mathbf{f}(x)\cdot\mathbf{f}(x')$ be the kernel function, note that $\mathbf{f}(x, a)\cdot\mathbf{f}(x', b) = \mathbb{1}(a = b)K(x, x')$. Hence, we have:

$$\mathbf{w}\cdot\mathbf{f}(x, b) = \sum_{i, a\in \mathbf{y}_i} \frac{\alpha_i}{|\mathbf{y}_i|}\mathbb{1}(a = b)K(x_i, x) - \sum_{i, a\notin \mathbf{y}_i} \alpha_{i,a}\mathbb{1}(a = b)K(x_i, x).$$

This transformation allows us to use kernels with standard off-the-shelf binary SVM implementations.

7. Controlled Partial Labeling Experiments

We first perform a series of controlled experiments to analyze our *Convex Learning from Partial Labels* (CLPL) framework on several data sets, including standard benchmarks from the **UCI repository** (Asuncion and Newman, 2007), a **speaker identification** task from audio extracted from movies, and a **face naming task** from Labeled Faces in the Wild (Huang et al., 2007b). In Section 8 we also consider the challenging task of **naming characters in TV shows** throughout an entire season. In each case the goal is to correctly label faces/speech segments/instances from examples that have multiple potential labels (transductive case), as well as learn a model that can generalize to other unlabeled examples (inductive case).

We analyze the effect on learning of the following factors: distribution of ambiguous labels, size of ambiguous bags, proportion of instances which contain an ambiguous bag, entropy of the ambiguity, distribution of true labels and number of distinct labels. We compare our CLPL approach against a number of **baselines**, including a generative model, a discriminative maximum-entropy model, a naive model, two K-nearest neighbor models, as well as models that ignore the ambiguous bags. We also propose and compare several variations on our cost function. We conclude with a comparative summary, analyzing our approach and the baselines according to several criteria: accuracy, applicability, space/time complexity and running time.

7.1 Baselines

In the experiments, we compare CLPL with the following baselines.

7.1.1 CHANCE BASELINE

We define the *chance* baseline as randomly guessing between the possible ambiguous labels only. Defining the (empirical) average ambiguous size to be $\mathbb{E}_S[|\mathbf{y}|] = \frac{1}{m}\sum_{i=1}^m |\mathbf{y}_i|$, then the expected error from the *chance* baseline is given by $\text{error}_{\text{chance}} = 1 - \frac{1}{\mathbb{E}_S[|\mathbf{y}|]}$.

7.1.2 NAIVE MODEL

We report results on an un-normalized version of the naive model introduced in Equation 3: $\sum_{a\in\mathbf{y}} \psi(g_a(x)) + \sum_{a\notin\mathbf{y}} \psi(-g_a(x))$, but both normalized and un-normalized versions produce very similar results. After training, we predict the label with the highest score (in the transductive setting): $\hat{y} = \arg\max_{a\in\mathbf{y}} g_a(x)$.

7.1.3 IBM MODEL 1

This generative model was originally proposed in Brown et al. (1993) for machine translation, but we can adapt it to the ambiguous label case. In our setting, the conditional probability of observing example $x \in \mathbb{R}^d$ given that its label is a is Gaussian: $x \sim N(\mu_a, \Sigma_a)$. We use the expectation-maximization (EM) algorithm to learn the parameters of the Gaussians (mean μ_a and diagonal covariance matrix $\Sigma_a = diag(\sigma_a)$ for each label).

7.1.4 DISCRIMINATIVE EM

We compare with the model proposed in Jin and Ghahramani (2002), which is a discriminative model with an EM procedure adapted for the ambiguous label setting. The authors minimize the KL divergence between a maximum entropy model P (estimated in the M-step) and a distribution over ambiguous labels $\hat{P}$ (estimated in the E-step):

$$J(\theta, \hat{P}) = \sum_i \sum_{a\in\mathbf{y}} \hat{P}(a \mid x_i) \log\left(\frac{\hat{P}(a \mid x_i)}{P(a \mid x_i, \theta)}\right).$$

7.1.5 K-NEAREST NEIGHBOR

Following Hullermeier and Beringer (2006), we adapt the k-Nearest Neighbor Classifier to the ambiguous label setting as follows:

$$knn(x) = \arg\max_{a\in\mathbf{y}} \sum_{i=1}^{k} w_i \mathbb{1}(a \in \mathbf{y}_i), \tag{5}$$

where x_i is the i^{th} nearest-neighbor of x using Euclidean distance, and w_i are a set of weights. We use two kNN baselines: **kNN** assumes uniform weights $w_i = 1$ (model used in Hullermeier and Beringer, 2006), and **weighted kNN** uses linearly decreasing weights $w_i = k - i + 1$. We use $k = 5$ and break ties randomly as in Hullermeier and Beringer (2006).

7.1.6 SUPERVISED MODELS

Finally we also consider two baselines that *ignore* the ambiguous label setting. The first one, denoted as **supervised model**, removes from Equation 3 the examples with $|\mathbf{y}| > 1$. The second model, denoted as **supervised kNN**, removes from Equation 5 the same examples.

7.2 Data Sets and Feature Description

We describe below the different data sets used to report our experiments. The experiments for automatic naming of characters in TV shows can be found in Section 8. A concise summary is given in Table 2.

Data Set	# instances (m)	# features (d)	# labels (L)	prediction task
UCI: dermatology	366	34	6	disease diagnostic
UCI: ecoli	336	8	8	site prediction
UCI: abalone	4177	8	29	age determination
FIW(10b)	500	50	10 (balanced)	face recognition
FIW(10)	1456	50	10	face recognition
FIW(100)	3011	50	100	face recognition
Lost audio	522	50	19	speaker id
TV+movies	10,000	50	100	face recognition

Table 2: Summary of data sets used in our experiments. The TV+movies experiments are treated in Section 8. Faces in the Wild (1) uses a balanced distribution of labels (first 50 images for the top 10 most frequent people).

7.2.1 UCI DATA SETS

We selected three biology related data sets from the publicly available UCI repository (Asuncion and Newman, 2007): dermatology, ecoli, abalone. As a preprocessing step, each feature was independently scaled to have zero mean and unit variance.

7.2.2 FACES IN THE WILD (FIW)

We experiment with different subsets of the publicly available Labeled Faces in the Wild (Huang et al., 2007a) data set. We use the images registered with funneling (Huang et al., 2007a), and crop out the central part corresponding to the approximate face location, which we resize to 60x90. We project the resulting grayscale patches (treated as 5400x1 vectors) onto a 50-dimensional subspace using PCA.[2] In Table 2, FIW(10b) extracts the first 50 images for each of the top 10 most frequent people (balanced label distribution); FIW(10) extracts *all* images for each of the top 10 most frequent people (heavily unbalanced label distribution, with 530 hits for George Bush and 53 hits for John Ashcroft); FIW(100) extracts up to 100 faces for each of the top 100 most frequent people (again, heavily unbalanced label distribution).

2. We kept the features simple by design; more sophisticated part-based registration and representation would further improve results, as we will see in Section 8.

7.2.3 SPEAKER IDENTIFICATION FROM AUDIO

We also investigate a speaker identification task based on audio in an uncontrolled environment. The audio is extracted from an episode of *Lost* (season 1, episode 5) and is initially completely unaligned. Compared to recorded conversation in a controlled environment, this task is more realistic and very challenging due to a number of factors: background noise, strong variability in tone of voice due to emotions, and people shouting or talking at the same time. We use the Hidden Markov Model Toolkit (HTK) (`http://htk.eng.cam.ac.uk/`) to compute forced alignment (Moreno et al., 1998; Sjölander, 2003), between the closed captions and the audio (given the rough initial estimates from closed caption time stamps, which are often overlapping and contain background noise). After alignment, our data set is composed of 522 utterances (each one corresponding to a closed caption line, with aligned audio and speaker id obtained from aligned screenplay), with 19 different speakers. For each speech segment (typically between 1 and 4 seconds) we extract standard voice processing audio features: pitch (Talkin, 1995), Mel-Frequency Cepstral Coefficients (MFCC) (Mermelstein, 1976), Linear predictive coding (LPC) (Proakis and Manolakis, 1996). This results in a total of 4,000 features, which we normalize to the range $[-1,1]$ and then project onto 50 dimensions using PCA.

7.3 Experimental Setup

For the **inductive experiments**, we split randomly in half the instances into (1) **ambiguously labeled training set**, and (2) **unlabeled testing set**. The ambiguous labels in the training set are generated randomly according to different noise models which we specify in each case. For each method and parameter setting, we report the **average test error rate** over **20 trials** after training the model on the ambiguous train set. We also report the corresponding **standard deviation** as an error bar in the plots. Note, in the inductive setting we consider the test set as unlabeled, thus the classifier votes among *all* possible labels:

$$a^* = h(x) = \arg \max_{a \in \{1..L\}} g_a(x).$$

For the **transductive experiments**, there is no test set; we report the error rate for disambiguating the ambiguous labels (also averaged over 20 trials corresponding to random settings of ambiguous labels). The main differences with the inductive setting are: (1) the model is trained on all instances and tested on the same instances; and (2) the classifier votes only among the ambiguous labels, which is easier:

$$a^* = h(x) = \arg\max_{a \in \mathbf{y}} g_a(x).$$

We compare our CLPL approach (denoted as **mean** in figures, due to the form of the loss) against the **baselines** presented in Section 7.1: Chance, Model 1, Discriminative EM model, k-Nearest Neighbor, weighted k-Nearest Neighbor, Naive model, supervised model, and supervised kNN. Note, in our experiments the Discriminative EM model was much slower to converge than all the other methods, and we only report the first series of experiments with this baseline.

Table 3 summarizes the different settings used in each experiment. We experiment with different noise models for ambiguous bags, parametrized by p, q, ε, see Figure 6. p represents the proportion of examples that are ambiguously labeled. q represents the number of *extra* labels for each ambiguous example. ε represents the degree of ambiguity (defined in 1) for each ambiguous example.[3] We also vary the dimensionality by increasing the number of PCA components from 1 to 200, with half of extra labels added uniformly at random. In Figure 7, we vary the ambiguity size q for three different subsets of Faces in the Wild. We report results on additional data sets in Figure 8.

Experiment	**fig**	**induct.**	**data set**	**parameter**
# of ambiguous bags	6	yes	FIW(10b)	$p \in [0, 0.95], q = 2$
degree of ambiguity	6	yes	FIW(10b)	$p = 1, q = 1, \varepsilon \in [1/(L-1), 1]$
degree of ambiguity	6	**no**	FIW(10b)	$p = 1, q = 1, \varepsilon \in [1/(L-1), 1]$
dimension	6	yes	FIW(10b)	$p = 1, q = \frac{L-1}{2}, d \in [1, .., 200]$
ambiguity size	7	yes	FIW(10b)	$p = 1, q \in [0, 0.9(L-1)]$
ambiguity size	7	yes	FIW(10)	$p = 1, q \in [0, 0.9(L-1)]$
ambiguity size	7	yes	FIW(100)	$p = 1, q \in [0, 0.9(L-1)]$
ambiguity size	8	yes	*Lost* audio	$p = 1, q \in [0, 0.9(L-1)]$
ambiguity size	8	yes	ecoli	$p = 1, q \in [0, 0.9(L-1)]$
ambiguity size	8	yes	derma	$p = 1, q \in [0, 0.9(L-1)]$
ambiguity size	8	yes	abalone	$p = 1, q \in [0, 0.9(L-1)]$

Table 3: Summary of controlled experiments. We experiment with 3 different noise models for ambiguous bags, parametrized by p, q, ε. p represents the proportion of examples that are ambiguously labeled. q represents the number of *extra* labels for each ambiguous example (generated uniformly without replacement). ε represents the degree of ambiguity for each ambiguous example (see definition 1). L is the total number of labels. We also study the effects of data set choice, inductive vs transductive learning, and feature dimensionality.

7.3.1 EXPERIMENTS WITH A BOOSTING VERSION OF CLPL

We also experiment with a boosting version of our CLPL optimization, as presented in Appendix A. Results are shown in Figure 9, comparing our method with kNN and the naive method (also using boosting). Despite the change in learning algorithm and loss function, the trends remain the same.

7.4 Comparative Summary

We can draw several conclusions. Our proposed CLPL model **uniformly outperformed** all baselines in all but one experiment (UCI dermatology data set), where it ranked second closely behind Model 1. In particular CLPL always uniformly outperformed the naive model. The naive model ranks in second. As expected, increasing ambiguity size monotonically affects error rate. We also see that increasing ε significantly affects error, even though the ambiguity size is constant, consistent with our bounds in Section 3.3. We also note that the supervised models defined in Section 7.1.6 (which ignore the ambiguously labeled examples) consistently perform worse than their counterparts adapted for the ambiguous setting. For example, in Figure 6 (Top Left), a model trained with nearly all examples ambiguously labeled ("mean" curve", $p = 95\%$) performs as good as a model which uses 60% of *fully labeled* examples ("supervised" curve, $p = 40\%$). The same holds between the "kNN" curve at $p = 95\%$ and the "supervised kNN" curve at $p = 40\%$.

3. We first choose at random for each label a dominant co-occurring label which is sampled with probability ε; the rest of the labels are sampled uniformly with probability $(1-\varepsilon)/(L-2)$ (there is a single extra label per example).

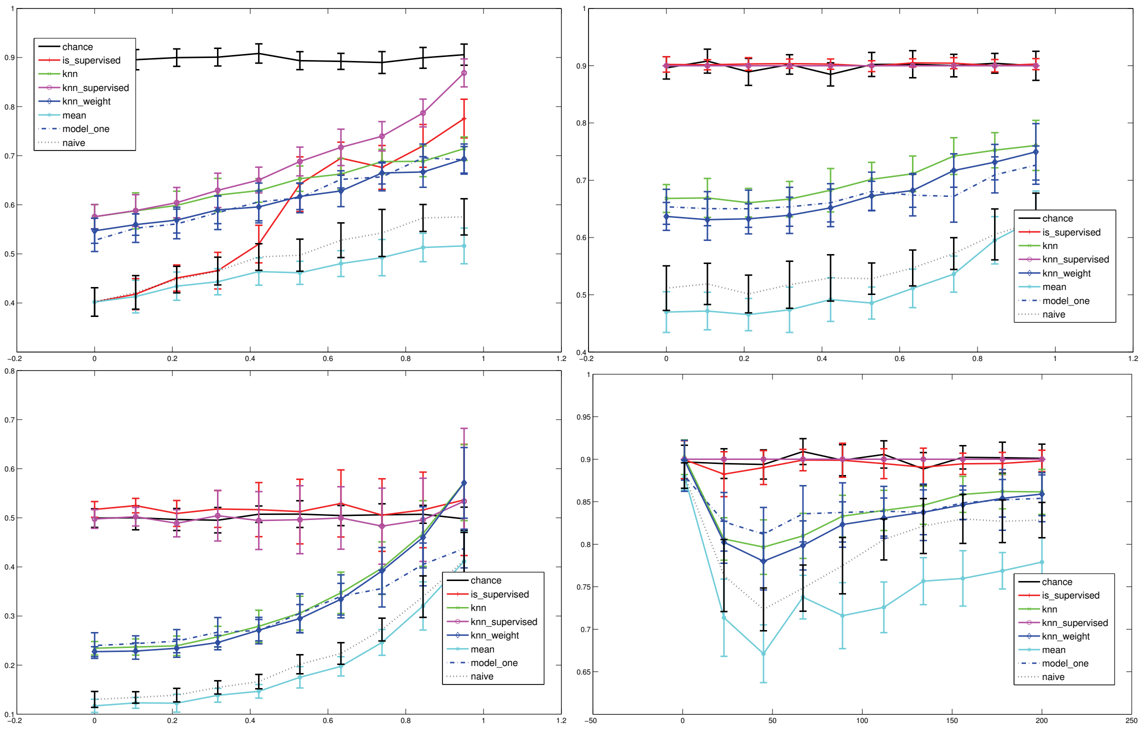


Figure 6: Results on Faces in the Wild in different settings, comparing our proposed CLPL (denoted as **mean**) to several baselines. In each case, we report the average error rate (y-axis) and standard deviation over 20 trials as in Figure 7. **(top left)** increasing proportion of ambiguous bags q, inductive setting. **(top right)** increasing ambiguity degree ε (Equation 1), inductive setting. **(bottom left)** increasing ambiguity degree ε (Equation 1), transductive setting. **(bottom right)** increasing dimensionality, inductive setting.

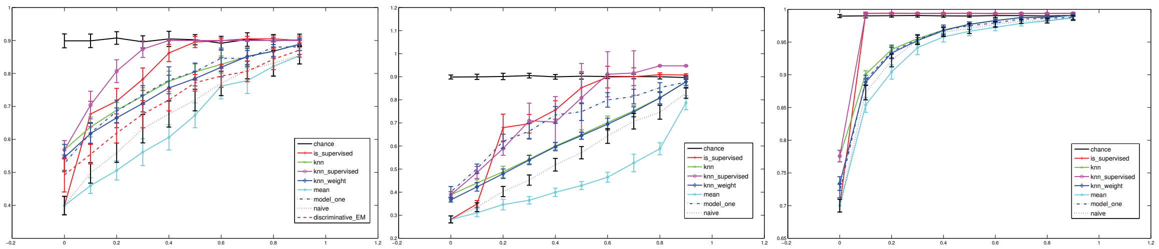


Figure 7: Additional results on Faces in the Wild, obtained by varying the ambiguity size q on the x-axis (inductive case). **Left:** balanced data set using 50 faces for each of the top 10 labels. **Middle:** unbalanced data set using all faces for each of the top 10 labels. **Right:** unbalanced data set using up to 100 faces for each of the top 100 labels.

7.4.1 COMPARISON WITH VARIANTS OF OUR APPROACH

In order to get some intuition on CLPL (Equation 2), which we refer to as the **mean** model in our experiments, we also compare with the following **sum** and **contrastive** alternatives:

$$\mathcal{L}_{\psi}^{\text{sum}}(g(x),\mathbf{y}) = \psi\left(\sum_{a\in\mathbf{y}} g_a(x)\right) + \sum_{a\notin\mathbf{y}} \psi(-g_a(x)), \tag{6}$$

$$\mathcal{L}_{\psi}^{\text{contrastive}}(g(x),\mathbf{y}) = \sum_{a'\notin\mathbf{y}} \psi\left(\frac{1}{|\mathbf{y}|}\sum_{a\in\mathbf{y}} g_a(x) - g_{a'}(x)\right). \tag{7}$$

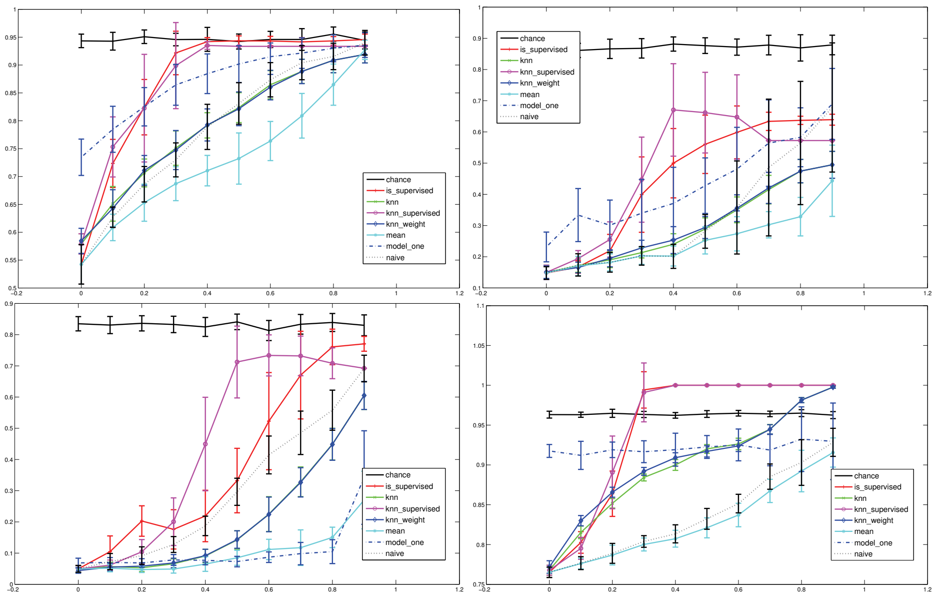


Figure 8: Inductive results on different data sets. In each case, we report the average error rate (y-axis) and standard deviation over 20 trials as in Figure 7. **Top Left:** speaker identification from *Lost* audio. **Top Right:** ecoli data set (UCI). **Bottom Left:** dermatology data set (UCI). **Bottom Right:** abalone data set (UCI).

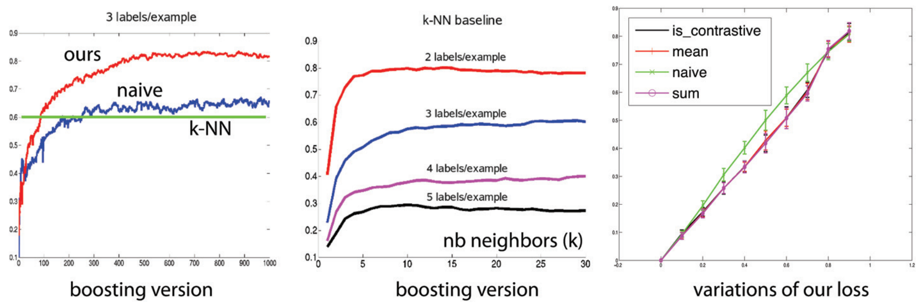


Figure 9: **Left:** We experiment with a boosting version of the ambiguous learning, and compare to a boosting version of the naive baseline (here with ambiguous bags of size 3). We plot *accuracy* vs number of boosting rounds. The green horizontal line corresponds to the best performance (across k) of the k-NN baseline. **Middle:** accuracy of k-NN baseline across k. **Right:** we compare CLPL (labeled **mean**) with two variants defined in Equation 6,Equation 7, along with the naive model (same setting as Figure 6, Top Left).

When $\psi(\cdot)$ is the hinge loss, the mean and sum model are very similar, but this is not the case for strictly convex binary losses. Figure 9 shows that variations on our cost function have *little effect* in the transductive setting. In the inductive setting, other experiments we performed show that the mean and sum version are still very similar, but the contrastive version is worse. In general it seems that models based on minimization of a convex loss function (naive and different versions of our model) usually outperform the other models.

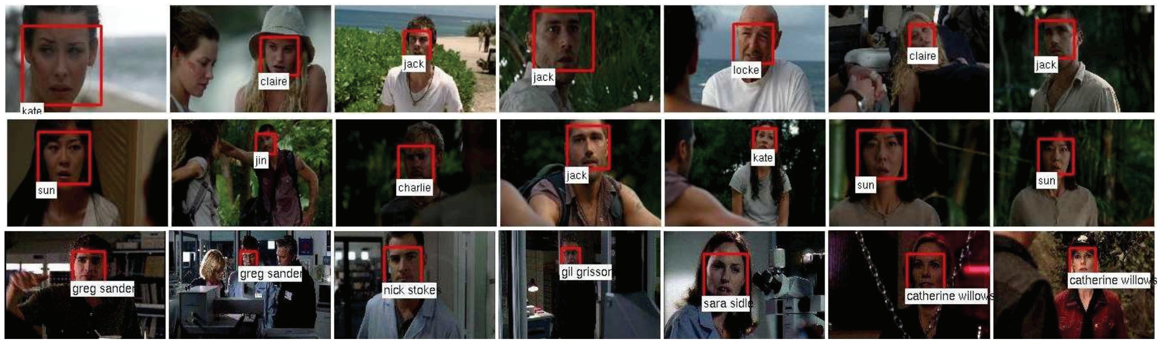


Figure 10: Predictions on *Lost* and *C.S.I.*. Incorrect examples are: row 1, column 3 (truth: Boone); row 2, column 2 (truth: Jack).

8. Experiments with Partially Labeled Faces in Videos

We now return to our introductory motivating example, naming people in TV shows (Figure 1, right). Our goal is to identify characters given ambiguous labels derived from the screenplay. Our data consists of 100 hours of *Lost* and *C.S.I.*, from which we extract ambiguously labeled faces to learn models of common characters. We use the same features, learning algorithm and loss function as in Section 7.2.2. We also explore using additional person- and movie-specific constraints to improve performance. Sample results are shown in Figure 10.

8.1 Data Collection

We adopt the following filtering pipeline to extract face tracks, inspired by Everingham et al. (2006): **(1)** Run the off-the-shelf OpenCV face detector over all frames, searching over rotations and scales. **(2)** Run face part detectors[4] over the face candidates. **(3)** Perform a 2D rigid transform of the parts to a template. **(4)** Compute the score of a candidate face $s(x)$ as the sum of part detector scores plus rigid fit error, normalizing each to weight them equally, and filtering out faces with low score. **(5)** Assign faces to tracks by associating face detections within a shot using normalized cross-correlation in RGB space, and using dynamic programming to group them together into tracks. **(6)** Subsample face tracks to avoid repetitive examples. In the experiments reported here we use the best scoring face in each track, according to $s(x)$.

Concretely, for a particular episode, step (1) finds approximately 100,000 faces, step (4) keeps approximately 10,000 of those, and after subsampling tracks in step (6) we are left with 1000 face examples.

8.2 Ambiguous Label Selection

Screenplays for popular TV series and movies are readily available for free on the web. Given an alignment of the screenplay to frames, we have ambiguous labels for characters in each scene: the set of speakers mentioned at some point in the scene, as shown in Figure 1. Alignment of screenplay to video uses methods presented in Cour et al. (2008) and Everingham et al. (2006), linking closed captions to screenplay.

4. The detectors use boosted cascade classifiers of Haar features for the eyes, nose and mouth.

Lost (#labels, #episodes)	(8,16)	(16,8)†	(16,16)	(32,16)
Naive	14%	18.6%	16.5%	18.5%
ours (CLPL / "mean")	10%	12.6%	14%	17%
ours+constraints	**6%**	n/a	**11%**	**13%**

Table 4: Misclassification rates of different methods on TV show *Lost*. In comparison, for (16,16) the baseline performances are *knn*: 30%; *Model 1*: 44%; *chance*: 53%. †: This column contains results exactly reproducible from our publicly available reference implementation, which can be found at `http://vision.grasp.upenn.edu/video`. For simplicity, this public code does not include a version with extra constraints.

We use the ambiguous sets to select face tracks filtered through our pipeline. We prune scenes which contain characters other than the set we choose to focus on for experiments (top $\{8,16,32\}$ characters), or contain 4 or more characters. This leaves ambiguous bags of size 1, 2 or 3, with an average bag size of 2.13 for *Lost*, and 2.17 for *C.S.I.*.

8.3 Errors in Ambiguous Label Sets

In the TV episodes we considered, we observed that approximately 1% of ambiguous label sets were wrong, in that they didn't contain the ground truth label of the face track. This came from several reasons: presence of a non-english speaking character (Jin Kwon in *Lost*, who speaks Korean) whose dialogue is not transcribed in the closed captions; sudden occurence of an unknown, uncredited character on screen, and finally alignment problems due to large discrepencies between screenplay and closed captions. While this is not a major problem, it becomes so when we consider additional cues (mouth motion, gender) that restrict the ambiguous label set. We will see how we tackle this issue with a robust confidence measure for obtaining good precision recall curves in Section 8.5.

8.4 Results with the Basic System

Now that we have a set of instances (face tracks), feature descriptors for the face track and ambiguous label sets for each face track, we can apply the same method as described in the previous section. We use a transductive setting: we test our method on our ambiguously labeled training set.

The confusion matrix displaying the distribution of ambiguous labels for the top 16 characters in *Lost* is shown in Figure 11 (left). The confusion matrix of our predictions after applying our ambiguous learning algorithm is shown in Figure 11 (right). Our method had the most trouble disambiguating Ethan Rom from Claire Littleton (Ethan Rom only appears in 0.7% of the ambiguous bags, 3 times less then the second least common character) and Liam Pace from Charlie Pace (they are brothers and co-occur frequently, as can be seen in the top figure). The case of Sun Kwon and Jin Kwon is a bit special, as Jin does not speak English in the series and is almost never mentioned in the closed-captions, which creates alignment errors between screenplay and closed captions. These difficulties illustrate some of the interesting challenges in ambiguously labeled data sets. As we can see, the most difficult classes are the ones with which another class is strongly correlated in the ambiguous label confusion matrix. This is consistent with the theoretical bounds we obtained in Section 3.3, which establish a relation between the class specific error rate and class specific degree of ambiguity ε.

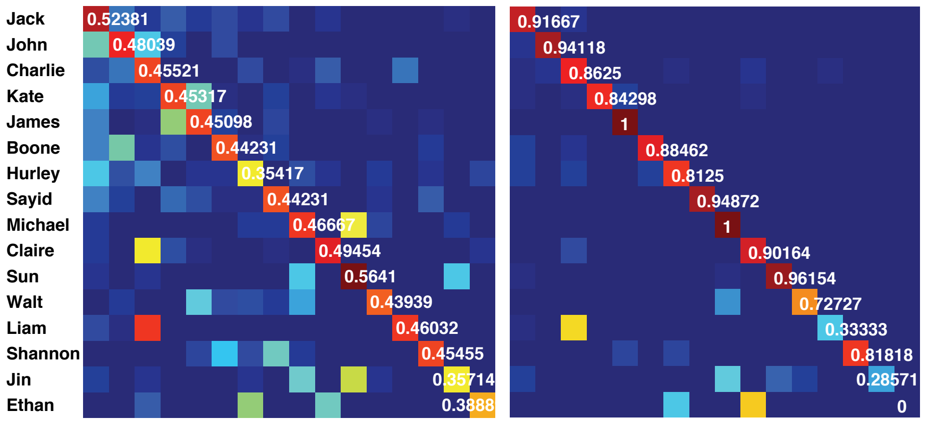


Figure 11: **Left**: Label distribution of top 16 characters in *Lost* (using the standard matlab color map). Element D_{ij} represents the proportion of times class i was seen with class j in the ambiguous bags, and $D\mathbf{1} = \mathbf{1}$. **Right**: Confusion matrix of predictions from Section 8.4. Element A_{ij} represents the proportion of times class i was classified as class j, and $A\mathbf{1} = \mathbf{1}$. Class priors for the most frequent, the median frequency, and the least frequent characters in *Lost* are Jack Shephard, 14%; Hugo Reyes, 6%; Liam Pace 1%.

Quantitative results are shown in Table 4. We measure error according to average 0-1 loss with respect to hand-labeled groundtruth labeled in 8 entire episodes of *Lost*. Our model outperforms all the baselines, and we will further improve results. We now compare several methods to obtain the best possible precision at a given recall, and propose a confidence measure to this end.

8.5 Improved Confidence Measure for Precision-recall Evaluation

We obtain a precision-recall curve using a refusal to predict scheme, as used by Everingham et al. (2006): we report the precision p for the r most confident predictions, varying $r \in [0,1]$. We compare several **confidence measures** based on the classifier scores $g(x)$ and propose a novel one that significantly improves precision-recall, see Figure 12 for results.

1. the **max** and **ratio** confidence measures (as used in Everingham et al., 2006) are defined as:

$$C_{\max}(g(x)) = \max_a g_a(x),$$

$$C_{\text{ratio}}(g(x)) = \max_a \frac{\exp(g_a(x))}{\sum_b \exp(g_b(x))}.$$

2. the **relative** score can be defined as the difference between the best and second best scores over all classifiers $(g_a)_{a\in\{1..L\}}$ (where $a^* = \arg\max_{a\in\{1..L\}} g_a(x)$):

$$C_{\text{rel}}(g(x)) = g_{a^*}(x) - \max_{a\in\{1..L\}-\{a^*\}} g_a(x).$$

3. we can define the **relative-constrained** score as an adaptation to the ambiguous setting; we only consider votes among ambiguous labels $\mathbf{y}$ (where $a^* = \arg\max_{a\in\mathbf{y}} g_a(x)$):

$$C_{\text{rel},\mathbf{y}}(g(x)) = g_{a^*}(x) - \max_{a\in\mathbf{y}-\{a^*\}} g_a(x).$$

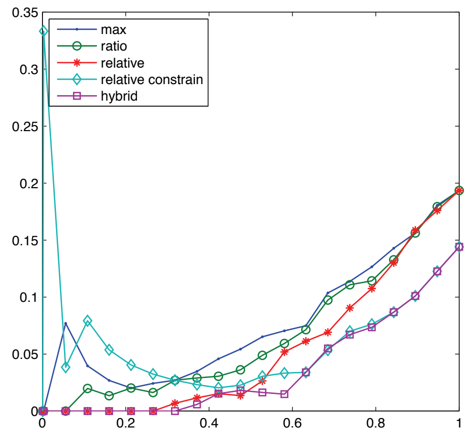


Figure 12: Improved **hybrid** confidence measure for precision-recall evaluation. x axis: recall; y axis: naming error rate for CLPL on 16 episodes of *Lost* (top 16 characters). **max** confidence score performs rather poorly as it ignores other labels. **relative** improves the high precision/low recall region by considering the margin instead. The **relative-constrain** improves the high-recall/low-precision region by only voting among the ambiguous bags, but it suffers in high-precision/low recall region because some ambiguous bags may be erroneous. Our **hybrid** confidence score gets the best of both worlds.

There are some problems with all of those choices, especially in the case where we have some errors in ambiguous label set ($a \notin Y$ for the true label a). This can occur for example if we restrict them with some heuristics to prune down the amount of ambiguity, such as the ones we consider in Section 8.6 (mouth motion cue, gender, etc). At **low recall**, we want maximum precision, therefore we cannot trust too much the heuristic used in relative-constrained confidence. At **high recall**, the errors in the classifier dominate the errors in ambiguous labels, and relative-constrained confidence gives better precision because of the restriction. We introduce a **hybrid** confidence measure that performs well for all recall levels r, interpolating between the two confidence measures:

$$h_r^a(x) = \begin{cases} g_a(x) & \text{if } a \in \mathbf{y}, \\ (1-r)g_a(x) + r\min_b g_b(x) & \text{else.} \end{cases}$$

$$C_r(g(x)) = C_{\text{rel}}(h_r(x)).$$

By design, in the limit $r \to 0, C_r(g(x)) \approx C_{\text{rel}}(g(x))$. In the limit $r \to 1, h_r^a(x)$ is small for $a \notin \mathbf{y}$ and so $C_r(g(x)) \approx C_{\text{rel},\mathbf{y}}(g(x))$.

8.6 Additional Cues

We investigate additional features to further improve the performance of our system: mouth motion, grouping constraints, gender. Final misclassification results are reported in Table 4.

8.6.1 MOUTH MOTION

We use a similar approach to Everingham et al. (2006) to detect mouth motion during dialog and adapt it to our ambiguous label setting.[5] For a face track x with ambiguous label set $\mathbf{y}$ and a temporally overlapping utterance from a speaker $a \in \{1..L\}$ (after aligning screenplay and closed captions), we restrict $\mathbf{y}$ as follows:

$$\mathbf{y} := \begin{cases} \{a\} & \text{if mouth motion,} \\ \mathbf{y} & \text{if refuse to predict or } \mathbf{y} = \{a\}, \\ \mathbf{y} - \{a\} & \text{if absence of mouth motion.} \end{cases}$$

8.6.2 GENDER CONSTRAINTS

We introduce a gender classifier to constrain the ambiguous labels based on predicted gender. The gender classifier is trained on a data set of registered male and female faces, by boosting a set of decision stumps computed on Haar wavelets. We use the average score over a face track output by the gender classifier. We assume known the gender of names mentioned in the screenplay (using automatically extracted cast list from IMDB). We use gender by filtering out the labels that do not match by gender the predicted gender of a face track, if the confidence exceeds a threshold (one for females and one for males are set on a validation data to achieve 90% precision for each direction of the gender prediction). Thus, we modify ambiguous label set $\mathbf{y}$ as:

$$\mathbf{y} := \begin{cases} \mathbf{y} & \text{if gender uncertain,} \\ \mathbf{y} - \{a : a \text{ is male}\} & \text{if gender predicts female,} \\ \mathbf{y} - \{a : a \text{ is female}\} & \text{if gender predicts male.} \end{cases}$$

8.6.3 GROUPING CONSTRAINTS

We propose a very simple must-not-link constraint, which states $y_i \neq y_j$ if face tracks x_i, x_j are in two consecutive shots (modeling alternation of shots, common in dialogs). This constraint is active only when a scene has 2 characters. Unlike the previous constraints, this constraint is incorporated as additional terms in our loss function, as in Yan. et al. (2006). We also propose *groundtruth* grouping constraints for comparison: $y_i = y_j$ for each pair of face tracks x_i, x_j of the same label, and that are separated by at most one shot.

8.7 Ablative Analysis

Figure 13 is an ablative analysis, showing error rate vs recall curves for different sets of cues. We see that the constraints provided by mouth motion help most, followed by gender and link constraints. The best setting (without using groundtruth) combines the former two cues. Also, we notice, once again, a significant performance improvement of our method over the naive method.

8.8 Qualitative Results and Video Demonstration

We show examples with predicted labels and corresponding accuracy, for various characters in *C.S.I.*, see Figure 14. Those results were obtained with the basic system of Section 8.4. Full-frame

5. Motion or absence of motion are detected with a low and high threshold on normalized cross-correlation around mouth regions in consecutive frames.

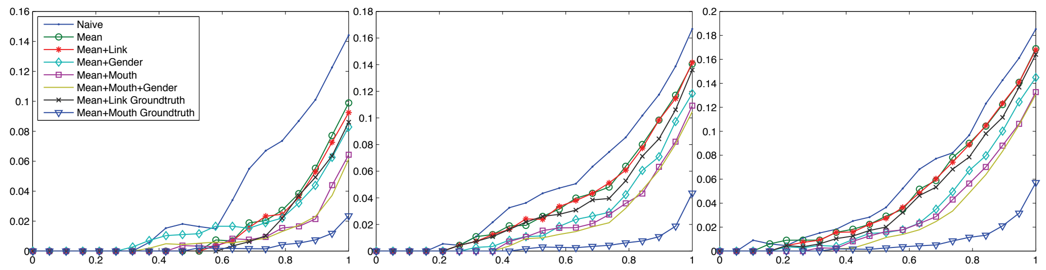


Figure 13: Ablative analysis. *x*-axis: recall; *y*-axis: error rate for character naming across 16 episodes of *Lost*, and the 8, 16, and 32 most common labels (respectively for the left, middle, right plots). We compare our method, **mean**, to the **Naive** model and show the effect of adding several cues to our system. **Link**: simple must-not-link constraints from shot alternation, **Gender**: gender cue for simplification of ambiguous bags; **Mouth**: mouth motion cue for detecting the speaker with synchronous mouth motion; we also consider the combination **Mouth+Gender**, as well as swapping in perfect components such as **Groundtruth link** constraints and **Groundtruth Mouth** motion.

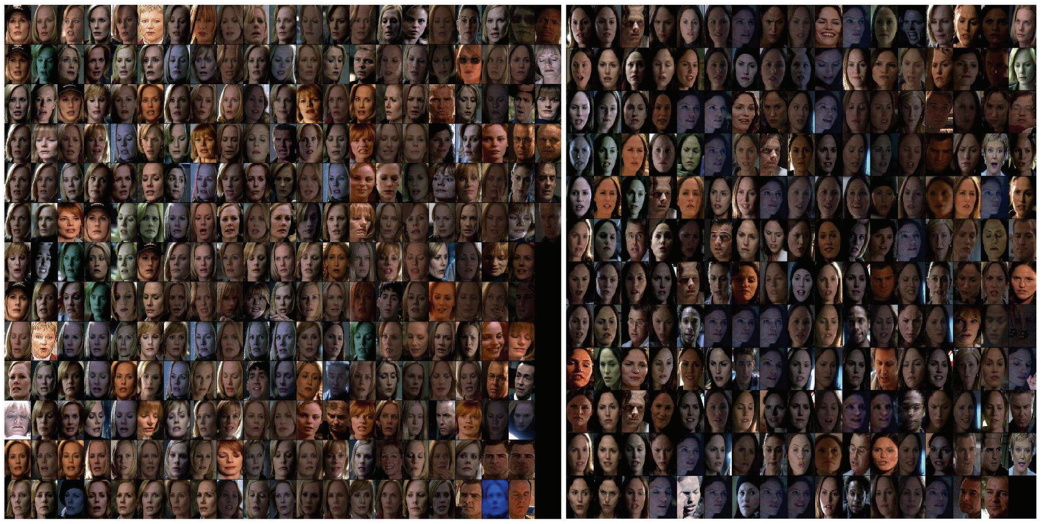


Figure 14: **Left:** Examples classified as Catherine Willows in *C.S.I.* data set using our method (zoom-in for details). Results are sorted by classifier score, in column major format; this explains why most of the errors occur in the last columns. The precision is 85.3%. **Right:** Examples classified as Sara Sidle in *C.S.I.*. The precision is 78.3%.

detections for *Lost* and *C.S.I.* data sets can be seen in Figure 10. We also propagate the predicted labels of our model to all faces in the same face track throughout an episode. Video results of several episodes can be found at the following website `http://www.youtube.com/user/AmbiguousNaming`.

9. Conclusion

We have presented an effective learning approach for partially labeled data, where each instance is tagged with more than one label. Theoretically, under reasonable assumptions on the data distribution, one can show that our algorithm will produce an accurate classifier. We applied our method to two partially-supervised naming tasks: one on still images and one on video from TV series. We also compared to several strong competing algorithms on the same data sets and demonstrated that our algorithm achieves superior performance. We attribute the success of the approach to better modeling of the mutual exclusion between labels than the simple multi-label approach. Moreover, unlike recently published techniques that address similar ambiguously labeled problems, our method does not rely on heuristics and does not suffer from local optima of non-convex methods.

Acknowledgments

The authors would like to thank Jean Ponce and the anonymous reviewers for helpful suggestions. This work was supported by NSF grant 0803538 and a grant from Google. B. Taskar was partially supported by DARPA CSSG, the ONR Young Investigator Award N000141010746 and the Sloan Fellowship.

Appendix A. CLPL with Feature Selection Using Boosting

We derive Algorithm 1 by taking the second order Taylor expansion of the loss $\mathcal{L}_\psi(g(x), \mathbf{y})$, with $\psi(u) = \exp(-u)$. The updates of the algorithm are similar to a multiclass version of Gentleboost (Friedman et al., 2000), but keep a combined weight v_i for the positive example $\mathbf{f}(x_i, \mathbf{y}_i)$ and weights $v_{i,a}$ for the negative examples $\mathbf{f}(x_i, a), a \notin \mathbf{y}_i$.

Algorithm 1 Boosting for CLPL with exponential loss

```
1: Initialize weights: $v_i = 1 \ \ \forall i, \quad v_{i,a} = 1 \ \ \forall i, a \notin \mathbf{y}_i$
2: for $t = 1 \dots T$ do
3:     for $a = 1 \dots L$ do
4:         Fit the parameters of each weak classifier $u(x)$ to minimize the second-order Taylor
           approximation of the cost function with respect to the $a^{th}$ classifier:
           $$\frac{1}{2}\sum_i \left[v_i \cdot \mathbb{1}(a \in \mathbf{y}_i)(u(x_i)/|\mathbf{y}_i| - 1)^2 + v_{i,a} \cdot \mathbb{1}(a \notin \mathbf{y}_i)(u(x_i)+1)^2\right] + constant.$$
5:     end for
6:     Choose the combination of $u, a$ with lowest residual error.
7:     Update $g_a(x) = g_a(x) + u(x)$
8:     for $i = 1 \dots m$ do
9:         if $a \in \mathbf{y}_i$ then
10:            $v_i = v_i \cdot \exp(-u(x_i))$
11:        else
12:            $v_{i,a} = v_{i,a} \cdot \exp(u(x_i))$
13:        end if
14:    end for
15:    Normalize $v$ to sum to 1.
16: end for
```

Appendix B. Proofs

Proof of Proposition 1 (Partial loss bound via ambiguity degree ϵ). The first inequality comes from the fact that $h(x) \notin \mathbf{y} \implies h(x) \neq y$. For the second inequality, fix an $x \in \mathcal{X}$ with $P(X = x) > 0$ and define $\mathbb{E}_P[\cdot \mid x]$ as the expectation with respect to $P(\mathbf{Y} \mid X = x)$.

$$\begin{aligned}\mathbb{E}_P[\mathcal{L}_A(h(x),\mathbf{Y})|x] &= P(h(x) \notin \mathbf{Y} \mid X = x) = P(h(x) \neq Y, h(x) \notin \mathbf{Z} \mid X = x) \\ &= \sum_{a \neq h(x)} P(Y = a \mid X = x)(1 - \underbrace{P(h(x) \in \mathbf{Z} \mid X = x, Y = a)}_{\leq \epsilon \text{ by definition}}) \\ &\geq \sum_{a \neq h(x)} P(Y = a \mid X = x)(1 - \epsilon) = (1-\epsilon)\mathbb{E}_P[\mathcal{L}(h(x),Y)|x].\end{aligned}$$

Hence, $\mathbb{E}_P[\mathcal{L}(h(x),Y)|x] \leq \frac{1}{1-\epsilon}\mathbb{E}_P[\mathcal{L}_A(h(x),\mathbf{Y})|x]$ for any x. We conclude by taking expectation over x.The first inequality is tight: equality can be achieved, for example, when $P(y|x)$ is deterministic, and a perfect classifier h such that for all x, $h(x) = y$. The second inequality is also tight: for example consider the uniform case with a fixed ambiguity size $|\mathbf{z}| = C$ and for all $x, y, z \neq y$, $P(z \in \mathbf{z} \mid X = x, Y = y) = C/(L-1)$. In the proof above (second inequality), the only inequality becomes an equality. In fact, this also shows that for any (rational) ϵ, we can find a number of labels L, a distribution P and a classifer h such that there is equality. ■

Proof of Proposition 3 (Partial loss bound via (ϵ, δ)). We split up the expectation in two parts:

$$\begin{aligned}\mathbb{E}_P[\mathcal{L}(h(X),Y)] &= \mathbb{E}_P[\mathcal{L}(h(X),Y)|(X,Y) \in G](1-\delta) + \mathbb{E}_P[\mathcal{L}(h(X),Y)|(X,Y) \notin G]\delta \\ &\leq \mathbb{E}_P[\mathcal{L}(h(X),Y)|(X,Y) \in G](1-\delta) + \delta \\ &\leq \frac{1}{1-\epsilon}\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y})|(X,Y) \in G](1-\delta) + \delta.\end{aligned}$$

We applied Proposition 1 in the last step. Using a symmetric argument,

$$\begin{aligned}\mathbb{E}_P[\mathcal{L}_A(h(X),Y)] &= \mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y})|(X,Y) \in G](1-\delta) + \mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y})|(X,Y) \notin G]\delta \\ &\geq \mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y})|(X,Y) \in G](1-\delta).\end{aligned}$$

Finally we obtain $\quad \mathbb{E}_P[\mathcal{L}(h(X),Y)] \leq \frac{1}{1-\epsilon}\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y})] + \delta.$ ■

Proof of Proposition 4 (Label-specific partial loss bound). Fix $x \in \mathcal{X}$ such that $P(X = x) > 0$ and $P(Y = a|x) > 0$ and define $\mathbb{E}_P[\cdot \mid x, a]$ as the expectation w.r.t. $P(\mathbf{Z} \mid X = x, Y = a)$. We consider two cases:

a) if $h(x) = a$, $\quad \mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y}) \mid x, a] = P(h(x) \neq a, h(x) \notin \mathbf{y} \mid X = x, Y = a) = 0.$

b) if $h(x) \neq a$, $\quad \mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y}) \mid x, a] = P(h(x) \notin \mathbf{Z} \mid X = x, Y = a)$
$= 1 - P(h(x) \in \mathbf{Z} \mid X = x, Y = a) \geq 1 - \epsilon_a.$

We conclude by taking expectation over x:

$$\begin{aligned}\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y}) \mid Y = a] &= P(h(X) = a|Y = a)\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y}) \mid h(X) = a, Y = a] \\ &\quad + P(h(X) \neq a|Y = a)\mathbb{E}_P[\mathcal{L}_A(h(X),\mathbf{Y}) \mid h(X) \neq a, Y = a] \\ &\geq 0 + P(h(X) \neq a \mid Y = a) \cdot (1 - \epsilon_a) \\ &= (1-\epsilon_a) \cdot \mathbb{E}_P[\mathcal{L}(h(X),Y) \mid Y = a].\end{aligned}$$

■

Proof of Proposition 5 (Partial label consistency). We assume $g(x)$ is found by minimizing over an appropriately rich sequence of function classes (Tewari and Bartlett, 2005), in our case, as $m \to \infty$, $\mathcal{G} \to \mathbb{R}^L$. Hence we can focus on analysis for a fixed x (with $P(X = x) > 0$), writing $g_a = g_a(x)$, and for any set $\mathbf{c} \subseteq \{1, \dots, L\}$, $g_\mathbf{c} = \sum_{a \in \mathbf{c}} g_a / |\mathbf{c}|$ and $P_\mathbf{c} = P(\mathbf{Y} = \mathbf{c} | X = x)$. We also write $P_a = P(a \in \mathbf{Y} | X = x)$ for any label a, and use shorthand $P_{\mathbf{c},a} = P_{\mathbf{c} \cup \{a\}}$ and $g_{\mathbf{c},a} = g_{\mathbf{c} \cup \{a\}}$. We have:

$$\mathcal{L}_\psi(g) = \sum_\mathbf{c} P_\mathbf{c} \cdot \left(\psi(g_\mathbf{c}) + \sum_{a \notin \mathbf{c}} \psi(-g_a) \right).$$

Note that the derivative $\psi'(\cdot)$ exists and is non-positive and non-decreasing by assumption and $\psi'(z) < 0$ for $z \leq 0$. The assumptions imply that $\psi(-\infty) \to \infty$, so assuming that $P_a < 1$, minimizers are upper-bounded: $g_a < \infty$. The case of $P_a = 0$ leads to $g_a \to -\infty$ and it can be ignored without loss of generality, so we can assume that optimal g is bounded for fixed p with $0 < P_a < 1$.

Taking the derivative of the loss with respect to g_a and setting to 0, we have the first order optimality conditions:

$$\frac{\partial \mathcal{L}_\psi(g)}{\partial g_a} = \sum_{\mathbf{c} : a \notin \mathbf{c}} \frac{P_{\mathbf{c},a} \psi'(g_{\mathbf{c},a})}{|\mathbf{c}| + 1} - (1 - P_a)\psi'(-g_a) = 0.$$

Now suppose (for contradiction) that at a minimizer g, $b \in \arg\max_{a'} g_{a'}$ but $P_a > P_b$ for some $a \in \arg\max_{a'} P_{a'}$. Subtracting the optimality conditions for a, b from each other, we get

$$\sum_{\mathbf{c} : a,b \notin \mathbf{c}} \frac{P_{\mathbf{c},a} \psi'(g_{\mathbf{c},a}) - P_{\mathbf{c},b} \psi'(g_{\mathbf{c},b})}{|\mathbf{c}| + 1} = (1 - P_a)\psi'(-g_a) - (1 - P_b)\psi'(-g_b).$$

Since $g_a \leq g_b$, $\psi'(g_{\mathbf{c},a}) \leq \psi'(g_{\mathbf{c},b})$ and $\psi'(-g_a) \geq \psi'(-g_b)$. Plugging in on both sides:

$$\sum_{\mathbf{c} : a,b \notin \mathbf{c}} \frac{(P_{\mathbf{c},a} - P_{\mathbf{c},b}) \psi'(g_{\mathbf{c},b})}{|\mathbf{c}| + 1} \geq (P_b - P_a)\psi'(-g_b).$$

By dominance assumption, $(P_{\mathbf{c},a} - P_{\mathbf{c},b}) \geq 0$ and since $(P_b - P_a) < 0$ and $\psi'(\cdot)$ is non-positive, the only possibility of the inequality holding is that $\psi'(-g_b) = 0$ (which implies $g_b > 0$) and $(P_{\mathbf{c},a} - P_{\mathbf{c},b})\psi'(g_{\mathbf{c},a}) = 0$ for all $\mathbf{c}$. But $(P_b - P_a) < 0$ implies that there exists a subset $\mathbf{c}$ such that $(P_{\mathbf{c},a} - P_{\mathbf{c},b}) > 0$. Since $b \in \arg\max g$, $g_{\mathbf{c},b} \leq g_b$, so $g_{\mathbf{c},b} \leq 0$, hence $\psi'(g_{\mathbf{c},b}) < 0$, a contradiction.

When $P(y \mid x)$ is deterministic, let $P(y|x) = \mathbb{1}(y = a)$. Clearly, if $\varepsilon < 1$, then $a = \arg\max_{a'} P_{a'}$ and $P_a = 1 > P_{a'}, \forall a' \neq a$. Then the minimizer g satisfies either (1) $g_a \to \infty$ (this happens if $\psi'(\cdot) < 0$ for finite arguments) while $g_{a'}$ are finite because of $(1 - P_{a'})\psi(-g_{a'})$ terms in the objective or (2) g is finite and the proof above applies since dominance holds: $P_{\mathbf{c},b} = 0$ if $a \notin \mathbf{c}$, so we can apply the theorem.

■

Proof of Proposition 6 (Comparison between partial losses). Let $a^* = \arg\max_{a \in 1..L} g_a(x)$. For the first inequality, if $a^* \in \mathbf{y}$, $\mathcal{L}_\psi^{max}(g(x), \mathbf{y}) \geq 0 = 2\mathcal{L}_A(g(x), \mathbf{y})$. Otherwise $a^* \notin \mathbf{y}$:

$$\begin{aligned}\mathcal{L}_\psi^{max}(g(x), \mathbf{y}) &\geq \psi(\max_{a \in \mathbf{y}} g_a(x)) + \psi(-g_{a^*}(x)) \geq \psi(g_{a^*}(x)) + \psi(-g_{a^*}(x)) \\ &\geq 2\psi\left(\frac{g_{a^*}(x) - g_{a^*}(x)}{2}\right) = 2\psi(0) \geq 2\mathcal{L}_A(g(x), \mathbf{y}).\end{aligned}$$

The second inequality comes from the fact that

$$\max_{a \in \mathbf{y}} g_a(x) \geq \frac{1}{|\mathbf{y}|} \sum_{a \in \mathbf{y}} g_a(x).$$

For the third inequality, we use the convexity of ψ:

$$\psi\left(\frac{1}{|\mathbf{y}|} \sum_{a \in \mathbf{y}} g_a(x)\right) \leq \frac{1}{|\mathbf{y}|} \sum_{a \in \mathbf{y}} \psi(g_a(x)).$$

For the tightness proof: When $g_a(x) = \mathit{constant}$ over $a \in \mathbf{y}$, we have

$$\psi\left(\max_{a \in \mathbf{y}} g_a(x)\right) = \psi\left(\frac{1}{|\mathbf{y}|} \sum_{a \in \mathbf{y}} g_a(x)\right) = \frac{1}{|\mathbf{y}|} \sum_{a \in \mathbf{y}} \psi\left(g_a(x)\right),$$

implying $\mathcal{L}_\psi^{max}(g(x), \mathbf{y}) = \mathcal{L}_\psi(g(x), \mathbf{y}) = \mathcal{L}_\psi^{naive}(g(x), \mathbf{y})$.

As for the first inequality, we provide a sequence $g^{(n)}$ that verifies equality in the limit: let $g_a^{(n)}(x) = -1/n$ if $a \in \mathbf{y}$, $g_b^{(n)}(x) = 0$ for some $b \notin \mathbf{y}$, and $g_c^{(n)}(x) = -n$ for all $c \notin \mathbf{y}, c \neq b$. Then provided $\psi(0) = 1$ and $\lim_{u \to \infty} \psi(u) = 0$, we have $\lim_{n \to +\infty} \mathcal{L}_\psi^{max}(g^{(n)}(x), \mathbf{y}) = 2$ and for all n, $\mathcal{L}_A(g^{(n)}(x), \mathbf{y}) = 1$. ■

Proof of Proposition 7 (Generalization bounds). The proof uses Definition 11 for Rademacher and Gaussian complexity, Lemma 12, Theorem 13 and Theorem 14 from Bartlett and Mendelson (2002), reproduced below and adapted to our notations for completeness. We apply Theorem 13 with $L := \frac{1}{L}\mathcal{L}_A$, $\phi := \frac{1}{L}\mathcal{L}_{\psi_\gamma}$:

$$\frac{1}{L}\mathbb{E}_P[\mathcal{L}_A(g(X), \mathbf{Y})] \leq \frac{1}{L}\mathbb{E}_S[\mathcal{L}_{\psi_\gamma}(g(X), \mathbf{Y})] + R_m(\phi \circ \mathcal{G}) + \sqrt{\frac{8\log(2/\eta)}{m}}.$$

From Lemma 12, $R_m(\phi \circ \mathcal{G}) \leq \frac{1}{c} G_m(\phi \circ \mathcal{G})$. From Theorem 14, $G_m(\phi \circ \mathcal{G}) \leq 2\lambda \sum_{a=1}^{L} \hat{G}_m(\mathcal{G}_a)$. Let (ν_i) be m independent standard normal random variables.

$$\begin{aligned}
\hat{G}_m(\mathcal{G}_a) &= \mathbb{E}_\nu \left[\sup_{g_a \in \mathcal{G}_a} \frac{2}{m} \sum_i \nu_i g_a(x_i) \mid S \right] = \frac{2}{m} \mathbb{E}_\nu \left[\sup_{||\mathbf{w}_a|| \leq B} \mathbf{w}_a \cdot \sum_i \nu_i \mathbf{f}(x_i) \mid S \right] \\
&= \frac{2B}{m} \mathbb{E}_\nu \left[|| \sum_i \nu_i \mathbf{f}(x_i) || \mid S \right] = \frac{2B}{m} \mathbb{E}_\nu \left[\sqrt{\sum_{ij} \nu_i \nu_j \mathbf{f}(x_i)^T \mathbf{f}(x_j)} \mid S \right] \\
&\leq \frac{2B}{m} \sqrt{\mathbb{E}_\nu \left[\sum_{ij} \nu_i \nu_j \mathbf{f}(x_i)^T \mathbf{f}(x_j) \mid S \right]} = \frac{2B}{m} \sqrt{\sum_i \mathbb{E}_\nu \left[\nu_i^2 ||\mathbf{f}(x_i)||^2 \mid S \right]} \\
&= \frac{2B}{m} \sqrt{\sum_i ||\mathbf{f}(x_i)||^2}.
\end{aligned}$$

Putting everything together, $R_m(\phi \circ \mathcal{G}) \leq \frac{2\lambda L}{c} \hat{G}_m(\mathcal{G}_a) \leq \frac{4\lambda LB}{mc} \sqrt{\sum_i ||\mathbf{f}(x_i)||^2}$ and:

$$\mathbb{E}_P[\mathcal{L}_A(g(X), Y)] \leq \mathbb{E}_S[\mathcal{L}_{\psi_\gamma}(g(X), \mathbf{Y})] + \frac{4\lambda BL^2}{mc} \sqrt{\sum_i ||\mathbf{f}(x_i)||^2} + L\sqrt{\frac{8\log(2/\eta)}{m}}.$$

The Lipschitz constant from 14 can be computed as $\lambda := \frac{p}{\gamma}\sqrt{L}$, using the Lipschitz constant of the scalar function ψ_γ, which is $\frac{p}{\gamma}$, and the fact that $||g(x)||_1 \leq \sqrt{L}||g(x)||_2$. ■

Definition 11 (Definition 2 from Bartlett and Mendelson (2002)) *Let μ be a probability distribution on a set $\mathcal{X}$ and suppose that $S = \{x_i\}_{i=1}^m$ are independent samples sampled from μ. Let $\mathcal{G}$ be a class of functions $\mathcal{X} \to \mathbb{R}$. Define the random variables*

$$\begin{aligned}
\hat{R}_m(\mathcal{F}) &= \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \frac{2}{m} \sum_i \sigma_i f(x_i) \mid S \right], \\
\hat{G}_m(\mathcal{F}) &= \mathbb{E}_\nu \left[\sup_{f \in \mathcal{F}} \frac{2}{m} \sum_i \nu_i f(x_i) \mid S \right],
\end{aligned}$$

where (σ_i) are m independent uniform $\{\pm 1\}$-valued random variables and (ν_i) are m independent standard normal random variables. Then the Rademacher (resp. Gaussian) complexity of $\mathcal{G}$ is $R_m(\mathcal{F}) = \mathbb{E}_S[\hat{R}_m(\mathcal{F})]$ (resp. $G_m(\mathcal{F}) = \mathbb{E}_S[\hat{F}_m(\mathcal{F})]$).

$R_m(\mathcal{F})$ and $G_m(\mathcal{F})$ quantify how much can a $f \in \mathcal{F}$ be correlated with a noise sequence of length m.

Lemma 12 (Lemma 4 from Bartlett and Mendelson (2002)) *There are absolute constants c and C such that for every class $\mathcal{G}$ and every integer m,*

$$cR_m(\mathcal{G}) \leq G_m(\mathcal{G}) \leq C \log m R_m(\mathcal{G}).$$

Theorem 13 (Theorem 8 from Bartlett and Mendelson (2002)) *Consider a loss function $\mathcal{L} : A \times \mathcal{Y} \mapsto [0,1]$ and a dominating cost function $\phi : A \times \mathcal{Y} \to [0,1]$, where A is an arbitrary output space. Let $\mathcal{G}$ be a class of functions mapping from $\mathcal{X}$ to A and let $S = \{(x_i, y_i)\}_{i=1}^m$ be independently selected according to the probability measure P. Define $\phi \circ \mathcal{G} = \{(x,y) \mapsto \phi(g(x),y) - \phi(0,y) : g \in \mathcal{G}\}$. Then, for any integer m and any $\eta \in (0,1)$, with probability at least $1-\eta$ over samples of length m, $\forall g \in \mathcal{G}$:*

$$\mathbb{E}_P[\mathcal{L}(g(X),Y)] \leq \mathbb{E}_S \phi(g(X),y) + R_m(\phi \circ \mathcal{G}) + \sqrt{\frac{8\log(2/\eta)}{m}}.$$

Theorem 14 (Theorem 14 from Bartlett and Mendelson (2002)) *Let $A = \mathbb{R}^L$, and let $\mathcal{G}$ be a class of functions mapping $\mathcal{X}$ to A. Suppose that there are real-valued classes $\mathcal{G}_1, ..., \mathcal{G}_L$ such that $\mathcal{G}$ is a subset of their direct sum. Assume further that $\phi : A \times \mathcal{Y} \to \mathbb{R}$ is such that, for all $y \in \mathcal{Y}$, $\phi(\cdot, y)$ is a Lipschitz function (with respect to Euclidean distance on A) with constant λ which passes through the origin and is uniformly bounded. For $g \in \mathcal{G}$, define $\phi \circ g$ as the mapping $(x,y) \mapsto \phi(g(x),y)$. Then, for every integer m and every sample $S = \{(x_i, y_i)\}_{i=1}^m$,*

$$\hat{G}_m(\phi \circ \mathcal{G}) \leq 2\lambda \sum_{a=1}^{L} \hat{G}_m(\mathcal{G}_a),$$

where $\hat{G}_m(\phi \circ \mathcal{G})$ are the Gaussian averages of $\phi \circ \mathcal{G}$ with respect to the sample $\{(x_i, y_i)\}_{i=1}^m$ and $\hat{G}_m(\mathcal{G}_a)$ are the Gaussian averages of $\mathcal{G}_a$ with respect to the sample $\{x_i\}_{i=1}^m$.

Proof of Proposition 8 (Generalization bounds on true loss). This follows from Propositions 7 and 1. ■

Proof of Lemma 9. Let us write $\mathbf{z} = \mathbf{z}(x), y = y(x), \mathbf{y} = \mathbf{y}(x)$.

- Let $a \in \mathbf{z}$. By hypothesis, $\exists x' \in B_\eta(x) : g_a(x') \leq -\frac{\eta}{2}$. By definition of $B_\eta(x)$,

  $$g_a(x) = g_a(x') + \mathbf{w}_a \cdot (\mathbf{f}(x) - \mathbf{f}(x')) \leq g_a(x') + ||\mathbf{w}_a||^* \eta \leq g_a(x') + \eta \leq \frac{\eta}{2}.$$

  In fact, we also have $g_a(x) < \frac{\eta}{2}$, by considering two cases ($\mathbf{w}_a = 0$ or $\mathbf{w}_a \neq 0$) and using the fact that $||\mathbf{f}(x) - \mathbf{f}(x')|| < \eta$.

- Let $a \notin \mathbf{y}$. Since $\mathcal{L}_\psi(g(x), \mathbf{y}) \leq \psi(\eta/2)$ and each term is nonnegative, we have:

  $$\psi(-g_a(x)) \leq \psi(\frac{\eta}{2}) \implies g_a(x) \leq -\frac{\eta}{2}.$$

- Let $a = y$. $\mathcal{L}_\psi(g(x), \mathbf{y}) \leq \psi(\eta/2)$ also implies the following:

  $$\begin{array}{rcl} & \psi\left(\frac{1}{|\mathbf{y}|}\sum_{b \in \mathbf{y}} g_b(x)\right) & \leq \psi(\frac{\eta}{2}) \\ \implies & \frac{1}{|\mathbf{y}|}\sum_{b \in \mathbf{y}} g_b(x) & \geq \frac{\eta}{2} \\ \implies & g_y(x) & \geq \frac{|\mathbf{y}|\eta}{2} - \sum_{b \in \mathbf{z}} g_b(x) \\ & & > \frac{|\mathbf{y}|\eta}{2} - \frac{|\mathbf{z}|\eta}{2} = \frac{\eta}{2}. \end{array}$$

Finally, $\forall a \neq y, g_a(x) < g_y(x)$ and g classifies x correctly. ■

Proof of corollary 10. Let $a \in \mathbf{z}(x)$, by the empty intersection hypothesis, $\exists i \geq 1 : a \notin \mathbf{z}(x_i)$ and since $y(x_i) = y(x)$ and $a \neq y(x)$ we also have $a \notin \mathbf{y}(x_i)$. Since $\mathcal{L}_\psi(g(x_i), \mathbf{y}(x_i) \leq \psi(\eta/2)$, we have $g_a(x_i) \leq -\frac{\eta}{2}$, as in the previous proof. We can apply Lemma 9 (with $x' = x_i$). ■

References

C. Ambroise, T. Denoeux, G. Govaert, and P. Smets. Learning from an imprecise teacher: Probabilistic and evidential approaches. In *Applied Stochastic Models and Data Analysis*, volume 1, pages 100–105, 2001.

S. Andrews and T. Hofmann. Multiple instance learning via disjunctive programming boosting. In *Advances in Neural Information Processing Systems*, 2004.

A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

K. Barnard, P. Duygulu, D.A. Forsyth, N. de Freitas, D.M. Blei, and M.I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

T.L. Berg, A.C. Berg, J.Edwards, M.Maire, R.White, Y.W. Teh, E.G. Learned-Miller, and D.A. Forsyth. Names and faces in the news. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 848–854, 2004.

M.R. Boutell, J. Luo, X. Shen, and C.M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

P. E. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311, 1993.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 2006.

E. Côme, L. Oukhellou, T. Denœux, and P. Aknin. Mixture model estimation with soft labels. *International Conference on Soft Methods in Probability and Statistics*, 2008.

T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar. Movie/script: Alignment and parsing of video and text transcription. In *Proc. European Conference on Computer Vision*, 2008.

T. Cour, B. Sapp, C. Jordan, and B. Taskar. Learning from ambiguously labeled images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

T.G. Dietterich, R.H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

P. Duygulu, K. Barnard, J.F.G. de Freitas, and D.A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proc. European Conference on Computer Vision*, pages 97–112, 2002.

M. Everingham, J. Sivic, and A. Zisserman. Hello! My name is... Buffy – automatic naming of characters in tv video. In *British Machine Vision Conference*, 2006.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.

A.C. Gallagher and T. Chen. Using group prior to identify people in consumer images. In *CVPR Workshop on Semantic Learning Applications in Multimedia*, 2007.

Y. Grandvalet and Y. Bengio. Learning from partial labels with minimum entropy. *Centre interuniversitaire de recherche en analyse des organisations (CIRANO)*, 2004.

G.B. Huang, V. Jain, and E. Learned-Miller. Unsupervised joint alignment of complex images. In *Proc. International Conference on Computer Vision*, 2007a.

G.B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007b.

E. Hullermeier and J. Beringer. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 10(5):419–439, 2006.

R. Jin and Z. Ghahramani. Learning with multiple labels. In *Advances in Neural Information Processing Systems*, pages 897–904, 2002.

H. Kuck and N. de Freitas. Learning about individuals from group statistics. In *Uncertainty in Artificial Intelligence*, 2005.

I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

J. Luo and F. Orabona. Learning from candidate labeling sets. In *Advances in Neural Information Processing Systems*, 2010.

P. Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, pages 374–388, 1976.

P.J. Moreno, C. Joerg, J.M.V. Thong, and O. Glickman. A recursive algorithm for the forced alignment of very long audio segments. In *International Conference on Spoken Language Processing*, 1998.

J.G. Proakis and D.G. Manolakis. *Digital signal processing: principles, algorithms, and applications*. Prentice Hall, 1996.

N. Quadrianto, A.J. Smola, T.S. Caetano, and Q.V. Le. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10:2349–2374, 2009. ISSN 1532-4435.

D. Ramanan, S. Baker, and S. Kakade. Leveraging archival video for building face datasets. In *Proc. International Conference on Computer Vision*, 2007.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

S. Satoh, Y. Nakamura, and T. Kanade. Name-it: Naming and detecting faces in news videos. *IEEE MultiMedia*, 6(1):22–35, 1999.

K. Sjölander. An HMM-based system for automatic segmentation and alignment of speech. In *Fonetik*, pages 93–96, 2003.

D. Talkin. A robust algorithm for pitch tracking (RAPT). *Speech Coding and Synthesis*, pages 495–518, 1995.

A. Tewari and P. L. Bartlett. On the consistency of multiclass classification methods. In *International Conference on Learning Theory*, volume 3559, pages 143–157, 2005.

G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. *Data Mining and Knowledge Discovery Handbook*, pages 667–685, 2010.

P. Vannoorenberghe and P. Smets. Partially supervised learning by a credal EM approach. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 956–967, 2005.

P. Viola, J. Platt, and C. Zhang. Multiple instance boosting for object detection. *Advances in Neural Information Processing Systems*, 18:1417, 2006.

R. Yan., J. Zhang, J. Yang, and A.G. Hauptmann. A discriminative learning framework with pairwise constraints for video object classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):578–593, 2006.

T. Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004. ISSN 1533-7928.

Z.H. Zhou and M.L. Zhang. Multi-instance multi-label learning with application to scene classification. *Advances in Neural Information Processing Systems*, 19:1609, 2007.

X. Zhu and A.B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009.

Journal of Machine Learning Research 12 (2011) 1537-1586 Submitted 5/10; Revised 12/10; Published 5/11

Super-Linear Convergence of Dual Augmented Lagrangian Algorithm for Sparsity Regularized Estimation

Ryota Tomioka TOMIOKA@MIST.I.U-TOKYO.AC.JP
Taiji Suzuki S-TAIJI@STAT.T.U-TOKYO.AC.JP
Department of Mathematical Informatics
The University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan

Masashi Sugiyama SUGI@CS.TITECH.AC.JP
Department of Computer Science
Tokyo Institute of Technology
2-12-1-W8-74, O-okayama, Meguro-ku, Tokyo, 152-8552, Japan

Editor: Tong Zhang

Abstract

We analyze the convergence behaviour of a recently proposed algorithm for regularized estimation called Dual Augmented Lagrangian (DAL). Our analysis is based on a new interpretation of DAL as a proximal minimization algorithm. We theoretically show under some conditions that DAL converges super-linearly in a non-asymptotic and global sense. Due to a special modelling of sparse estimation problems in the context of machine learning, the assumptions we make are milder and more natural than those made in conventional analysis of augmented Lagrangian algorithms. In addition, the new interpretation enables us to generalize DAL to wide varieties of sparse estimation problems. We experimentally confirm our analysis in a large scale ℓ_1-regularized logistic regression problem and extensively compare the efficiency of DAL algorithm to previously proposed algorithms on both synthetic and benchmark data sets.

Keywords: dual augmented Lagrangian, proximal minimization, global convergence, sparse estimation, convex optimization

1. Introduction

Sparse estimation through convex regularization has become a common practice in many application areas including bioinformatics and natural language processing. However facing the rapid increase in the size of data-sets that we analyze everyday, clearly needed is the development of optimization algorithms that are tailored for machine learning applications.

Regularization-based sparse estimation methods estimate unknown variables through the minimization of a loss term (or a data-fit term) plus a regularization term. In this paper, we focus on convex methods; that is, both the loss term and the regularization term are convex functions of unknown variables. Regularizers may be nondifferentiable on some points; the nondifferentiability can promote various types of sparsity on the solution.

Although the problem is convex, there are three factors that challenge the straight-forward application of general tools for convex optimization (Boyd and Vandenberghe, 2004) in the context of machine learning.

The first factor is the diversity of loss functions. Arguably the squared loss is most commonly used in the field of signal/image reconstruction, in which many algorithms for sparse estimation have been developed (Figueiredo and Nowak, 2003; Daubechies et al., 2004; Cai et al., 2008). However the variety of loss functions is much wider in machine learning, to name a few, logistic loss and other log-linear loss functions. Note that these functions are not necessarily strongly convex like the squared loss. See Table 1 for a list of loss functions that we consider.

The second factor is the nature of the data matrix, which we call the design matrix in this paper. For a regression problem, the design matrix is defined by stacking input vectors along rows. If the input vectors are numerical (e.g., gene expression data), the design matrix is dense and has no structure. In addition, the characteristics of the matrix (e.g., the condition number) is unknown until the data is provided. Therefore, we would like to minimize assumptions about the design matrix, such as, sparse, structured, or well conditioned.

The third factor is the large number of unknown variables (or parameters) compared to observations. This is a situation regularized estimation methods are commonly applied. This factor may have been overlooked in the context of signal denoising, in which the number of observations and the number of parameters are equal.

Various methods have been proposed for efficient sparse estimation (see Figueiredo and Nowak, 2003; Daubechies et al., 2004; Combettes and Wajs, 2005; Andrew and Gao, 2007; Koh et al., 2007; Wright et al., 2009; Beck and Teboulle, 2009; Yu et al., 2010, and the references therein). Many previous studies focus on the *nondifferentiability* of the regularization term. In contrast, we focus on the *couplings* between variables (or non-separability) caused by the design matrix. In fact, if the optimization problem can be decomposed into smaller (e.g., containing a single variable) problems, optimization is easy. Recently Wright et al. (2009) showed that the so called iterative shrinkage/thresholding (IST) method (see Figueiredo and Nowak, 2003; Daubechies et al., 2004; Combettes and Wajs, 2005; Figueiredo et al., 2007a) can be seen as an iterative *separable approximation* process.

In this paper, we show that a recently proposed dual augmented Lagrangian (DAL) algorithm (Tomioka and Sugiyama, 2009) can be considered as an *exact* (up to finite tolerance) version of the iterative approximation process discussed in Wright et al. (2009). Our formulation is based on the connection between the proximal minimization (Rockafellar, 1976a) and the augmented Lagrangian (AL) algorithm (Hestenes, 1969; Powell, 1969; Rockafellar, 1976b; Bertsekas, 1982). The proximal minimization framework also allows us to rigorously study the convergence behaviour of DAL. We show that DAL converges super-linearly under some mild conditions, which means that the number of iterations that we need to obtain an ε-accurate solution grows no greater than logarithmically with $1/\varepsilon$. Due to the generality of the framework, our analysis applies to a wide variety of practically important regularizers. Our analysis improves the classical result on the convergence of augmented Lagrangian algorithms in Rockafellar (1976b) by taking special structures of sparse estimation into account. In addition, we make no asymptotic arguments as in Rockafellar (1976b) and Kort and Bertsekas (1976); instead our convergence analysis is build on top of the recent result in Beck and Teboulle (2009).

Augmented Lagrangian formulations have also been considered in Yin et al. (2008) and Goldstein and Osher (2009) for sparse signal reconstruction. What differentiates DAL approach of Tomioka and Sugiyama (2009) from those studied earlier is that the AL algorithm is applied to the dual problem (see Section 2.2), which results in an inner minimization problem that can be solved efficiently exploiting the sparsity of intermediate solutions (see Section 4.1). Applying AL

	primal loss $f_\ell(\mathbf{z})$	conjugate loss $f_\ell^*(-\alpha)$	gradient $-\nabla f_\ell^*(-\alpha)$	Hessian $\nabla^2 f_\ell^*(-\alpha)$	γ
Squared loss	$\frac{1}{2}\sum_{i=1}^n \frac{(y_i-z_i)^2}{\sigma_i^2}$	$\frac{1}{2}\sum_{i=1}^n \sigma_i^2(\alpha_i-y_i)^2$	$\mathrm{diag}\left(\sigma_1^2,\dots,\sigma_m^2\right)(\alpha-\mathbf{y})$	$\mathrm{diag}\left(\sigma_1^2,\dots,\sigma_m^2\right)$	$\min_i \sigma_i^2$
Logistic loss	$\sum_{i=1}^n \log(1+\exp(-y_iz_i))$	$\sum_{i=1}^n \big((\alpha_iy_i)\log(\alpha_iy_i)$ $+(1-\alpha_iy_i)\log(1-\alpha_iy_i)\big)$	$\left(y_i\log\frac{\alpha_iy_i}{1-\alpha_iy_i}\right)_{i=1}^m$	$\mathrm{diag}(\frac{1}{\alpha_iy_i(1-\alpha_iy_i)})_{i=1}^m$	4
Hyperbolic secant likelihood (Haufe et al., 2010)	$\sum_{i=1}^n \log\left(e^{y_i-z_i}+e^{-y_i+z_i}\right)$	$\frac{1}{2}\sum_{i=1}^n \big((1-\alpha_i)\log(1-\alpha_i)$ $+(1+\alpha_i)\log(1+\alpha_i)-2\alpha_iy_i\big)$	$(\frac{1}{2}\log\frac{1+\alpha_i}{1-\alpha_i}-y_i)_{i=1}^m$	$\mathrm{diag}(\frac{1}{2(1-\alpha_i)(1+\alpha_i)})_{i=1}^m$	2
Multi-class logit (Tomioka and Müller, 2010)	$\sum_{i=1}^m \big(-\sum_{k=1}^{c-1} z_{i(k)}y_{ik}$ $+\log\big(\sum_{k=1}^{c-1} e^{z_{i(k)}}+1\big)\big)$	$\sum_{i=1}^m\big(\sum_{k=1}^{c-1}(y_{ik}-\alpha_{i(k)})\log(y_{ik}-\alpha_{i(k)})$ $+(y_{ic}+\sum_{k=1}^{c-1}\alpha_{i(k)})\log(y_{ic}+\sum_{k=1}^{c-1}\alpha_{i(k)})\big)$ $(0\le y_{ik}-\alpha_{i(k)}\le 1$ $(k=1,\dots,c-1),$ $0\le y_{ic}+\sum_{k=1}^{c-1}\alpha_{i(k)}\le 1)$	$\left(-\log\frac{y_{ik}-\alpha_{i(k)}}{y_{ic}+\sum_{k=1}^{c-1}\alpha_{i(k)}}\right)_{i(k)=1}^{m(c-1)}$ $(i=1,\dots,m;$ $k=1,\dots,c-1)$	$\left(\frac{\delta_{i,j}\delta_{k,l}}{y_{ik}-\alpha_{i(k)}}+\frac{\delta_{i,j}}{y_{ic}+\sum_{k'=1}^{c-1}\alpha_{i(k')}}\right)_{i(k),j(l)=1}^{m(c-1)}$ $(i,j=1,\dots,m;$ $k,l=1,\dots,c-1)$	1

Table 1: List of loss functions and their convex conjugates. Constant terms are ignored. For the multi-class logit loss, $\mathbf{z},\alpha\in\mathbb{R}^{m(c-1)}$, where m is the number of samples and c is the number of classes; $y_{ik}=1$ if the ith sample belongs to the kth class, and zero otherwise; $i(k):=(i-1)c+k$ denotes the linear index corresponding to the kth output for the ith sample; $\delta_{i,k}$ denotes the Kronecker delta function.

formulation to the dual problem also plays an important role in the convergence analysis because some loss functions (e.g., logistic loss) are not strongly convex in the primal; see Section 5. Recently Yang and Zhang (2009) compared primal and dual augmented Lagrangian algorithms for ℓ_1-problems and reported that the dual formulation was more efficient. See also Tomioka et al. (2011b) for related discussions.

This paper is organized as follows. In Section 2, we mathematically formulate the sparse estimation problem and we review DAL algorithm. We derive DAL algorithm from the proximal minimization framework in Section 3; special instances of DAL algorithm are discussed in Section 4. In Section 5, we theoretically analyze the convergence behaviour of DAL algorithm. We discuss previously proposed algorithms in Section 6 and contrast them with DAL. In Section 7 we confirm our analysis in a simulated ℓ_1-regularized logistic regression problem. Moreover, we extensively compare recently proposed algorithms for ℓ_1-regularized logistic regression including DAL in synthetic and benchmark data sets under a variety of conditions. Finally we summarize our contribution in Section 8. Most of the proofs are given in the appendix.

2. Sparse Estimation Problem and DAL Algorithm

In this section, we first formulate the sparse estimation problem as a convex optimization problem, and state our assumptions. Next we derive DAL algorithm for ℓ_1-problem as an augmented Lagrangian method in the dual.

2.1 Objective

We consider the problem of estimating an n dimensional parameter vector from m training examples as described in the following optimization problem:

$$\underset{\mathbf{w}\in\mathbb{R}^n}{\text{minimize}} \quad \underbrace{f_\ell(\mathbf{A}\mathbf{w}) + \phi_\lambda(\mathbf{w})}_{=:f(\mathbf{w})}, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^n$ is the parameter vector to be estimated, $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a design matrix, and $f_\ell(\cdot)$ is a loss function. We call the first term in the minimand the loss term and the second term the regularization term, or the regularizer.

We assume that the loss function $f_\ell : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is a closed proper strictly convex function.[1] See Table 1 for examples of loss functions. We assume that f_ℓ has Lipschitz continuous gradient with modulus $1/\gamma$ (see Assumption **(A2)** in Section 5.2). If f_ℓ is twice differentiable, this condition is equivalent to saying that the maximum eigenvalue of the Hessian of f_ℓ is uniformly bounded by $1/\gamma$. Such γ exists for example for quadratic loss, logistic loss, and other log-linear losses. However, non-smooth loss functions (e.g., the hinge loss and the absolute loss) are excluded. Note that since we separate the data matrix $\mathbf{A}$ from the loss function, we can quantify the above constant γ without examining the data. Moreover, we assume that the convex conjugate[2] f_ℓ^* is (essentially) twice differentiable. Note that the first order differentiability of the convex conjugate f_ℓ^* is implied by the strict convexity of the loss function f_ℓ (Rockafellar, 1970, Theorem 26.3).

1. "Closed" means that the epigraph $\{(\mathbf{z}, y) \in \mathbb{R}^{m+1} : y \geq f_\ell(\mathbf{z})\}$ is a closed set, and "proper" means that the function is not everywhere $+\infty$; see, for example, Rockafellar (1970). In the sequel, we use the word "convex function" in the meaning of "closed proper convex function".
2. The convex conjugate of a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a function f^* over $\mathbb{R}^n$ that takes values in $\mathbb{R}^n \cup \{+\infty\}$ and is defined as $f^*(\mathbf{y}) = \sup_{\mathbf{x}\in\mathbb{R}^n}(\mathbf{y}^\top\mathbf{x} - f(\mathbf{x}))$.

The regularization term $\phi_\lambda(\mathbf{w})$ is a convex possibly nondifferentiable function. In addition, we assume that for all $\eta > 0$, $\eta\phi_\lambda(\mathbf{w}) = \phi_{\eta\lambda}(\mathbf{w})$.

An important special case, which has been studied by many authors (Tibshirani, 1996; Efron et al., 2004; Andrew and Gao, 2007; Koh et al., 2007) is the ℓ_1-regularization:

$$\underset{\mathbf{w}\in\mathbb{R}^n}{\text{minimize}} \quad f_\ell(\mathbf{A}\mathbf{w}) + \lambda\|\mathbf{w}\|_1, \tag{2}$$

where $\|\mathbf{w}\|_1 = \sum_{j=1}^n |w_j|$ is the ℓ_1-norm of $\mathbf{w}$.

2.2 Dual Augmented Lagrangian (DAL) Algorithm

In this subsection, we review DAL algorithm following the line of Tomioka and Sugiyama (2009). Although, the squared loss function and the ℓ_1-regularizer were considered in the original paper, we deal with a slightly more general setting in Equation (2) for notational convenience; that is, we consider general closed convex loss functions instead of the squared loss. For general information on augmented Lagrangian algorithms (Powell, 1969; Hestenes, 1969; Rockafellar, 1976b), see Bertsekas (1982) and Nocedal and Wright (1999).

Let $\phi_\lambda(\mathbf{w})$ be the ℓ_1-regularizer, that is, $\phi_\lambda(\mathbf{w}) = \lambda\|\mathbf{w}\|_1 = \lambda\sum_{j=1}^n |w_j|$. Using the Fenchel duality theorem (Rockafellar, 1970), the dual of the problem (2) can be written as follows:

$$\underset{\alpha\in\mathbb{R}^m,\mathbf{v}\in\mathbb{R}^n}{\text{maximize}} \quad -f_\ell^*(-\alpha) - \delta_\lambda^\infty(\mathbf{v}), \tag{3}$$

$$\text{subject to} \quad \mathbf{v} = \mathbf{A}^\top\alpha, \tag{4}$$

where δ_λ^∞ is the indicator function (Rockafellar, 1970, p28) of the ℓ_∞-ball of radius λ, namely

$$\delta_\lambda^\infty(\mathbf{v}) = \sum_{j=1}^n \delta_\lambda^\infty(v_j), \tag{5}$$

where $\delta_\lambda^\infty(v_j) = 0$, if $|v_j| \le \lambda$, and $+\infty$ otherwise.

Let us consider the augmented Lagrangian (AL) function L_η with respect to the above dual problem (3)

$$L_\eta(\alpha, \mathbf{v}; \mathbf{w}) = -f_\ell^*(-\alpha) - \delta_\lambda^\infty(\mathbf{v}) + \mathbf{w}^\top(\mathbf{v} - \mathbf{A}^\top\alpha) - \frac{\eta}{2}\|\mathbf{v} - \mathbf{A}^\top\alpha\|^2, \tag{6}$$

where the primal variable $\mathbf{w} \in \mathbb{R}^n$ is interpreted as a Lagrangian multiplier vector in the AL framework. Note that the AL function is the ordinary Lagrangian if $\eta = 0$.

Let $\eta_0, \eta_1, \ldots$ be a non-decreasing sequence of positive numbers. At every time step t, given the current primal solution $\mathbf{w}^t$, we maximize the AL function $L_{\eta_t}(\alpha, \mathbf{v}; \mathbf{w}^t)$ with respect to α and $\mathbf{v}$. The maximizer $(\alpha^t, \mathbf{v}^t)$ is used to update the primal solution (Lagrangian multiplier) $\mathbf{w}^t$ as follows:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \eta_t(\mathbf{A}^\top\alpha^t - \mathbf{v}^t). \tag{7}$$

Note that the maximization of the AL function (6) with respect to $\mathbf{v}$ can be carried out in a closed form, because the terms involved in the maximization can be separated into n terms, each containing single v_j, as follows:

$$L_{\eta_t}(\alpha, \mathbf{v}) = -f_\ell^*(-\alpha) - \sum_{j=1}^n \left(\frac{\eta_t}{2}(v_j - (\mathbf{w}^t/\eta_t + \mathbf{A}^\top\alpha)_j)^2 + \delta_\lambda^\infty(v_j)\right),$$

where $(\cdot)_j$ denotes the jth element of a vector. Since $\delta_\lambda^\infty(v_j)$ is infinity outside the domain $-\lambda \leq v_j \leq \lambda$, the maximizer $\mathbf{v}^t(\alpha)$ is obtained as a projection onto the ℓ_∞ ball of radius λ as follows (see also Figure 9):

$$\mathbf{v}^t(\alpha) = \mathrm{proj}_{[-\lambda,\lambda]}\left(\mathbf{w}^t/\eta_t + \mathbf{A}^\top\alpha\right) := \left(\min\left(|y_j|,\lambda\right)\frac{y_j}{|y_j|}\right)_{j=1}^n, \tag{8}$$

where $(y_j)_{j=1}^n$ denotes an n-dimensional vector whose jth element is given by y_j. Note that the ratio $y_j/|y_j|$ is defined to be zero[3] if $y_j = 0$. Substituting the above $\mathbf{v}^t$ back into Equation (7), we obtain the following update equation:

$$\mathbf{w}^{t+1} = \mathrm{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t\mathbf{A}^\top\alpha^t),$$

where $\mathrm{prox}_{\lambda\eta_t}^{\ell_1}$ is called the soft-threshold operation[4] and is defined as follows:

$$\mathrm{prox}_\lambda^{\ell_1}(\mathbf{y}) := \left(\max(|y_j| - \lambda, 0)\frac{y_j}{|y_j|}\right)_{j=1}^n. \tag{9}$$

The soft-threshold operation is well known in signal processing community and has been studied extensively (Donoho, 1995; Figueiredo and Nowak, 2003; Daubechies et al., 2004; Combettes and Wajs, 2005).

Furthermore, substituting the above $\mathbf{v}^t(\alpha)$ into Equation (6), we can express α^t as the minimizer of the function

$$\varphi_t(\alpha) := -L_{\eta_t}(\alpha, \mathbf{v}^t(\alpha); \mathbf{w}^t) = f_\ell^*(-\alpha) + \frac{1}{2\eta_t}\|\mathrm{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t\mathbf{A}^\top\alpha)\|^2, \tag{10}$$

which we also call an AL function with a slight abuse of terminology. Note that the maximization in Equation (6) is turned into a minimization of the above function by negating the AL function.

3. Proximal Minimization View

The first contribution of this paper is to derive DAL algorithm we reviewed in Section 2.2 from the proximal minimization framework (Rockafellar, 1976a), which allows for a new interpretation of the algorithm (see Section 3.3) and rigorous analysis of its convergence (see Section 5).

3.1 Proximal Minimization Algorithm

Let us consider the following iterative algorithm called the proximal minimization algorithm (Rockafellar, 1976a) for the minimization of the objective (1).

1. Choose some initial solution $\mathbf{w}^0$ and a sequence of non-decreasing positive numbers $\eta_0 \leq \eta_1 \leq \cdots$.

3. This is equivalent to defining $y_j/|y_j| = \mathrm{sign}(y_j)$. We use $y_j/|y_j|$ instead of $\mathrm{sign}(y_j)$ to define the soft-threshold operations corresponding to ℓ_1 and the group-lasso regularizations (see Section 4.2) in a similar way.
4. This notation is a simplified version of the general notation we introduce later in Equation (15).

2. Repeat until some criterion (e.g., duality gap Wright et al., 2009; Tomioka and Sugiyama, 2009) is satisfied:

$$\mathbf{w}^{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \left(f(\mathbf{w}) + \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}^t\|^2 \right), \tag{11}$$

   where $f(\mathbf{w})$ is the objective function in Equation (1) and η_t controls the influence of the additional *proximity term*.

The proximity term tries to keep the next solution $\mathbf{w}^{t+1}$ close to the current solution $\mathbf{w}^t$. Importantly, the objective (11) is strongly convex even if the original objective (1) is not; see Rockafellar (1976b). Although at this point it is not clear how we are going to carry out the above minimization, by definition we have $f(\mathbf{w}^{t+1}) + \frac{1}{2\eta_t}\|\mathbf{w}^{t+1}-\mathbf{w}^t\|^2 \leq f(\mathbf{w}^t)$; that is, provided that the step-size is positive, the function value decreases monotonically at every iteration.

3.2 Iterative Shrinkage/Thresholding Algorithm from the Proximal Minimization Framework

The function to be minimized in Equation (11) is strongly convex. However, there seems to be no obvious way to minimize Equation (11), because it is still (possibly) nondifferentiable and cannot be decomposed into smaller problems because the elements of $\mathbf{w}$ are coupled.

One way to make the proximal minimization algorithm practical is to linearly approximate (see Wright et al., 2009) the loss term at the current point $\mathbf{w}^t$ as

$$f_\ell(\mathbf{A}\mathbf{w}) \simeq f_\ell(\mathbf{A}\mathbf{w}^t) + \left(\nabla f_\ell^t\right)^\top \mathbf{A}\left(\mathbf{w}-\mathbf{w}^t\right), \tag{12}$$

where ∇f_ℓ^t is a short hand for $\nabla f_\ell(\mathbf{A}\mathbf{w}^t)$. Substituting the above approximation (12) into the iteration (11), we obtain

$$\mathbf{w}^{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \left(\left(\nabla f_\ell^t\right)^\top \mathbf{A}\mathbf{w} + \phi_\lambda(\mathbf{w}) + \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}^t\|^2 \right), \tag{13}$$

where constant terms are omitted from the right-hand side. Note that because of the linear approximation, there is no coupling between the elements of $\mathbf{w}$. For example, if $\phi_\lambda(\mathbf{w}) = \lambda\|\mathbf{w}\|$, the minimand in the right-hand side of the above equation can be separated into n terms each containing single w_j, which can be separately minimized.

Rewriting the above update equation, we obtain the well-known iterative shrinkage/ thresholding (IST) method[5] (Figueiredo and Nowak, 2003; Daubechies et al., 2004; Combettes and Wajs, 2005; Figueiredo et al., 2007a). The IST iteration can be written as follows:

$$\mathbf{w}^{t+1} := \operatorname{prox}_{\phi_{\lambda\eta_t}}\left(\mathbf{w}^t - \eta_t\mathbf{A}^\top\nabla f_\ell^t\right), \tag{14}$$

where the proximity operator $\operatorname{prox}_{\phi_{\lambda\eta_t}}$ is defined as follows:

$$\operatorname{prox}_{\phi_\lambda}(\mathbf{y}) = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} \left(\frac{1}{2}\|\mathbf{y}-\mathbf{x}\|^2 + \phi_\lambda(\mathbf{x}) \right). \tag{15}$$

Note that the soft-threshold operation $\operatorname{prox}_\lambda^{\ell_1}$ (9) is the proximity operator corresponding to the ℓ_1-regularizer $\phi_\lambda^{\ell_1}(\mathbf{w}) = \lambda\|\mathbf{w}\|_1$.

5. It is also known as the forward-backward splitting method (Lions and Mercier, 1979; Combettes and Wajs, 2005; Duchi and Singer, 2009); see Section 6.

3.3 DAL Algorithm from the Proximal Minimization Framework

The above IST approach can be considered to be constructing a linear lower bound of the loss term in Equation (11) at the *current point* $\mathbf{w}^t$. In this subsection we show that we can precisely (to finite precision) minimize Equation (11) using a *parametrized* linear lower bound that can be adjusted to be the tightest at the *next point* $\mathbf{w}^{t+1}$. Our approach is based on the convexity of the loss function f_ℓ. First note that we can rewrite the loss function f_ℓ as a point-wise maximum as follows:

$$f_\ell(\mathbf{A}\mathbf{w}) = \max_{\alpha\in\mathbb{R}^m}\left((-\alpha)^\top \mathbf{A}\mathbf{w} - f_\ell^*(-\alpha)\right), \tag{16}$$

where f_ℓ^* is the convex conjugate functions of f_ℓ. Now we substitute this expression into the iteration (11) as follows:

$$\mathbf{w}^{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n}\max_{\alpha\in\mathbb{R}^m}\left\{-\alpha^\top\mathbf{A}\mathbf{w} - f_\ell^*(-\alpha) + \phi_\lambda(\mathbf{w}) + \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}^t\|^2\right\}. \tag{17}$$

Note that now the loss term is expressed as a *linear* function as in the IST approach; see Equation (13). Now we exchange the order of minimization and maximization because the function to be minimaxed in Equation (17) is a saddle function (i.e., convex with respect to $\mathbf{w}$ and concave with respect to α Rockafellar, 1970), as follows:

$$\begin{aligned}&\min_{\mathbf{w}\in\mathbb{R}^n}\max_{\alpha\in\mathbb{R}^m}\left\{-\alpha^\top\mathbf{A}\mathbf{w} - f_\ell^*(-\alpha) + \phi_\lambda(\mathbf{w}) + \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}^t\|^2\right\}\\ &\quad= \max_{\alpha\in\mathbb{R}^m}\left\{-f_\ell^*(-\alpha) + \min_{\mathbf{w}\in\mathbb{R}^n}\left(-\alpha^\top\mathbf{A}\mathbf{w} + \phi_\lambda(\mathbf{w}) + \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}^t\|^2\right)\right\}.\end{aligned} \tag{18}$$

Notice the similarity between the two minimizations (13) and (18) (with fixed α).

The minimization with respect to $\mathbf{w}$ in Equation (18) gives the following update equation

$$\mathbf{w}^{t+1} = \operatorname{prox}_{\phi_{\lambda\eta_t}}\left(\mathbf{w}^t + \eta_t\mathbf{A}^\top\alpha^t\right), \tag{19}$$

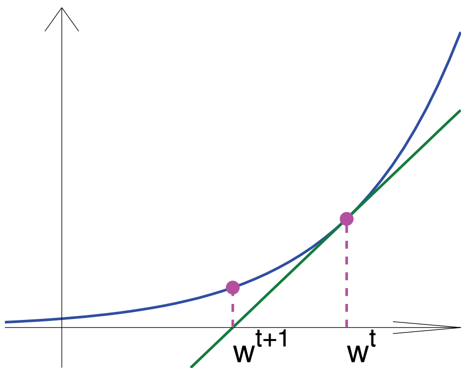
where α^t denotes the maximizer with respect to α in Equation (18). Note that α^t is in general different from $-\nabla f_\ell^t$ used in the IST approach (14). Actually, we show below that $\alpha^t = -\nabla f_\ell^{t+1}$ if the max-min problem (18) is solved exactly. Therefore taking $\alpha^t = -\nabla f_\ell^t$ can be considered as a naive approximation to this.

The final step to derive DAL algorithm is to compute the maximizer α^t in Equation (18). This step is slightly involved and the derivation is presented in Appendix B. The result of the derivation can be written as follows (notice that the maximization in Equation (18) is turned into a minimization by reversing the sign):

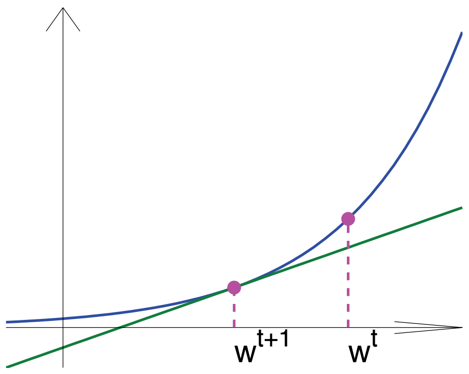
$$\alpha^t = \operatorname*{argmin}_{\alpha\in\mathbb{R}^m}\underbrace{\left(f_\ell^*(-\alpha) + \frac{1}{\eta_t}\Phi^*_{\lambda\eta_t}(\mathbf{w}^t + \eta_t\mathbf{A}^\top\alpha)\right)}_{=:\varphi_t(\alpha)}, \tag{20}$$

where the function $\Phi^*_{\lambda\eta_t}$ is called the Moreau envelope of ϕ^*_λ (see Moreau, 1965; Rockafellar, 1970) and is defined as follows:

$$\Phi^*_\lambda(\mathbf{w}) = \min_{\mathbf{x}\in\mathbb{R}^n}\left(\phi^*_\lambda(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{w}\|^2\right). \tag{21}$$



(a) Gradient-based lower-bound used in IST



(b) Variational lower-bound used in DAL

Figure 1: Comparison of the lower bounds used in IST and DAL.

See Appendix A for more details. Since the function $\varphi_t(\alpha)$ in Equation (20) generalizes the AL-function (10), we call it an augmented Lagrangian (AL) function.

What we need to do at every iteration is to minimize the AL function $\varphi_t(\alpha)$ and update the Lagrangian multiplier $\mathbf{w}^t$ as in Equation (19) using the minimizer α^t in Equation (20). Of course in practice we would like to stop the inner minimization at a finite tolerance. We discuss the stopping condition in Section 5.

The algorithm we derived above is indeed a generalization of DAL algorithm we reviewed in Section 2.2. This can be shown by computing the proximity operator (19) and the Moreau envelope (21) for the specific case of ℓ_1-regularization; see Section 4.1 and also Table 2.

The AL function $\varphi_t(\alpha)$ is continuously differentiable, because the AL function is a sum of f_ℓ^* (differentiable by assumption) and an envelope function (differentiable; see Appendix A). In fact, using Lemma 10 in Appendix A, the derivative of the AL function can be evaluated as follows:

$$\nabla\varphi_t(\alpha) = -\nabla f_\ell^*(-\alpha) + \mathbf{A}\mathbf{w}^{t+1}(\alpha), \tag{22}$$

where $\mathbf{w}^{t+1}(\alpha) := \mathrm{prox}_{\phi_{\lambda\eta_t}}\left(\mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha\right)$. The expression for the second derivative depends on the particular regularizer chosen.

Notice again that the above update Equation (19) is very similar to the one in the IST approach Equation (14). However, $-\alpha$, which is the slope of the lower-bound (16) is optimized in the inner minimization (20) so that the lower-bound is the tightest at the *next point* $\mathbf{w}^{t+1}$. In fact, if $\nabla\varphi_t(\alpha) = 0$ then $\nabla f_\ell(\mathbf{A}\mathbf{w}^{t+1}) = -\alpha^t$ because of Equation (22) and $\nabla f_\ell(\nabla f_\ell^*(-\alpha^t)) = -\alpha^t$. The difference between the strategies used in IST and DAL to construct a lower-bound is highlighted in Figure 1. IST uses a fixed gradient-based lower-bound which is tightest at the current solution $\mathbf{w}^t$, whereas DAL uses a variational lower-bound, which can be adjusted to become tightest at the next solution $\mathbf{w}^{t+1}$.

The general connection between the augmented Lagrangian algorithm and the proximal minimization algorithm, and (asymptotic) convergence results can be found in Rockafellar (1976b) and Bertsekas (1982). The derivation we show above is a special case when the objective function $f(\mathbf{w})$ can be split into a part that is easy to handle (regularization term $\phi_\lambda(\mathbf{w})$) and the rest (loss term $f_\ell(\mathbf{A}\mathbf{w})$).

Description	Regularizer	Proximity operator prox_λ	Envelope function Φ^*_λ
ℓ_1-regularizer (Tibshirani, 1996)	$\phi^{\ell_1}_\lambda(\mathbf{w}) = \lambda\sum_{j=1}^n \lvert w_j\rvert$	$\mathrm{prox}^{\ell_1}_\lambda(\mathbf{w}) = \left((\lvert w_j\rvert - \lambda)_+ \frac{w_j}{\lvert w_j\rvert}\right)_{j=1}^n$	$\Phi^*_\lambda(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^n (\lvert w_j\rvert - \lambda)_+^2$
Group lasso (Yuan and Lin, 2006)	$\phi^{\mathfrak{G}}_\lambda(\mathbf{w}) = \lambda\sum_{\mathfrak{g}\in\mathfrak{G}} \lVert\mathbf{w}_\mathfrak{g}\rVert$	$\mathrm{prox}^{\mathfrak{G}}_\lambda(\mathbf{w}) = \left((\lVert\mathbf{w}_\mathfrak{g}\rVert - \lambda)_+ \frac{\mathbf{w}_\mathfrak{g}}{\lVert\mathbf{w}_\mathfrak{g}\rVert}\right)_{\mathfrak{g}\in\mathfrak{G}}$	$\Phi^*_\lambda(\mathbf{w}) = \frac{1}{2}\sum_{\mathfrak{g}\in\mathfrak{G}} (\lVert\mathbf{w}_\mathfrak{g}\rVert - \lambda)_+^2$
Trace norm (Fazel et al., 2001; Srebro et al., 2005; Tomioka et al., 2010)	$\phi^{\mathrm{tr}}_\lambda(\mathbf{w}) = \lambda\sum_{j=1}^n \sigma_j(\mathbf{w})$	$\mathrm{prox}^{\mathrm{tr}}_\lambda(\mathbf{w}) = \mathrm{vec}\left(\mathbf{U}(\mathbf{S}-\lambda)_+\mathbf{V}^\top\right)$	$\Phi^*_\lambda(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^r (\sigma_j(\mathbf{w}) - \lambda)_+^2$
Elastic-net (Zou and Hastie, 2005; Tomioka and Suzuki, 2010)	$\phi^{\mathrm{en}}_\lambda(\mathbf{w}) = \lambda\sum_{j=1}^n ((1-\theta)\lvert w_j\rvert + \frac{\theta}{2}w_j^2)$	$\mathrm{prox}^{\mathrm{en}}_\lambda(\mathbf{w}) = \left(\frac{(\lvert w_j\rvert - \lambda(1-\theta))_+}{1+\lambda\theta}\frac{w_j}{\lvert w_j\rvert}\right)_{j=1}^n$	$\Phi^*_\lambda(\mathbf{w}) = \frac{1+\lambda\theta}{2}\sum_{j=1}^n \left(\frac{\lvert w_j\rvert - \lambda(1-\theta)}{1+\lambda\theta}\right)_+^2$

Table 2: List of regularizers and their corresponding proximity operators (15) and the Envelope function (21). The operation $(\cdot)_+$ is defined as $(x)_+ := \max(0,x)$ and applies element-wise to a matrix.

Algorithm 1 DAL algorithm for ℓ_1-regularization

```
1: Input: design matrix $\mathbf{A}$, loss function $f_\ell$, regularization constant $\lambda$, sequence of proximity parameters $\eta_t$ $(t = 0, 1, 2, \ldots)$, initial solution $\mathbf{w}^0$, tolerance $\varepsilon$.
2: Set $t = 0$.
3: repeat
4:     Minimize the augmented Lagrangian function $\varphi_t(\alpha)$ (see Equation 10) with the gradient and Hessian given in Equations (24) and (25), respectively, using Newton's method. Let $\alpha^t$ be the approximate minimizer
           $\alpha^t \simeq \operatorname*{argmin}_{\alpha\in\mathbb{R}^m}\left(f_\ell^*(-\alpha) + \frac{1}{2\eta_t}\left\|\mathrm{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t\mathbf{A}^\top\alpha)\right\|^2\right),$
       with the stopping criterion (see Section 5.2)
           $\|\nabla\varphi_t(\alpha^t)\| \leq \sqrt{\frac{\gamma}{\eta_t}}\left\|\mathrm{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t\mathbf{A}^\top\alpha^t) - \mathbf{w}^t\right\|,$
       where $\varphi_t(\alpha)$ is the derivative of the inner objective (24) and $1/\gamma$ is the Lipschitz constant of $\nabla f_\ell$.
5:     Update $\mathbf{w}^{t+1} := \mathrm{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t\mathbf{A}^\top\alpha^t)$, $t \leftarrow t + 1$.
6: until relative duality gap (see Section 7.1.2) is less than the tolerance $\varepsilon$.
7: Output: the final solution $\mathbf{w}^t$.
```

4. Exemplary Instances

In this section, we discuss special instances of DAL framework presented in Section 3 and qualitatively discuss the efficiency of minimizing the inner objective. We first discuss the simple case of ℓ_1-regularization (Section 4.1), and then group-lasso (Section 4.2) and other more general regularization using the so-called support functions (Section 4.3). In addition, the case of component-wise regularization is discussed in Section 4.4. See also Table 2 for a list of regularizers.

4.1 Dual Augmented Lagrangian Algorithm for ℓ_1-Regularization

For the ℓ_1-regularization, $\phi_\lambda^{\ell_1}(\mathbf{w}) = \lambda\|\mathbf{w}\|_1$, the update Equation (19) can be rewritten as follows:

$$\mathbf{w}^{t+1} = \mathrm{prox}_{\lambda\eta_t}^{\ell_1}\left(\mathbf{w}^t + \eta_t\mathbf{A}^\top\alpha^t\right), \tag{23}$$

where $\mathrm{prox}_\lambda^{\ell_1}$ is the proximity operator corresponding to the ℓ_1-regularizer defined in Equation (9). Moreover, noticing that the convex conjugate of the ℓ_1-regularizer is the indicator function δ_λ^∞ in Equation (5), we can derive the envelope function Φ_λ^* in Equation (21) as follows (see also Figure 9):

$$\Phi_\lambda^*(\mathbf{w}) = \frac{1}{2}\left\|\mathrm{prox}_\lambda^{\ell_1}(\mathbf{w})\right\|^2.$$

Therefore, the AL function (10) in Tomioka and Sugiyama (2009) is derived from the proximal minimization framework (see Equation 20) in Section 3.

We use Newton's method for the minimization of the inner objective $\varphi_t(\alpha)$. The overall algorithm is shown in Algorithm 1. The gradient and Hessian of the AL function (10) can be evaluated as follows (Tomioka and Sugiyama, 2009):

$$\nabla\varphi_t(\alpha) = -\nabla f_\ell^*(-\alpha) + \mathbf{A}\mathbf{w}^{t+1}(\alpha), \tag{24}$$

$$\nabla^2\varphi_t(\alpha) = \nabla^2 f_\ell^*(-\alpha) + \eta_t \mathbf{A}_+ {\mathbf{A}_+}^\top, \tag{25}$$

where $\mathbf{w}^{t+1}(\alpha) := \mathrm{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha)$, and $\mathbf{A}_+$ is the matrix that consists of columns of $\mathbf{A}$ that corresponds to "active" variables (i.e., the non-zero elements of $\mathbf{w}^{t+1}(\alpha)$). Note that Equation (24) equals the general expression (22) from the proximal minimization framework.

It is worth noting that in both the computation of matrix-vector product in Equation (24) and the computation of matrix-matrix product in Equation (25), the cost is only proportional to the number of non-zero elements of $\mathbf{w}^{t+1}(\alpha)$. Thus when we are aiming for a sparse solution, the minimization of the AL function (10) can be performed efficiently.

4.2 Group Lasso

Let ϕ_λ be the group-lasso penalty (Yuan and Lin, 2006), that is,

$$\phi_\lambda^{\mathfrak{G}}(\mathbf{w}) = \lambda \sum_{\mathfrak{g}\in\mathfrak{G}} \|\mathbf{w}_\mathfrak{g}\|, \tag{26}$$

where $\mathfrak{G}$ is a disjoint partition of the index set $\{1,\ldots,n\}$, and $\mathbf{w}_\mathfrak{g} \in \mathbb{R}^{|\mathfrak{g}|}$ is a sub-vector of $\mathbf{w}$ that consists of rows of $\mathbf{w}$ indicated by $\mathfrak{g} \subseteq \{1,\ldots,n\}$. The proximity operator corresponding to the group-lasso regularizer $\phi_\lambda^{\mathfrak{G}}$ is obtained as follows:

$$\mathrm{prox}_\lambda^{\mathfrak{G}}(\mathbf{y}) := \mathrm{prox}_{\phi_\lambda^{\mathfrak{G}}}(\mathbf{y}) = \left(\max(\|\mathbf{y}_\mathfrak{g}\| - \lambda, 0)\frac{\mathbf{y}_\mathfrak{g}}{\|\mathbf{y}_\mathfrak{g}\|}\right)_{\mathfrak{g}\in\mathfrak{G}}, \tag{27}$$

where similarly to Equation (9), $(\mathbf{y}_\mathfrak{g})_{\mathfrak{g}\in\mathfrak{G}}$ denotes an n-dimensional vector whose $\mathfrak{g}$ component is given by $\mathbf{y}_\mathfrak{g}$. Moreover, analogous to update Equation (23) (see also Equation 10) in the ℓ_1-case, the update equations can be written as follows:

$$\mathbf{w}^{t+1} = \mathrm{prox}_{\lambda\eta_t}^{\mathfrak{G}}\left(\mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha^t\right),$$

where α^t is the minimizer of the AL function

$$\varphi_t(\alpha) = f_\ell^*(-\alpha) + \frac{1}{2\eta_t}\|\mathrm{prox}_{\lambda\eta_t}^{\mathfrak{G}}(\mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha)\|^2. \tag{28}$$

The overall algorithm is obtained by replacing the soft-thresholding operations in Algorithm 1 by the one defined above (27). In addition, the gradient and Hessian of the AL function $\varphi_t(\alpha)$ can be written as follows:

$$\nabla\varphi_t(\alpha) = -\nabla f_\ell^*(-\alpha) + \mathbf{A}\mathbf{w}^{t+1}(\alpha), \tag{29}$$

$$\nabla^2\varphi_t(\alpha) = \nabla^2 f_\ell^*(-\alpha) + \eta_t \sum_{\mathfrak{g}\in\mathfrak{G}^+} \mathbf{A}_\mathfrak{g}\left(\left(1 - \frac{\lambda\eta_t}{\|\mathbf{q}_\mathfrak{g}\|}\right)\mathbf{I}_{|\mathfrak{g}|} + \frac{\lambda\eta_t}{\|\mathbf{q}_\mathfrak{g}\|}\tilde{\mathbf{q}}_\mathfrak{g}{\tilde{\mathbf{q}}_\mathfrak{g}}^\top\right){\mathbf{A}_\mathfrak{g}}^\top, \tag{30}$$

where $\mathbf{w}^{t+1}(\alpha) = \mathrm{prox}^{\mathfrak{G}}_{\lambda\eta_t}(\mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha)$, and $\mathfrak{G}^+$ is a subset of $\mathfrak{G}$ that consists of active groups, namely $\mathfrak{G}^+ := \{\mathfrak{g} \in \mathfrak{G} : \|\mathbf{w}^{t+1}_{\mathfrak{g}}(\alpha)\| > 0\}$; $\mathbf{A}_{\mathfrak{g}}$ is a sub-matrix of $\mathbf{A}$ that consists of columns corresponding to the index-set $\mathfrak{g}$; $\mathbf{I}_{|\mathfrak{g}|}$ is the $|\mathfrak{g}| \times |\mathfrak{g}|$ identity matrix; the vector $\mathbf{q} \in \mathbb{R}^n$ is defined as $\mathbf{q} := \mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha$ and $\tilde{\mathbf{q}}_{\mathfrak{g}} := \mathbf{q}_{\mathfrak{g}} / \|\mathbf{q}_{\mathfrak{g}}\|$, where $\mathbf{q}_{\mathfrak{g}}$ is defined analogously to $\mathbf{w}_{\mathfrak{g}}$. Note that in the above expression, $\lambda\eta_t / \|\mathbf{q}_{\mathfrak{g}}\| \leq 1$ for $\mathfrak{g} \in \mathfrak{G}^+$ by the soft-threshold operation (27).

Similarly to the ℓ_1-case in the last subsection, the sparsity of $\mathbf{w}^{t+1}(\alpha)$ (i.e., $|\mathfrak{G}^+| \ll |\mathfrak{G}|$) can be exploited to efficiently compute the gradient (29) and the Hessian (30).

4.3 Support Functions

The ℓ_1-norm regularization and the group lasso regularization in Equation (26) can be generalized to the class of support functions. The support function of a convex set C_λ is defined as follows:

$$\phi_\lambda(\mathbf{x}) = \sup_{\mathbf{y} \in C_\lambda} \mathbf{x}^\top \mathbf{y}. \tag{31}$$

For example, the ℓ_1-norm is the support function of the ℓ_∞ unit ball (see Rockafellar, 1970) and the group lasso regularizer (26) is the support function of the group-generalized ℓ_∞-ball defined as $\{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y}_{\mathfrak{g}}\| \leq \lambda, \forall \mathfrak{g} \in \mathfrak{G}\}$. It is well known that the convex conjugate of the support function (31) is the indicator function of C (see Rockafellar, 1970), namely,

$$\phi^*_\lambda(\mathbf{y}) = \begin{cases} 0 & (\text{if } \mathbf{y} \in C_\lambda), \\ +\infty & (\text{otherwise}). \end{cases} \tag{32}$$

The proximity operator corresponding to the support function (31) can be written as follows:

$$\mathrm{prox}^{\sup}_{C_\lambda}(\mathbf{y}) := \mathbf{y} - \mathrm{proj}_{C_\lambda}(\mathbf{y}),$$

where $\mathrm{proj}_{C_\lambda}$ is the projection onto C_λ; see Lemma 8 in Appendix A. Finally, by computing the Moreau envelope (21) corresponding to the above ϕ^*_λ, we have

$$\varphi_t(\alpha) = f^*_\ell(-\alpha) + \frac{1}{2\eta_t} \|\mathrm{prox}^{\sup}_{C_{\lambda\eta_t}}(\mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha)\|^2, \tag{33}$$

where we used the fact that for the indicator function in Equation (32), $\phi^*_\lambda(\mathrm{proj}_{C_\lambda}(\mathbf{z})) = 0$ $(\forall \mathbf{z})$ and Lemma 8. Note that $C_\lambda = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y}\|_\infty \leq \lambda\}$ gives $\mathrm{prox}^{\sup}_{C_\lambda} = \mathrm{prox}^{\ell_1}_\lambda$ (see Equation 10), and $C_\lambda = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y}_{\mathfrak{g}}\| \leq \lambda, \forall \mathfrak{g} \in \mathfrak{G}\}$ gives $\mathrm{prox}^{\sup}_{C_\lambda} = \mathrm{prox}^{\mathfrak{G}}_\lambda$ (see Equation (28).

4.4 Handling Different Regularization Constant for Each Component

The ℓ_1-regularizer in Section 4.1 and the group lasso regularizer in Section 4.2 assume that all the components (variables or groups) are regularized by the same constant λ. However the general formulation in Section 3.3 allows using different regularization constant for each component.

For example, let us consider the following regularizer:

$$\phi_\lambda(\mathbf{w}) = \sum_{j=1}^{n} \lambda_j |w_j|, \tag{34}$$

where $\lambda_j \geq 0$ $(j = 1, \dots, n)$. Note that we can also include unregularized terms (e.g., a bias term) by setting the corresponding regularization constant $\lambda_j = 0$. The soft-thresholding operation corresponding to the regularizer (34) is written as follows:

$$\mathrm{prox}_{\lambda}^{\ell_1}(\mathbf{y}) = \left(\max(|y_j| - \lambda_j, 0) \frac{y_j}{|y_j|} \right)_{j=1}^{n},$$

where again the ratio $y_j/|y_j|$ is defined to be zero if $y_j = 0$. Note that if $\lambda_j = 0$, the soft-thresholding operation is an identity mapping for that component. Moreover, by noticing that the regularizer (34) is a support function (see Section 4.3), the envelope function Φ_{λ}^{*} in Equation (21) is written as follows:

$$\Phi_{\lambda}^{*}(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^{n} \max^2(|w_j| - \lambda_j, 0),$$

which can also be derived by noticing that $\Phi_{\lambda}^{*}(0) = 0$ and $\nabla \Phi_{\lambda}^{*}(\mathbf{y}) = \mathrm{prox}_{\lambda}^{\ell_1}(\mathbf{y})$ (Lemma 10 in Appendix A).

As a concrete example, let b be an unregularized bias term and let us assume that all the components of $\mathbf{w} \in \mathbb{R}^n$ are regularized by the same regularization constant λ. In other words, we aim to solve the following optimization problem:

$$\underset{\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}}{\text{minimize}} \quad f_{\ell}(\mathbf{A}\mathbf{w} + \mathbf{1}_m b) + \lambda \|\mathbf{w}\|_1,$$

where $\|\mathbf{w}\|_1$ is the ℓ_1-norm of $\mathbf{w}$, and $\mathbf{1}_m$ is an m-dimensional all one vector. The update Equations (19) and (20) can be written as follows:

$$\mathbf{w}^{t+1} = \mathrm{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t \mathbf{A}^{\top} \alpha^t), \tag{35}$$

$$b^{t+1} = b^t + \eta_t \mathbf{1}_m{}^{\top} \alpha^t, \tag{36}$$

where α^t is the minimizer of the AL function as follows:

$$\alpha^t = \underset{\alpha \in \mathbb{R}^m}{\mathrm{argmin}} \left(f_{\ell}^{*}(-\alpha) + \frac{1}{2\eta_t} \left(\|\mathrm{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t \mathbf{A}^{\top} \alpha)\|^2 + (b^t + \eta_t \mathbf{1}_m{}^{\top} \alpha)^2 \right) \right). \tag{37}$$

5. Analysis

In this section, we first show the convergence of DAL algorithm assuming that the inner minimization problem (20) is solved exactly (Section 5.1), which is equivalent to the proximal minimization algorithm (11). The convergence is presented both in terms of the function value and the norm of the residual. Next, since it is impractical to perform the inner minimization to high precision, the finite tolerance version of the two theorems are presented in Section 5.2. The convergence rate obtained in Section 5.2 is slightly worse than the exact case. In Section 5.3, we show that the convergence rate can be improved by performing the inner minimization more precisely. Most of the proofs are given in Appendix C for the sake of readability.

Our result is inspired partly by Beck and Teboulle (2009) and is similar to the one given in Rockafellar (1976a) and Kort and Bertsekas (1976). However, our analysis does not require asymptotic arguments as in Rockafellar (1976a) or rely on the strong convexity of the objective as in Kort and Bertsekas (1976). Importantly the stopping criterion we discuss in Section 5.2 can be checked in practice. Key to our analysis is the Lipschitz continuity of the gradient of the loss function ∇f_ℓ and the assumption that the proximation with respect to ϕ_λ (see Equation 15) can be computed exactly. Connections between our assumption and the ones made in earlier studies are discussed in Section 5.4.

5.1 Exact Inner Minimization

Lemma 1 (Beck and Teboulle, 2009) *Let $\mathbf{w}^1, \mathbf{w}^2, \ldots$ be the sequence generated by the proximal minimization algorithm (Equation 11). For arbitrary $\mathbf{w} \in \mathbb{R}^n$ we have*

$$\eta_t(f(\mathbf{w}^{t+1}) - f(\mathbf{w})) \leq \frac{1}{2}\|\mathbf{w}^t - \mathbf{w}\|^2 - \frac{1}{2}\|\mathbf{w}^{t+1} - \mathbf{w}\|^2. \tag{38}$$

Proof First notice that $(\mathbf{w}^t - \mathbf{w}^{t+1})/\eta_t \in \partial f(\mathbf{w}^{t+1})$ because $\mathbf{w}^{t+1}$ minimizes Equation (11). Therefore using the convexity of f, we have[6]

$$\begin{aligned}\eta_t(f(\mathbf{w}) - f(\mathbf{w}^{t+1})) &\geq \langle \mathbf{w} - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^{t+1} \rangle \\ &= \langle \mathbf{w} - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w} + \mathbf{w} - \mathbf{w}^{t+1} \rangle \\ &\geq \|\mathbf{w} - \mathbf{w}^{t+1}\|^2 - \|\mathbf{w} - \mathbf{w}^{t+1}\|\|\mathbf{w}^t - \mathbf{w}\| \\ &\geq \frac{1}{2}\|\mathbf{w} - \mathbf{w}^{t+1}\|^2 - \frac{1}{2}\|\mathbf{w}^t - \mathbf{w}\|^2,\end{aligned} \tag{39}$$

where the third line follows from Cauchy-Schwartz inequality and the last line follows from the inequality of arithmetic and geometric means. ■

Note that DAL algorithm (Equations 19 and 20) with exact inner minimization generates a sequence from the proximal minimization algorithm (Equation 11). Therefore we have the following theorem.

Theorem 2 *Let $\mathbf{w}^1, \mathbf{w}^2, \ldots$ be the sequence generated by DAL algorithm (Equations 19 and 20); let W^* be the set of minimizers of the objective* (1) *and let $f(W^*)$ denote the minimum objective value. If the inner minimization (Equation 20) is solved exactly and the proximity parameter η_t is increased exponentially, then the residual function value obtained by the DAL algorithm converges exponentially fast to zero as follows:*

$$f(\mathbf{w}^{k+1}) - f(W^*) \leq \frac{\|\mathbf{w}^0 - W^*\|^2}{2C_k}, \tag{40}$$

where $\|\mathbf{w}^0 - W^*\|$ denotes the minimum distance between the initial solution $\mathbf{w}^0$ and W^*, namely, $\|\mathbf{w}^0 - W^*\| = \min_{\mathbf{w}^* \in W^*} \|\mathbf{w}^0 - \mathbf{w}^*\|$. Note that $C_k = \sum_{t=0}^{k} \eta_t$ also grows exponentially.

6. We use the notation $\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{j=1}^{n} x_j y_j$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Proof Let $\mathbf{w}^*$ be any point in W^*. Substituting $\mathbf{w} = \mathbf{w}^*$ in Equation (38) and summing both sides from $t = 1$ to $t = k$, we have

$$\left(\textstyle\sum_{t=0}^{k}\eta_t\right)\left(\sum_{t=0}^{k}\frac{\eta_t f(\mathbf{w}^{t+1})}{\sum_{t=0}^{k}\eta_t} - f(\mathbf{w}^*)\right) \leq \frac{1}{2}\|\mathbf{w}^0 - \mathbf{w}^*\|^2 - \frac{1}{2}\|\mathbf{w}^{k+1} - \mathbf{w}^*\|^2$$
$$\leq \frac{1}{2}\|\mathbf{w}^0 - \mathbf{w}^*\|^2.$$

In addition, since $f(\mathbf{w}^{t+1}) \leq f(\mathbf{w}^t)$ $(t = 0, 1, 2, \ldots)$ from Equation (11), we have

$$\left(\textstyle\sum_{t=0}^{k}\eta_t\right)\left(f(\mathbf{w}^{k+1}) - f(\mathbf{w}^*)\right) \leq \frac{1}{2}\|\mathbf{w}^0 - \mathbf{w}^*\|^2.$$

Finally, taking the minimum of the right-hand side with respect to $\mathbf{w}^* \in W^*$ and using the equivalence of proximal minimization (11) and DAL algorithm (19)-(20) (see Section 3.3), we complete the proof. ■

The above theorem claims the convergence of the residual function values $f(\mathbf{w}^t) - f(\mathbf{w}^*)$ obtained along the sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots$. We can convert the above result into convergence in terms of the residual norm $\|\mathbf{w}^t - \mathbf{w}^*\|$ by introducing an assumption that connects the residual function value to the residual norm. In addition, we slightly generalize Lemma 1 to improve the convergence rate. Consequently, we obtain the following theorem.

Theorem 3 *Let $\mathbf{w}^1, \mathbf{w}^2, \ldots$ be the sequence generated by DAL algorithm (Equations 19 and 20) and let W^* be the set of minimizers of the objective* (1). *Let us assume that there are a positive constant σ and a scalar α ($1 \leq \alpha \leq 2$) such that*

(A1) $$f(\mathbf{w}^{t+1}) - f(W^*) \geq \sigma\|\mathbf{w}^{t+1} - W^*\|^{\alpha} \qquad (t = 0, 1, 2, \ldots), \tag{41}$$

where $f(W^*)$ denotes the minimum objective value, and $\|\mathbf{w} - W^*\|$ denotes the minimum distance between $\mathbf{w} \in \mathbb{R}^n$ and the set of minimizers W^* as $\|\mathbf{w} - W^*\| := \min_{\mathbf{w}^* \in W^*}\|\mathbf{w} - \mathbf{w}^*\|$.

If the inner minimization is solved exactly, we have the following inequality:

$$\|\mathbf{w}^{t+1} - W^*\| + \sigma\eta_t\|\mathbf{w}^{t+1} - W^*\|^{\alpha-1} \leq \|\mathbf{w}^t - W^*\|.$$

Moreover, this implies that

$$\|\mathbf{w}^{t+1} - W^*\|^{\frac{1+(\alpha-1)\sigma\eta_t}{1+\sigma\eta_t}} \leq \frac{1}{1+\sigma\eta_t}\|\mathbf{w}^t - W^*\|. \tag{42}$$

That is, $\mathbf{w}^t$ converges to W^* super-linearly *if $\alpha < 2$ or $\alpha = 2$ and η_t is increasing, in a* global and non-asymptotic *sense.*

Proof See Appendix C.1. ■

Note that the above super-linear convergence holds without the assumption in Theorem 2 that η_t is increased exponentially.

5.2 Approximate Inner Minimization

First we present a finite tolerance version of Lemma 1.

Lemma 4 *Let $\mathbf{w}^1, \mathbf{w}^2, \dots$ be the sequence generated by DAL algorithm (Equations 19 and 20). Let us assume the following conditions.*

(A2) *The loss function f_ℓ has a Lipschitz continuous gradient with modulus $1/\gamma$, that is,*

$$\left\|\nabla f_\ell(\mathbf{z}) - \nabla f_\ell(\mathbf{z}')\right\| \leq \frac{1}{\gamma}\|\mathbf{z}-\mathbf{z}'\| \qquad (\forall \mathbf{z}, \mathbf{z}' \in \mathbb{R}^m), \tag{43}$$

(A3) *The proximation with respect to ϕ_λ (see Equation 15) can be computed exactly.*

(A4) *The inner minimization (Equation 20) is solved to the following tolerance:*

$$\|\nabla \varphi_t(\alpha^t)\| \leq \sqrt{\frac{\gamma}{\eta_t}}\|\mathbf{w}^{t+1}-\mathbf{w}^t\|, \tag{44}$$

where γ is the constant in Equation (43).

Under assumptions **(A2)**–**(A4)***, for arbitrary $\mathbf{w} \in \mathbb{R}^n$ we have*

$$\eta_t(f(\mathbf{w}^{t+1}) - f(\mathbf{w})) \leq \frac{1}{2}\|\mathbf{w}^t - \mathbf{w}\|^2 - \frac{1}{2}\|\mathbf{w}^{t+1}-\mathbf{w}\|^2. \tag{45}$$

Proof See Appendix C.2. ■

Note that Lemma 4 states that even with the weaker stopping criterion **(A4)**, we can obtain inequality (45) as in Lemma 1.

The assumptions we make here are rather weak. In Assumption **(A2)**, the loss function f_ℓ does not include the design matrix $\mathbf{A}$ (see Table 1). Therefore, it is easy to compute the constant γ. Accordingly, the stopping criterion **(A4)** can be checked without assuming anything about the data.

Furthermore, summing both sides of inequality (45) and assuming that η_t is increased exponentially, we obtain Theorem 2 also under the approximate minimization **(A4)**.

Finally, an analogue of Theorem 3, which does not assume the exponential increase in η_t, is obtained as follows.

Theorem 5 *Let $\mathbf{w}^1, \mathbf{w}^2, \dots$ be the sequence generated by DAL algorithm and let W^* be the set of minimizers of the objective* (1)*. Under assumption* **(A1)** *in Theorem 3 and* **(A2)**-**(A4)** *in Lemma 4, we have*

$$\|\mathbf{w}^{t+1} - W^*\|^2 + 2\sigma\eta_t\|\mathbf{w}^{t+1}-W^*\|^\alpha \leq \|\mathbf{w}^t - W^*\|^2,$$

where $\|\mathbf{w}^t - W^*\|$ is the minimum distance between $\mathbf{w}^t$ and W^* as in Theorem 3. Moreover, this implies that

$$\|\mathbf{w}^{t+1} - W^*\|^{\frac{1+\alpha\sigma\eta_t}{1+2\sigma\eta_t}} \leq \frac{1}{\sqrt{1+2\sigma\eta_t}}\|\mathbf{w}^t - W^*\|. \tag{46}$$

That is, $\mathbf{w}^t$ converges to W^* super-linearly if $\alpha < 2$ or $\alpha = 2$ and η_t is increasing.

Proof Let $\mathbf{w}^t$ be the closest point in W^* from $\mathbf{w}^t$, that is, $\mathbf{w}^t := \mathrm{argmin}_{\mathbf{w}^* \in W^*} \|\mathbf{w}^t - \mathbf{w}^*\|$. Using Lemma 4 with $\mathbf{w} = \mathbf{w}^t$ and Assumption $(\mathbf{A1})$, we have the first part of the theorem as follows:

$$\begin{aligned}\|\mathbf{w}^t - W^*\|^2 = \|\mathbf{w}^t - \mathbf{w}^t\|^2 &\geq \|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 + 2\sigma\eta_t \|\mathbf{w}^{t+1} - W^*\|^\alpha \\ &\geq \|\mathbf{w}^{t+1} - W^*\|^2 + 2\sigma\eta_t \|\mathbf{w}^{t+1} - W^*\|^\alpha,\end{aligned}$$

where we used the minimality of $\|\mathbf{w}^{t+1} - \mathbf{w}^{t+1}\|$ in the second line. The last part of the theorem (46) can be obtained in a similar manner as that of Theorem 3 using Young's inequality (see Appendix C.1). ■

5.3 A Faster Rate

The factor $1/\sqrt{1+2\sigma\eta_t}$ obtained under the approximate minimization **(A4)** (see inequality (46) in Theorem 5) is larger than that obtained under the exact inner minimization (see inequality (42) in Theorem 3); that is, the statement in Theorem 5 is weaker than that in Theorem 3.

Here we show that a better rate can also be obtained for approximate minimization if we perform the inner minimization to $O(\|\mathbf{w}^{t+1} - \mathbf{w}^t\|/\eta_t)$ instead of $O(\|\mathbf{w}^{t+1} - \mathbf{w}^t\|/\sqrt{\eta_t})$ in Assumption **(A4)**.

Theorem 6 *Let* $\mathbf{w}^1, \mathbf{w}^2, \ldots$ *be the sequence generated by DAL algorithm and let* W^* *be the set of minimizers of the objective* (1). *Under assumption* **(A1)** *in Theorem 3 with* $\alpha = 2$, *and assumptions* **(A2)** *and* **(A3)** *in Lemma 4, for any* $\varepsilon < 1$ *such that* $\delta := (1-\varepsilon)/(\sigma\eta_t) \leq 3/4$, *if we solve the inner minimization to the following precision*

(A4′) $$\|\nabla\varphi_t(\alpha^t)\| \leq \frac{\sqrt{\gamma(1-\varepsilon)/\sigma}}{\eta_t}\|\mathbf{w}^{t+1} - \mathbf{w}^t\|,$$

then we have

$$\|\mathbf{w}^{t+1} - W^*\| \leq \frac{1}{1+\varepsilon\sigma\eta_t}\|\mathbf{w}^t - W^*\|.$$

Proof See Appendix C.3 ■

Note that the assumption $\delta < 3/4$ is rather weak, because if the factor δ is greater than one, the stopping criterion $(\mathbf{A4'})$ would be weaker than the earlier criterion $(\mathbf{A4})$. In order to be on the safe side, we can choose $\varepsilon = \max(\varepsilon_0, 1 - 3\sigma\eta_t/4)$ (assuming that we know the constant σ) and the above statement holds with $\varepsilon = \varepsilon_0$. Unfortunately, in exchange for obtaining a faster rate, the stopping criterion **(A4′)** now depends not only on γ, which can be computed, but also on σ, which is hard to know in practice. Therefore stopping condition **(A4′)** is not practical.

5.4 Validity of Assumption (A1)

In this subsection, we discuss the validity of assumption **(A1)** and its relation to the assumptions used in Rockafellar (1976a) and Kort and Bertsekas (1976). Roughly speaking, our assumption **(A1)** is milder than the one used in Rockafellar (1976a) and stronger than the one used in Kort and Bertsekas (1976).

First of all, assumption **(A1)** is unnecessary for convergence in terms of function value (Theorem 2 and its approximate version implied by Lemma 4). Exponential increase of the proximity parameter η_t may sound restrictive, but this is the setting we typically use in experiments (see Section 7.1). Assumption **(A1)** is only necessary in Theorem 3 and Theorem 5 to translate the residual function value $f(\mathbf{w}^{t+1}) - f(W^*)$ into the residual distance $\|\mathbf{w}^{t+1} - W^*\|$.

We can roughly think of Assumption **(A1)** as a *local strong convexity* assumption. Here we say a function f is *locally strongly convex around* the set of minimizers W^* if for all positive C, all the points $\mathbf{w}$ within distance C from the set W^*, the objective function f is bounded below by a quadratic function, that is,

$$f(\mathbf{w}) - f(W^*) \geq \sigma\|\mathbf{w} - W^*\|^2 \quad (\forall \mathbf{w} : \|\mathbf{w} - W^*\| \leq C), \tag{47}$$

where the positive constant σ may depend on C. If the set of minimizers W^* is bounded, all the level sets of f are bounded (see Rockafellar, 1970, Theorem 8.7). Therefore, if we make sure that the function value $f(\mathbf{w}^t)$ does not increase during the minimization, we can assume that all points generated by DAL algorithm are contained in some neighborhood around W^* that contains the level set defined by the initial function value $\{\mathbf{w} \in \mathbb{R}^n : f(\mathbf{w}) \leq f(\mathbf{w}^0)\}$; that is, the local strong convexity of f guarantees Assumption **(A1)** with $\alpha = 2$.

Note that Kort and Bertsekas (1976, p278) used a slightly weaker assumption than the local strong convexity (47); they assumed that there *exists* a positive constant $C' > 0$ such that the local strong convexity (47) is true for all $\mathbf{w}$ in the neighborhood $\|\mathbf{w} - W^*\| \leq C'$ for some $\sigma > 0$.

The local strong convexity (47) or Assumption **(A1)** fails when the objective function behaves like a constant function around the set of minimizers W^*. In this case, DAL converges rapidly in terms of function value due to Theorem 2; however it does not necessarily converge in terms of the distance $\|\mathbf{w}^t - W^*\|$.

Note that the objective function f is the sum of the loss term and the regularization term. Even if the minimum eigenvalue of the Hessian of the loss term is very close to zero, we can hope that the regularization term holds the function up from the minimum objective value $f(W^*)$. For example, when the loss term is *zero* and we only have the ℓ_1-regularization term $\phi_\lambda^{\ell_1}(\mathbf{w})$. The objective $f(\mathbf{w}) = \lambda \sum_{j=1}^n |w_j|$ can be lower-bounded as

$$f(\mathbf{w}) \geq \frac{\lambda}{C}\|\mathbf{w}\|^2 \quad (\forall \mathbf{w} : \|\mathbf{w}\| \leq C),$$

where the minimizer $\mathbf{w}^*$ is $\mathbf{w}^* = \mathbf{0}$. Note that the ℓ_1-regularizer is not (globally) strongly convex. The same observation holds also for other regularizers we discussed in Section 4.

In the context of asymptotic analysis of AL algorithm, Rockafellar (1976b) assumed that there exists $\tau > 0$, such that in the ball $\|\beta\| \leq \tau$ in $\mathbb{R}^n$, the gradient of the convex conjugate f^* of the objective function f is Lipschitz continuous with constant L, that is,

$$\|\nabla f^*(\beta) - \nabla f^*(0)\| \leq L\|\beta\|.$$

Note that because $\partial f(\nabla f^*(0)) \ni 0$ (Rockafellar, 1970, Corollary 23.5.1), $\nabla f^*(0)$ is the optimal solution $\mathbf{w}^*$ of Equation (1), and it is unique by the continuity assumed above.

Our assumption **(A1)** can be justified from Rockafellar's assumption as follows.

Theorem 7 *Rockafellar's assumption implies that the objective f is locally strongly convex with $C = c\tau L$ and $\sigma = \min(1, (2c-1)/c^2)/(2L)$ for any positive constant c (τ and L are constants from Rockafellar's assumption).*

Proof The proof is a *local* version of the proof of Theorem X.4.2.2 in Hiriart-Urruty and Lemaréchal (1993) (Lipschitz continuity of ∇f^* implies strong convexity of f). See Appendix C.4. ■

Note that as the constant c that bounds the distance to the set of minimizers W^* increases, the constant σ becomes smaller and the convergence guarantee in Theorem 3 and 5 become weaker (but still valid).

Nevertheless Assumption **(A1)** we use in Theorem 3 and 5 are weaker than the local strong convexity (47), because we need Assumption **(A1)** to hold only on the points generated by DAL algorithm. For example, if we only consider a finite number of steps, such a constant σ always exists.

Both assumptions in Rockafellar (1976b) and Kort and Bertsekas (1976) are made for asymptotic analysis. In fact, they require that as the optimization proceeds, the solution becomes closer to the optimum $\mathbf{w}^*$ in the sense of the distance $\|\mathbf{w}^t - W^*\|$ in Kort and Bertsekas (1976) and $\|\beta\|$ in Rockafellar (1976b). However in both cases, it is hard to predict how many iterations it takes for the solution to be sufficiently close to the optimum so that the super-linear convergence happens.

Our analysis is complementary to the above classical results. We have shown that super-linear convergence happens *non-asymptotically* under Assumption **(A1)**, which is trivial for a finite number of steps. Assumption **(A1)** can also be guaranteed for infinite steps using the local strong convexity around W^* (47).

6. Previous Studies

In this section, we discuss earlier studies in two categories. The first category comprises methods that try to overcome the difficulty posed by the nondifferentiability of the regularization term $\phi_\lambda(\mathbf{w})$. The second category, which includes DAL algorithm in this paper, consists of methods that try to overcome the difficulty posed by the coupling (or non-separability) introduced by the design matrix $\mathbf{A}$. The advantages and disadvantages of all the methods are summarized in Table 3.

6.1 Constrained Optimization, Upper-Bound Minimization, and Subgradient Methods

Many authors have focused on the *nondifferentiability* of the regularization term in order to efficiently minimize Equation (1). This view has lead to three types of approaches, namely, (i) constrained optimization, (ii) upper-bound minimization, and (iii) subgradient methods.

In the constrained optimization approach, auxiliary variables are introduced to rewrite the non-differentiable regularization term as a linear function of conically-constrained auxiliary variables. For example, the ℓ_1-norm of a vector $\mathbf{w}$ can be rewritten as:

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{n} \min_{w_j^{(+)}, w_j^{(-)} \geq 0} \left(w_j^{(+)} + w_j^{(-)} \right) \quad \text{s.t.} \quad w_j = w_j^{(+)} - w_j^{(-)},$$

where $w_j^{(+)}$ and $w_j^{(-)}$ $(j = 1, \dots, n)$ are auxiliary variables and they are constrained in the positive-orthant cone. Two major challenges of the auxiliary-variable formulation are the increased size of the problem and the complexity of solving a constrained optimization problem.

The projected gradient (PG) method (see Bertsekas, 1999) iteratively computes a gradient step and projects it back to the constraint-set. The PG method in Figueiredo et al. (2007b) converges

R-linearly,[7] if the loss function is quadratic. However, PG methods can be extremely slow when the design matrix $\mathbf{A}$ is poorly conditioned. To overcome the scaling problem, the L-BFGS-B algorithm (Byrd et al., 1995) can be applied for the simple positive orthant constraint that arises from the ℓ_1 minimization. However, this approach does not easily extend to more general regularizers, such as group lasso and trace norm regularization.

The interior-point (IP) method (see Boyd and Vandenberghe, 2004) is another algorithm that is often used for constrained minimization; see Koh et al. 2007; Kim et al. 2007 for the application of IP methods to sparse estimation problems. Basically an IP method generates a sequence that approximately follows the so called central path, which parametrically connects the analytic center of the constraint-set and the optimal solution. Although IP methods can tolerate poorly conditioned design matrices well, it is challenging to scale them up to very large dense problems. The convergence of the IP method in Koh et al. (2007) is empirically found to be linear.

The second approach (upper-bound minimization) constructs a differentiable upper-bound of the nondifferentiable regularization term. For example, the ℓ_1-norm of a vector $\mathbf{w}$ can be rewritten as follows:

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{n} \min_{\alpha_j \geq 0} \left(\frac{w_j^2}{2\alpha_j} + \frac{\alpha_j}{2} \right). \tag{48}$$

In fact, the right-hand side is an upper bound of the left-hand side for arbitrary non-negative α_j due to the inequality of arithmetic and geometric means, and the equality is obtained by setting $\alpha_j = |w_j|$. The advantage of the above parametric-upper-bound formulation is that for a fixed set of α_j, the problem (2) becomes a (weighted) quadratically regularized minimization problem, for which various efficient algorithms already exist. The iteratively reweighted shrinkage (IRS) method (Gorodnitsky and Rao, 1997; Bioucas-Dias, 2006; Figueiredo et al., 2007a) alternately solves the quadratically regularized minimization problem and tightens (re-weights) the upper-bound in Equation (48). A more general technique was studied in parallel by the name of variational EM (Jaakkola, 1997; Girolami, 2001; Palmer et al., 2006), which generalizes the above upper-bound using Fenchel's inequality (Rockafellar, 1970). A similar approach that is based on Jensen's inequality (Rockafellar, 1970) has been studied in the context of multiple-kernel learning (Micchelli and Pontil, 2005; Rakotomamonjy et al., 2008) and in the context of multi-task learning (Argyriou et al., 2007, 2008). The challenge in the IRS framework is the *singularity* (Figueiredo et al., 2007a) around the coordinate axis. For example, in the ℓ_1-problem in Equation (2), any zero component $w_j = 0$ in the initial vector $\mathbf{w}$ will remain zero after any number of iterations. Moreover, it is possible to create a situation that the convergence becomes arbitrarily slow for finite $|w_j|$ because the convergence in the ℓ_1 case is only linear (Gorodnitsky and Rao, 1997).

The third approach (subgradient methods) directly handles the nondifferentiability through subgradients; see, for example, Bertsekas (1999).

A (stochastic) subgradient method typically converges as $O(1/\sqrt{k})$ for non-smooth problems in general and as $O(1/k)$ if the objective is strongly convex; see Shalev-Shwartz et al. (2007); Lan (2010). However, since the method is based on gradients, it can easily fail when the problem is poorly conditioned (see, e.g., Yu et al., 2010, Section 2.2). Therefore, one of the challenges in subgradient-based approaches is to take the second-order curvature information into account. This

7. A sequence ξ^t converges to ξ R-linearly (R is for "root") if the residual $|\xi^t - \xi|$ is bounded by a sequence ε^t that linearly converges to zero (Nocedal and Wright, 1999).

	Constrained Optimization		Upper-bound Minimization	Subgradient Method	Iterative Proximation	
	PG	IP	IRS	OWLQN	AG	DAL
Poorly conditioned **A**	–	✓	✓	✓	–	✓
No singularity	✓	✓	–	✓	✓	✓
Extensibility	✓	✓	✓	–	✓	✓
Exploits sparsity of **w**	✓	–	–	✓	✓	✓
Efficient when	–	–	–	$m \gg n$	$m \gg n$	$m \ll n$
Convergence	$(O(e^{-k}))$	$(O(e^{-k}))$	$O(e^{-k})$	?	$O(1/k^2)$	$o(e^{-k})$

Table 3: Comparison of the algorithms to solve Equation (1). In the columns, six methods, namely, projected gradient (PG), interior point (IP), iterative reweighted shrinkage (IRS), orthant-wise limited-memory quasi Newton (OWLQN), accelerated gradient (AG), and dual augmented Lagrangian (DAL), are categorized into four groups discussed in the text. The first row: "Poorly conditioned **A**" means that a method can tolerate poorly conditioned design matrices well. The second row: "No singularity" means that a method does not suffer from singularity in the parametrization (see main text). The third row: "Extensibility" means that a method can be easily extended beyond ℓ_1-regularization. The forth row: "Exploits sparsity of **w**" means that a method can exploit the sparsity in the intermediate solution. The fifth row: "Efficient when" indicates the situations each algorithm runs efficiently, namely, more samples than unknowns ($m \gg n$), more unknowns than samples ($m \ll n$), or does not matter (–). The last row shows the rate of convergence known from literature. The super-linear convergence of DAL is established in this paper.

is especially important to tackle large-scale problems with a possibly poorly conditioned design matrix. Orthant-wise limited memory quasi Newton (OWLQN, Andrew and Gao, 2007) and sub-LBFGS (Yu et al., 2010) combine subgradients with the well known L-BFGS quasi Newton method (Nocedal and Wright, 1999). Although being very efficient for ℓ_1-regularization and piecewise linear loss functions, these methods depend on the efficiency of oracles that compute a descent direction and a step-size; therefore, it is challenging to extend these methods to combinations of general loss functions and general nondifferentiable regularizers. In addition, the convergence rates of the OWLQN and subLBFGS methods are not known.

6.2 Iterative Proximation

Yet another approach is to deal with the nondifferentiable regularization through the proximity operation. In fact, the proximity operator (15) is easy to compute for many practically relevant separable regularizers.

The remaining issue, therefore, is the coupling between variables introduced by the design matrix **A**. We have shown in Sections 3.2 and 3.3 that IST and DAL can be considered as two different strategies to remove this coupling.

Recently many studies have focused on methods that iteratively compute the proximal operation (15) (Figueiredo and Nowak, 2003; Daubechies et al., 2004; Combettes and Wajs, 2005; Nesterov, 2007; Beck and Teboulle, 2009; Cai et al., 2008), which can be described in an abstract manner as follows:

$$\mathbf{w}^{t+1} = \mathrm{prox}_{\phi_{\lambda_t}}\left(\mathbf{y}^t\right), \tag{49}$$

where $\mathrm{prox}_{\phi_\lambda}$ is the proximal operator defined in Equation (15). The above mentioned studies can be differentiated by the different $\mathbf{y}^t$ and λ_t that they use.

For example, the IST approach (also known as the *forward-backward splitting* Lions and Mercier, 1979; Combettes and Wajs, 2005; Duchi and Singer, 2009) can be described as follows:

$$\mathbf{y}^t := \mathbf{w}^t - \eta_t \mathbf{A}^\top \nabla f_\ell(\mathbf{A}\mathbf{w}^t),$$
$$\lambda_t := \lambda \eta_t.$$

What we need to do at every iteration is only to compute the gradient at the current point, take a gradient step, and then perform the proximal operation (Equation 49). Note that η_t can be considered as a step-size.

The IST method can be considered as a generalization of the projected gradient method. Since the proximal gradient step (13) reduces to an ordinary gradient step when $\phi_\lambda = 0$, the basic idea behind IST is to keep the non-smooth term ϕ_λ as a part of the proximity step (see Lan, 2010). Consequently, the convergence behaviour of IST is the same as that of (projected) gradient descent on the differentiable loss term. Note that Duchi and Singer (2009) analyze the case where the loss term is also nondifferentiable in both batch and online learning settings. Langford et al. (2009) also analyze the online setting with a more general threshold operation.

IST approach maintains sparsity of $\mathbf{w}^t$ throughout the optimization, which results in significant reduction of computational cost; this is an advantage of iterative proximation methods compared to interior-point methods (e.g., Koh et al., 2007), because the solution produced by interior-point methods becomes sparse only in an asymptotic sense; see Boyd and Vandenberghe (2004).

The downside of the IST approach is the difficulty to choose the step-size parameter η_t; this issue is especially problematic when the design matrix $\mathbf{A}$ is poorly conditioned. In addition, the best known convergence rate of a naive IST approach is $O(1/k)$ (Beck and Teboulle, 2009), which means that the number of iterations k that we need to obtain a solution $\mathbf{w}^k$ such that $f(\mathbf{w}^k) - f(\mathbf{w}^*) \leq \varepsilon$ grows linearly with $1/\varepsilon$, where $f(\mathbf{w}^*)$ is the minimal value of Equation (1).

SpaRSA (Wright et al., 2009) uses approximate second order curvature information for the selection of the step-size parameter η_t. TwIST (Bioucas-Dias and Figueiredo, 2007) is a "two-step" approach that tries to alleviate the poor efficiency of IST when the design matrix is poorly conditioned. However the convergence rates of SpaRSA and TwIST are unknown.

Accelerating strategies that use different choices of $\mathbf{y}^t$ have been proposed in Nesterov (2007) and Beck and Teboulle (2009) (denoted AG in Tab. 3), which have $O(1/k^2)$ guarantee with almost the same computational cost per iteration; see also Lan (2010).

DAL can be considered as a new member of the family of iterative proximation algorithms. We have qualitatively shown in Section 3.3 that DAL constructs a better lower bound of the loss term than IST. Moreover, we have rigorously studied the convergence rate of DAL and have shown that it converges super-linearly. Of course the fast convergence of DAL comes with the increased cost per iteration. Nevertheless, as we have qualitatively discussed in Section 4, this increase is mild, because the sparsity of intermediate solutions can be effectively exploited in the inner minimization. We empirically compare DAL with other methods in Section 7.

There is of course an issue on how much one should precisely optimize when the training error (plus the regularization term) is a crude approximation of the generalization error (Shalev-Shwartz and Srebro, 2008). However the reason we use sparse regularization is exactly that we are not only interested in the predictive power. We argue that when we are using sparse methods to gain insights into some problem, it is important that we are sure that we are doing what we write in our paper (e.g., "solve an ℓ_1-regularized minimization problem"), and someone else can reliably recover the same sparsity pattern using any optimization approach that employs some objective stopping criterion such as the duality gap. Of course the stability of the optimal solution itself must be analyzed (see Bickel et al., 2009; Zhao and Yu, 2006; Meinshausen and Bühlmann, 2006) and the trade-off between accuracy and sparsity should be discussed. However, this is beyond the scope of this paper.

7. Empirical Results

In this section, we confirm the super-linear convergence of DAL algorithm and compare it with other algorithms on ℓ_1-regularized logistic regression problems. The algorithms that we compare are FISTA (Beck and Teboulle, 2009), OWLQN (Andrew and Gao, 2007), SpaRSA (Wright et al., 2009), IRS (Figueiredo et al., 2007a), and L1_LOGREG (Koh et al., 2007). Note that IST is not included because SpaRSA and FISTA are shown to clearly outperform the naive IST approach. We describe the logistic regression problem and the implementation of all of the methods in Section 7.1. The synthetic experiments are presented in Section 7.2 and the benchmark experiments are presented in Section 7.3.

7.1 Implementation

In this subsection, we first describe the problem to be solved and then explain the implementation of the above mentioned algorithms in detail.

For all algorithms except for IRS, the initial solution $\mathbf{w}^0$ was set to an all zero vector. For IRS, the initial solution was sampled from an independent standard Gaussian distribution.

The CPU time was measured on a Linux server with two 3.1 GHz Opteron Processors and 32GB of RAM.

7.1.1 ℓ_1-REGULARIZED LOGISTIC REGRESSION

The logistic regression model is defined by the loss function

$$f_{\mathrm{LR}}(\mathbf{z}) = \sum_{i=1}^{m} \ell_{\mathrm{LR}}(z_i, y_i) := \sum_{i=1}^{m} \log(1 + e^{-y_i z_i}), \tag{50}$$

where $y_i \in \{-1, +1\}$ is a training label. The conjugate of the loss function can be obtained as follows:

$$f_{\mathrm{LR}}^{*}(-\alpha) = \sum_{i=1}^{m} \ell_{\mathrm{LR}}^{*}(-\alpha_i, y_i),$$

where

$$\ell_{\mathrm{LR}}^{*}(-\alpha_i, y_i) = \begin{cases} \alpha_i y_i \log(\alpha_i y_i) + (1-\alpha_i y_i)\log(1-\alpha_i y_i) & (\text{if } 0 \le \alpha_i y_i \le 1), \\ +\infty & (\text{otherwise}). \end{cases}$$

Rewriting the dual problem (3) we have the following expression:

$$\underset{\alpha \in \mathbb{R}^m}{\text{maximize}} \quad -f_{\mathrm{LR}}^*(-\alpha), \tag{51}$$

$$\text{subject to} \quad \|\mathbf{A}^\top \alpha\|_\infty \leq \lambda, \tag{52}$$

where $\|\mathbf{y}\|_\infty = \max_{j=1,\ldots,n} |y_j|$ is the ℓ_∞-norm; note that the implicit constraint in Equation (3) (through the indicator function δ_λ^∞) is made explicit in Equation (52).

For the experiments in this section, we reparametrize the regularization constant λ as $\lambda = \lambda \|\mathbf{A}^\top \mathbf{y}\|_\infty$. The reason for this reparametrization is that for all $\lambda \geq 0.5$ the solution $\mathbf{w}$ can be shown to be zero; thus we can measure the strength of the regularization relative to the problem using λ instead of λ. This is because the conjugate loss function f_{LR}^* takes the minimum at $\alpha_i = y_i/2$ and the minimum is attained for $\lambda \geq \|\mathbf{A}^\top(\mathbf{y}/2)\|_\infty$ (see Equation 52).

7.1.2 DUALITY GAP

We used the relative duality gap (RDG) as a stopping criterion with tolerance 10^{-3}. More specifically, we terminated all the algorithms described below when RDG fell below 10^{-3}. RDG was computed as follows for all algorithms except L1_LOGREG. For L1_LOGREG, we modified the stopping criterion implemented in the original code by the authors from absolute duality gap to relative duality gap. See also Koh et al. (2007), Wright et al. (2009) and Tomioka and Sugiyama (2009).

Let α^t be any candidate dual vector at tth iteration. For example, $\alpha^t = \alpha^t$ for DAL and $\alpha^t = -\nabla f_\ell(\mathbf{A}\mathbf{w}^{t+1})$ for OWLQN, SpaRSA, and IRS. Note that the above α^t does not necessarily satisfy the dual constraint (52). Thus we define $\tilde{\alpha}^t = \alpha^t \min(1, \lambda/\|\mathbf{A}^\top \alpha^t\|_\infty)$. Notice that $\|\mathbf{A}^\top \tilde{\alpha}^t\|_\infty \leq \lambda$ by construction. We compute the dual objective value as $d(\mathbf{w}^{t+1}) = -f_\ell^*(-\tilde{\alpha}^t)$; see Equation (51). Finally RDG^{t+1} is obtained as $\mathrm{RDG}^{t+1} = (f(\mathbf{w}^{t+1}) - d(\mathbf{w}^{t+1}))/f(\mathbf{w}^{t+1})$, where f is the primal objective function defined in Equation (1).

The norm of the minimum norm subgradient is also frequently used as a stopping criterion. However, there are two reasons for using RDG instead. First, the gradient at the current point is not evaluated in FISTA (Beck and Teboulle, 2009) and it requires additional computation, whereas the vector α^t in the computation of RDG does not need to be the gradient at the current point; in fact the gradient at any point (or any m-dimensional vector) gives a valid lower bound of the minimum objective value. Second, since the gradient can change discontinuously at nondifferentiable points, the norm of gradient does not reflect the distance from the solution well; this is a problem for, for example, an interior-point method, because it produces a sparse solution only asymptotically.

7.1.3 DAL

DAL algorithm is implemented in MATLAB.[8] The inner minimization problem (see Equation 10) is solved with Newton's method; we used the preconditioned conjugate gradient (PCG) method for solving the associated Newton system (`pcg` function in MATLAB); we use the diagonal elements of the Hessian matrix (see Equation 25) as the preconditioner. The inner minimization is terminated by the criterion (44) with $\gamma = 4$, because the Hessian of the loss function (50) is uniformly bounded by $1/4$ (see Table 1).

8. The software is available from `http://www.ibis.t.u-tokyo.ac.jp/ryotat/dal/`.

We chose the initial proximity parameter to be either $\eta_0 = 0.01/\lambda$ (conservative setting) or $\eta_0 = 1/\lambda$ (aggressive setting) and increased η_t by a factor of 2 at every iteration. Since η_t appears in the soft-thresholding operation multiplied by λ, it seems to be intuitive to choose η_t inversely proportional to λ but we do not have a formal argument yet. We empirically discuss the choice of η_0 in more detail in Section 7.2.4.

The algorithm was terminated when the RDG fell below 10^{-3}.

7.1.4 DAL-B

DAL-B is a variant of DAL with an unregularized bias term (see update Equations 35-37). This algorithm is included because L1_LOGREG implemented by Koh et al. (2007) estimates a bias term and therefore cannot be directly compared to DAL.

As an augmented Lagrangian algorithm, DAL-B solves the following dual problem:

$$\begin{aligned}
\underset{\alpha\in\mathbb{R}^m}{\text{maximize}} \quad & -f_{\mathrm{LR}}^*(-\alpha) - \delta_\lambda^\infty(\mathbf{v}), \\
\text{subject to} \quad & \mathbf{A}^\top\alpha = \mathbf{v}, && (53)\\
& \mathbf{1}^\top\alpha = 0. && (54)
\end{aligned}$$

See also Equations (3) and (4).

When implementing DAL-B, we noticed that sometimes the algorithm gets stuck in a plateau where the additional equality constraint (54) improves very little. This was more likely to happen when the condition of the design matrix was poor.

In order to avoid this undesirable slow-down, we heuristically adapt the proximity parameter η_t for the equality constraint (54). Note that this kind of modification cannot improve the theoretical convergence result without additional prior information. More specifically, we use proximity parameters $\eta_t^{(1)}$ and $\eta_t^{(2)}$ for equality constraints (53) and (54), respectively. The AL function (37) is rewritten as follows

$$\alpha^t = \underset{\alpha\in\mathbb{R}^m}{\operatorname{argmin}} \left(f_\ell^*(-\alpha) + \frac{1}{2\eta_t^{(1)}} \|\operatorname{prox}_{\lambda\eta_t}^{\ell_1}(\mathbf{w}^t + \eta_t^{(1)}\mathbf{A}^\top\alpha)\|^2 + \frac{1}{2\eta_t^{(2)}}(b^t + \eta_t^{(2)}\mathbf{1}_m{}^\top\alpha)^2 \right).$$

First we initialize $\eta_0^{(1)} = \eta_0^{(2)} = 0.01/\lambda$ (conservative setting) or $\eta_0^{(1)} = \eta_0^{(2)} = 1/\lambda$ (aggressive setting) as above. The proximity parameter $\eta_t^{(1)}$ with respect to Equation (53) is increased by the factor 2 at every iteration (the same as DAL). The proximity parameter $\eta_t^{(2)}$ with respect to Equation (54) is increased by a larger factor 40 if the following conditions are satisfied:

1. The iteration counter $t > 1$.
2. The violation of the equality constraint (54), namely $\mathrm{viol}^t := |\mathbf{1}^\top\alpha^t|$, does not sufficiently decrease; that is, $\mathrm{viol}^t > \mathrm{viol}^{t-1}/2$.
3. The violation viol^t is larger than 10^{-3} (the tolerance of optimization).

Otherwise, $\eta_t^{(2)}$ is increased by the same factor 2 as $\eta_t^{(1)}$.

Note that the theoretical results in Section 5 still holds if we replace η_t in Section 5 by $\eta_t^{(1)}$, because $\eta_t^{(1)} \le \eta_t^{(2)}$; that is, the stopping criterion (44) and the convergence rates simply become more conservative.

Since DAL-B has an additional equality constraint (54). We modified the computation of relative duality gap described above by defining the candidate dual vector α^t as $\alpha^t = \alpha^t - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m{}^\top\alpha^t$.

7.1.5 FAST ITERATIVE SHRINKAGE-THRESHOLDING ALGORITHM (FISTA)

FISTA algorithm (Beck and Teboulle, 2009) is implemented in MATLAB. The algorithm is terminated by the same RDG criterion except that the dual objective is evaluated at $\mathbf{y}^t$ in update equation (49) instead of $\mathbf{w}^{t+1}$; this approach saves unnecessary computation of gradients.

7.1.6 ORTHANT-WISE LIMITED MEMORY QUASI NEWTON (OWLQN)

OWLQN algorithm (Andrew and Gao, 2007) is also implemented in MATLAB because we found that our MATLAB implementation was faster than the C++ implementation provided by the authors; this is because MATLAB uses optimized linear algebra routines while authors' implementation does not. The algorithm is terminated by the same RDG criterion as DAL.

7.1.7 SPARSE RECONSTRUCTION BY SEPARABLE APPROXIMATION (SPARSA)

SpaRSA algorithm (Wright et al., 2009) is implemented in MATLAB. We modified the code provided by the authors[9] to handle the logistic loss function. The algorithm is terminated by the same RDG criterion.

7.1.8 ITERATIVE REWEIGHTED SHRINKAGE (IRS)

IRS algorithm is implemented in MATLAB. At every iteration IRS solves a ridge-regularized logistic regression problem with the regularizer defined in Equation (48). This problem can be converted into a standard ℓ_2-regularized logistic regression with the design matrix $\tilde{\mathbf{A}} = \mathbf{A}\mathrm{diag}(\sqrt{\alpha_1}, \ldots, \sqrt{\alpha_n})$ by reparametrizing w_j to $\tilde{w}_j = w_j/\sqrt{\alpha_j}$. The weight α_j is set to $|w_j^t|$ before solving the problem. Thus if any $w_j^t = 0$, the corresponding column of $\tilde{\mathbf{A}}$ becomes zero and it can be removed from the optimization. We use the limited memory BFGS quasi-Newton method (Nocedal and Wright, 1999) to solve each sub-problem.

7.1.9 INTERIOR POINT ALGORITHM (L1_LOGREG)

L1_LOGREG algorithm (Koh et al., 2007) is implemented in C. We modified the code provided by the authors[10] as a C-MEX function so that it can be called directly from MATLAB without saving matrices into files. We used the BLAS and LAPACK libraries provided together with MATLAB R2008b (`-lmwblas` and `-lmwlapack` options for the `mex` command). L1_LOGREG is also terminated by the RDG criterion.

Note that L1_LOGREG also estimates an unregularized bias term. DAL algorithm with a bias term (DAL-B) is included to make the comparison easy; see Section 7.1.4.

9. Code can be found at `http://www.lx.it.pt/~mtf/SpaRSA/`.
10. Code can be found at `http://www.stanford.edu/~boyd/l1_logreg/`.

7.2 Synthetic Experiment

In this subsection, we first confirm the convergence behaviour of DAL (Section 7.2.2); second we compare the scaling of various algorithms against the size of the problem (Section 7.2.3); finally we discuss how to choose the initial proximity parameter η_0 (Section 7.2.4).

7.2.1 EXPERIMENTAL SETTING

The elements of the design matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ were randomly sampled from an independent standard Gaussian distribution. The true classifier coefficient β was generated by filling randomly chosen element (4%) of a n-dimensional vector with samples from an independent standard Gaussian distribution; the remaining elements of the vector were set to zero. The training label vector $\mathbf{y}$ was obtained by taking the sign of $\mathbf{A}\beta + 0.01\xi$, where $\xi \in \mathbb{R}^m$ was a sample from an m-dimensional independent standard Gaussian distribution. The whole procedure was repeated ten times.

7.2.2 EMPIRICAL VALIDATION OF SUPER-LINEAR CONVERGENCE

In this section, we empirically confirm the validity of the convergence results (Theorems 2, 3 and 5) obtained in the previous section and compare the efficiency of DAL, FISTA, OWLQN, SpaRSA, and IRS for the number of samples $m = 1,024$ and the number of parameters $n = 16,384$. L1_LOGREG is not included because it solves a different minimization problem. We use the regularization constant $\lambda = 0.01$. For DAL, we used the aggressive setting ($\eta_t = 1/\lambda, 2/\lambda, 4/\lambda, \ldots$).

First in order to obtain the true minimizer[11] $\mathbf{w}^*$ of Equation (1), we ran DAL algorithm to obtain a solution with high precision (RDG $< 10^{-9}$). Assuming that the support of this solution is correct, we performed one Newton step of Equation (1) in the subspace of active variables. The solution $\mathbf{w}^*$ we obtained in this way satisfied $\|\nabla f(\mathbf{w}^*)\| < 10^{-13}$, where $\nabla f(\mathbf{w}^*)$ is the minimum norm subgradient of f at $\mathbf{w}^*$. The parameter σ in Equation (41) was estimated by taking the minimum of $(f(\mathbf{w}^t) - f(\mathbf{w}^*))/\|\mathbf{w}^t - \mathbf{w}^*\|^2$ along the trajectory obtained by the above minimization and multiplying the minimum value by a safety factor of 0.7. In order to estimate the residual norm $\|\mathbf{w}^t - \mathbf{w}^*\|$, we use bounds (42) and (46) with $\alpha = 2$ and the initial residual $\|\mathbf{w}^0 - \mathbf{w}^*\|$. The bound (40) in Theorem 2 is used with the same initial residual to estimate the reduction in the function value.

In Figure 2, we show a result of a typical (single) run of the algorithms described above. Note that the result is not averaged to keep the meaning of theoretical bounds.

In the top left panel of Figure 2, we can see that the convergence in terms of the norm of the residual vector $\mathbf{w}^t - \mathbf{w}^*$ happens indeed rapidly as predicted by the theorems in Section 5. The yellow curve shows the result of Theorem 3, which assumes exact minimization of Equation (20), and the magenta curve shows the result of Theorem 5, which allows some error in the minimization of Equation (20). We can see that the difference between the optimistic analysis of Theorem 3 and the realistic analysis of Theorem 5 is negligible. In this problem, in order to reach the quality of solution DAL achieves in 10 iterations OWLQN and SpaRSA take at least 100 iterations and FISTA takes 1,000 iterations. The IRS approach required about the same number of iterations as OWLQN and SpaRSA but each step was much heavier than those two algorithms (see also the top right panel in Figure 2) and it was terminated after 100 iterations.

11. We assume that the minimizer is unique.

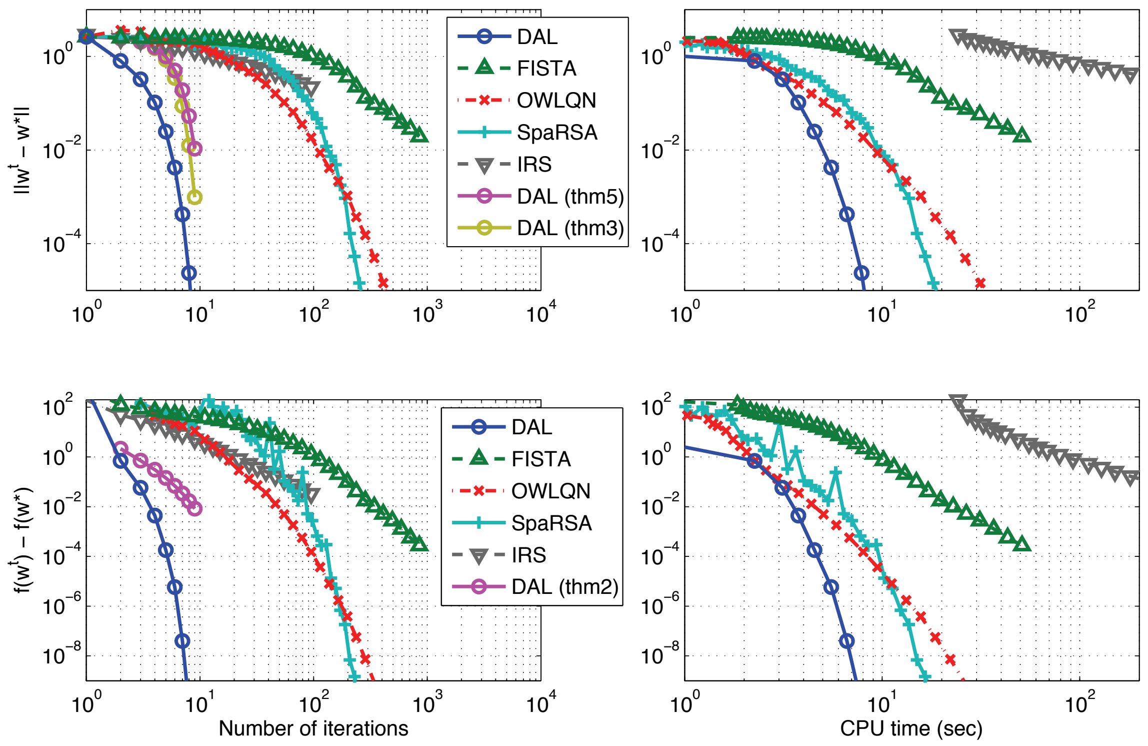


Figure 2: Empirical comparison of DAL, FISTA (Beck and Teboulle, 2009), OWLQN (Andrew and Gao, 2007), and SpaRSA (Wright et al., 2009). Top left: residual norm vs. number of iterations. Also the theoretical guarantees for DAL from Theorems 3 and 5 are shown. Top right: residual norm vs. CPU time. Bottom left: residual in the function value vs. number of iterations. Bottom right: residual in the function value vs. CPU time.

The bottom left panel of Figure 2 shows comparison of five algorithms DAL, FISTA, OWLQN, SpaRSA, and IRS in terms of the decrease in the function value. Also plotted is the decrease in the function value predicted by Theorem 2 (magenta curve). The convergence of DAL is the fastest also in terms of function value. OWLQN and SpaRSA are the next after DAL and are faster than FISTA.

DAL needs to solve a minimization problem at every iteration. Accordingly the operation required in each iteration is heavier than those in FISTA, OWLQN, and SpaRSA. Thus we compare the total CPU time spent by the algorithms in the right part of Figure 2. It can be seen that DAL can obtain a solution that is much more accurate in less than 10 seconds than the solution FISTA obtained after almost 60 seconds. In terms of computation time, DAL and OWLQN seem to be on par at low precision. However as the precision becomes higher DAL becomes clearly faster than OWLQN. SpaRSA seems to be slightly slower than DAL and OWLQN.

Two algorithms (DAL-B and L1_LOGREG) that also estimate an unregularized bias term are compared in Figure 3. The number of observations $m = 1{,}024$ and the number of parameters $n = 16{,}384$, and all other settings are identical as above. A variant of DAL-B that does not use the heuristics described in Section 7.1.4 is included for comparison. For DAL-B without the heuristics,

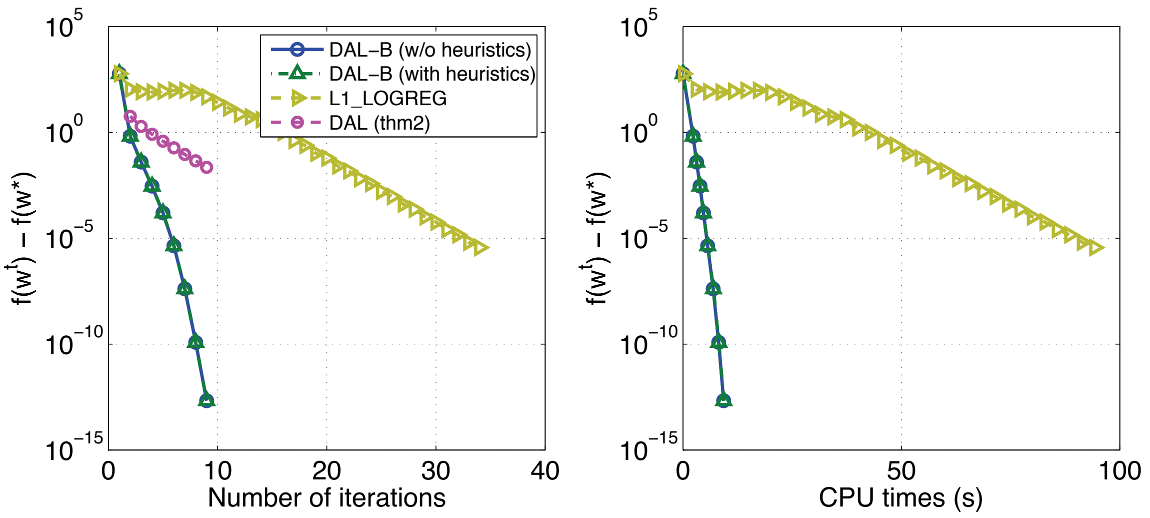


Figure 3: Comparison of DAL-B and L1_LOGREG (Koh et al., 2007). Both algorithms estimate an unregularized bias term. The left panel shows the residual function value against the number of iterations. The right panel shows the same against the CPU time spent by the algorithms.

the proximity parameters $\eta_t^{(1)}$ and $\eta_t^{(2)}$ are both initialized to $1/\lambda$ and increased by the factor 2. For DAL-B with the heuristics, the proximity parameter $\eta_t^{(2)}$ is increased more aggressively; see Section 7.1.4.

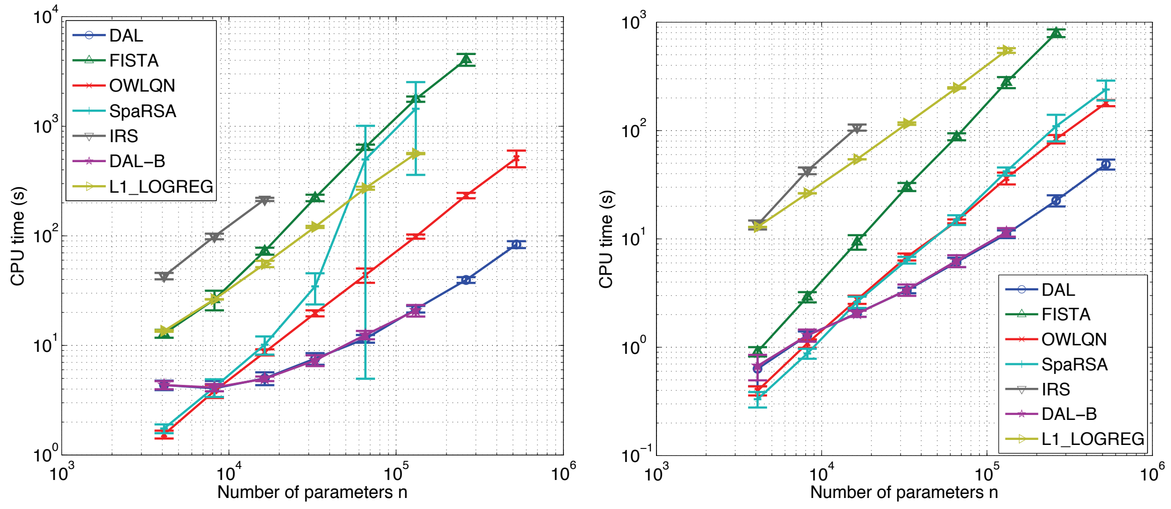
In the left panel in Figure 3, the residual of primal objective values of both algorithms are plotted against the number of iterations. As empirically observed in Koh et al. (2007), L1_LOGREG converges linearly; after roughly 10 iterations, the residual function value reduces by a factor around 2 in each iteration (a factor 1.85 was reported in Koh et al., 2007). The convergence of DAL-B is faster than L1_LOGREG and the curve is slightly concave downwards, which indicates the super-linearity of the convergence. Note also that the linear convergence bound from Theorem 2 is shown. The heuristics described in Section 7.1.4 shows almost no effect on this problem, probably because the design matrix is well conditioned.

The right panel in Figure 3 shows the same information against the CPU time spent by the algorithms. DAL-B is roughly 10 times faster than L1_LOGREG to achieve residual less than 10^{-5}.

7.2.3 Scaling Against the Size of the Problem

Here we compare how well different algorithms scale against the number of parameters n. We fixed the number of samples m at $m = 1{,}024$ and varied the number of parameters from $n = 4{,}096$ to $n = 524{,}288$. We used two regularization constants $\lambda = 0.1$ and $\lambda = 0.01$.

The results are summarized in Figure 4. Figures 4(a) and 4(b) show the results for $\lambda = 0.01$ and $\lambda = 0.1$, respectively. In each figure we plot the CPU time spent to reach RDG$< 10^{-3}$ against the number of parameters n.



(a) $\lambda = 0.01$. (b) $\lambda = 0.1$.

Figure 4: CPU time of various algorithms on synthetic logistic regression problems.

One can see that DAL has the mildest dependence on the number of parameters among the methods compared. In particular, DAL is faster than other algorithms for roughly $n > 10^4$. Also note that DAL and DAL-B show similar scaling against the number of parameters; that is, adding an unregularized bias term has no significant influence on the computational efficiency.

For $\lambda = 0.01$, SpaRSA shows sharp increase in the CPU time from around $n = 32{,}768$, which is similar to the result in Tomioka and Sugiyama (2009) (Figure 3). Also notice the increased errorbar. In fact, for $n \geq 65{,}536$, it had to be stopped after 5,000 iterations in some runs, whereas it converged after few hundred iterations in other runs. On the other hand, SpaRSA scales similarly to OWLQN and is more stable for $\lambda = 0.1$.

For all algorithms except L1_LOGREG, solving the problem for larger regularization constant $\lambda = 0.1$ requires less computation than for $\lambda = 0.01$. Nevertheless the advantage of the DAL algorithm is larger for the more computationally demanding situation of $\lambda = 0.01$ against FISTA, OWLQN, SpaRSA, and IRS. On the other hand, the advantage of DAL against L1_LOGREG is larger for $\lambda = 0.1$, because the CPU time of L1_LOGREG is almost constant in both cases. The CPU time of DAL with (DAL-B) and without (DAL) the bias term are almost the same.

7.2.4 CHOICE OF η_0

In this subsection, we show how the choice of the sequence η_t changes the behaviour of DAL algorithm. We ran DAL algorithm for $\lambda = 0.1$ with $\eta_0 = 1/\lambda$ (as above), which we call the aggressive setting, and $\eta_0 = 0.01/\lambda$, which we call the conservative setting. In both cases, η_t is increased by a factor of 2 as in the previous experiments. No bias term is used.

In Figure 5, plotted are the number of PCG steps for inner minimization and the CPU time spent by DAL algorithm with the conservative setting ($\eta_0 = 0.01/\lambda$, left) and the aggressive setting ($\eta_0 = 1/\lambda$, right). The average number of PCG steps and CPU time are shown as stacked barplots, in which each segment of a bar corresponds to one outer iteration. One can see that in the

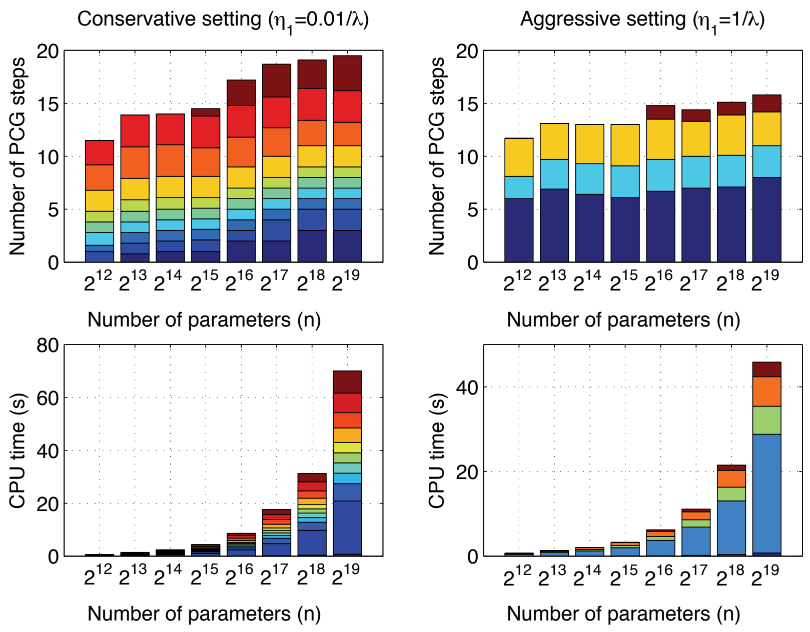


Figure 5: Comparison of behaviours of DAL algorithm for different choices of initial proximity parameter η_0. Left: $\eta_0 = 0.01/\lambda$ (conservative setting). Right: $\eta_0 = 1/\lambda$ (aggressive setting). On the top row, the cumulative numbers of PCG steps (inner steps) are shown. On the bottom row, the cumulative CPU time spent by the algorithm is shown.

conservative setting, DAL uses roughly 8 to 10 outer iterations, whereas in the aggressive setting, the number of outer iterations is reduced to less than a half (3 or 4). On the other hand, the total number of PCG steps is only slightly smaller in the aggressive setting. Therefore, in the aggressive setting DAL spends more PCG steps for each outer iteration. It is worth noting that almost half of the PCG iterations are spent for the first outer iteration in the aggressive setting, whereas in the conservative setting the PCG steps are more distributed. In terms of the CPU time, the aggressive setting is about 10–30% faster than the conservative setting because it saves both computation required for each outer-iteration and inner-iteration. However, generally speaking increasing the proximity parameter η_t makes the condition of the problem worse; in fact we found that the algorithm did not always converge for $\eta_0 = 100/\lambda$. Thus it is not recommended to use too large value for η_t.

Figure 6 compares the total CPU time spent by the two variants of DAL for $\lambda = 0.1$. As discussed above, the aggressive setting ($\eta_0 = 1/\lambda$) is faster than the conservative setting ($\eta_0 = 0.01/\lambda$). However the difference is minor compared to the change in the proximity parameter η_0,

7.3 Benchmark Data Sets

In this subsection, we apply the algorithms discussed in the previous subsection except IRS to benchmark data sets, and compare their efficiency on various problems; IRS is omitted because it was clearly outperformed by other methods on the synthetic data.

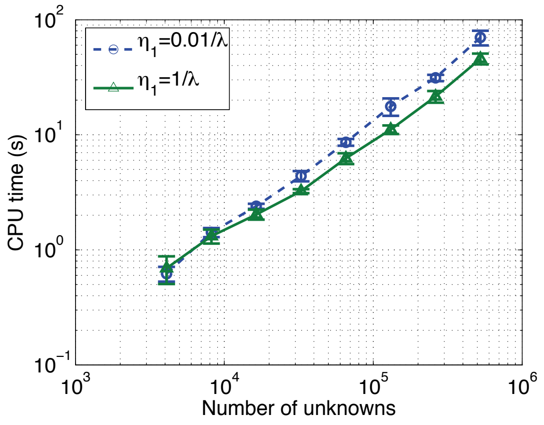


Figure 6: Comparison of conservative ($\eta_0 = 0.01/\lambda$) and aggressive ($\eta_0 = 1/\lambda$) choice of proximity parameter η_0 for $\lambda = 0.1$ (see also Figure 4(b)). Note that the aggressive setting is used in Sections 7.2.2 and 7.2.3 and the conservative setting is used in Section 7.3.

7.3.1 EXPERIMENTAL SETTING

The benchmark data sets we use are five data sets from NIPS 2003 Feature Selection Challenge,[12] 20 newsgroups data set,[13] and a bioinformatics data[14] provided by Baranzini et al. (2004)

The five data sets from the Feature Selection Challenge (**arcene, dexter**, **dorothea**, **gisette**, and **madelon**) are all split into training-, validation-, and test-set. We combine the training- and validation-sets and randomly split each data set into a training-set that contains two-thirds of the examples, and a test-set that contains the remaining one-third. We apply the ℓ_1-regularized logistic regression solvers to the training-set and report the accuracy on the test-set as well as the CPU time for training. This procedure was repeated 10 times (also for the two other data sets below). The numbers of training instances and features, and the format of each data set (sparse or dense) are summarized in Table 4.

From the 20 newsgroups data set (**20news**), we deal with the binary classification of category "alt.atheism" vs. "comp.graphics". We use the preprocessed MATLAB format data. The original data set consists of 1,061 training examples and 707 test examples. We again combine all the examples and randomly split them into a training-set containing two-thirds of the examples and a test-set containing the rest. The training example has $n = 61,188$ features which are provided as a sparse matrix.

The goal in Baranzini et al. (2004) is to predict the response (good or poor) to recombinant human interferon beta (rIFNβ) treatment of multiple sclerosis patients from gene-expression measurements. The data set is denoted as **gene**. The data set consists of gene-expression profile of 70 genes from 52 subjects. We again randomly select two-thirds of the subjects for training and the

12. The data sets are available from `http://www.nipsfsc.ecs.soton.ac.uk/datasets/`; see Guyon et al. (2006) for more information.
13. The data set is available from `http://people.csail.mit.edu/jrennie/20Newsgroups/`.
14. The data is available from `http://www.plosbiology.org/article/info:doi/10.1371/journal.pbio.0030002`.

remaining for testing. Following the setting in the original paper, we used only the expression data from the beginning of the treatment ($t = 0$) and preprocessed the data by taking all the polynomials up to third order, that is, we compute (i) x, x^2, and x^3 for each single feature x, (ii) xy, x^2y, and xy^2 for every pair of features (x,y), and (iii) xyz for every triplet of features (x,y,z). As a result we obtain 62,195 $(= 3 \cdot 70 + 3 \cdot 2{,}415 + 54{,}740)$ features.

In every data set, we standardized each feature to zero mean and unit standard deviation before applying the algorithms. Since the standardized design matrix $\tilde{\mathbf{A}}$ is usually dense even if the original matrix $\mathbf{A}$ is sparse, we provide function handles that compute $\tilde{\mathbf{A}}\mathbf{x}$ and $\tilde{\mathbf{A}}^\top\mathbf{y}$ instead of $\tilde{\mathbf{A}}$ itself with DAL, FISTA, OWLQN, and SpaRSA. This can be done by keeping the vector of means and standard deviations of the original design matrix as follows:

$$\tilde{\mathbf{A}}\mathbf{x} = \mathbf{A}\mathbf{S}^{-1}\mathbf{x} - \mathbf{1}_m\mathbf{m}^\top\mathbf{S}^{-1}\mathbf{x},$$
$$\tilde{\mathbf{A}}^\top\mathbf{y} = \mathbf{S}^{-1}\mathbf{A}^\top(\mathbf{y} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m{}^\top\mathbf{y}),$$

where $\mathbf{m} \in \mathbb{R}^n$ is the vector of means and $\mathbf{S}$ is a $n \times n$ diagonal matrix that has the standard deviations of the original features on the diagonal. If the standard deviation of any feature is zero, we placed one in the corresponding element of $\mathbf{S}$. L1_LOGREG is implemented with a similar technique; see Koh et al. (2007).

We compare the CPU time that is necessary to compute the whole regularization path. In order to define the regularization path, we choose 20 log-linearly separated values from $\lambda = 0.5$ to $\lambda = 0.001$, where λ is the normalized regularization constant defined in Section 7.1.1. We apply a warm start strategy to all the algorithms; that is, we sequentially solve problems for smaller and smaller regularization constants using the solution obtained from the last optimization (for a larger regularization constant) as the initial solution.

All the methods were terminated when the relative duality gap fell below 10^{-3}. For DAL algorithms (DAL and DAL-B) we choose the conservative setting, that is, we initialize $\eta_0^{(1)}$ and $\eta_0^{(2)}$ as $0.01/\lambda$.

7.3.2 RESULTS

Table 4 summarizes the problems and the performance of the algorithms. For each algorithm, we show the maximum test accuracy obtained in the regularization path and the CPU time spent to compute the whole path. The smallest and the second smallest CPU times are shown in bold-face and italic, respectively. One can see that DAL is the fastest in most cases when the number of parameters n is larger than the number of observations. In addition, the CPU time of two variants of DAL (with and without the bias term) tend to be similar except **dorothea** data set. For most data sets, the accuracy obtained by DAL algorithm is close to FISTA, OWLQN, and SpaRSA, and the accuracy obtained by DAL-B is close to L1_LOGREG.

Figure 7 illustrates a typical situation where DAL algorithm is efficient. Since the size of the inner minimization problem (20) is proportional to the number of observations m, when $n \gg m$, DAL is more efficient than other methods that work in the primal.

In contrast, Figure 8 illustrates the situation where DAL is not very efficient compared to other algorithms. In Figure 8, we can also see that for all algorithms except L1_LOGREG, the cost of solving one minimization problem grows larger as the regularization constant is reduced, whereas the cost seems almost constant for L1_LOGREG.

		arcene	dexter	dorothea	gisette	madelon	20news	gene
m		133	400	767	4667	1733	1179	35
n		10000	20000	100000	5000	500	61188	62195
format		dense	sparse	sparse	dense	dense	sparse	dense
DAL	accuracy	70.60	91.75	93.71	97.70	61.53	92.84	82.35
	time (s)	**3.47**	*4.20*	36.61	*77.02*	16.73	*28.10*	*5.56*
DAL-B	accuracy	72.54	92.00	93.05	97.71	61.43	92.87	81.18
	time (s)	*3.56*	*4.77*	**10.60**	*82.96*	17.96	*26.31*	**5.49**
FISTA	accuracy	70.60	91.75	93.79	97.71	61.51	92.80	82.35
	time (s)	25.34	7.24	284.59	**52.19**	*10.40*	*27.95*	108.27
OWLQN	accuracy	70.60	91.75	93.76	97.70	61.56	92.82	82.35
	time (s)	17.63	*5.25*	134.31	*70.96*	19.08	*23.11*	132.21
SpaRSA	accuracy	70.90	91.75	93.71	97.70	61.55	95.14	78.24
	time (s)	294.80	29.98	1377.20	*91.65*	*10.11*	310.96	1622.26
L1_LOG-REG	accuracy	72.84	92.05	93.05	97.71	61.48	92.85	81.18
	time (s)	8.98	**3.39**	109.92	*98.37*	**5.90**	**21.48**	16.58

Table 4: Results on benchmark data sets. We tested six algorithms, namely, DAL, DAL-B, FISTA, OWLQN, SpaRSA, L1_LOGREG on seven benchmark data sets. See main text for the description of the data sets. m is the number of observations. n is the number of features. For each algorithm, shown are the test accuracy and the CPU time spent to compute the regularization path with a warm-start strategy. All the numbers are averaged over 10 runs. Bold face numbers indicate the fastest CPU time. Italic numbers indicate CPU times that are within two times of the fastest CPU time.

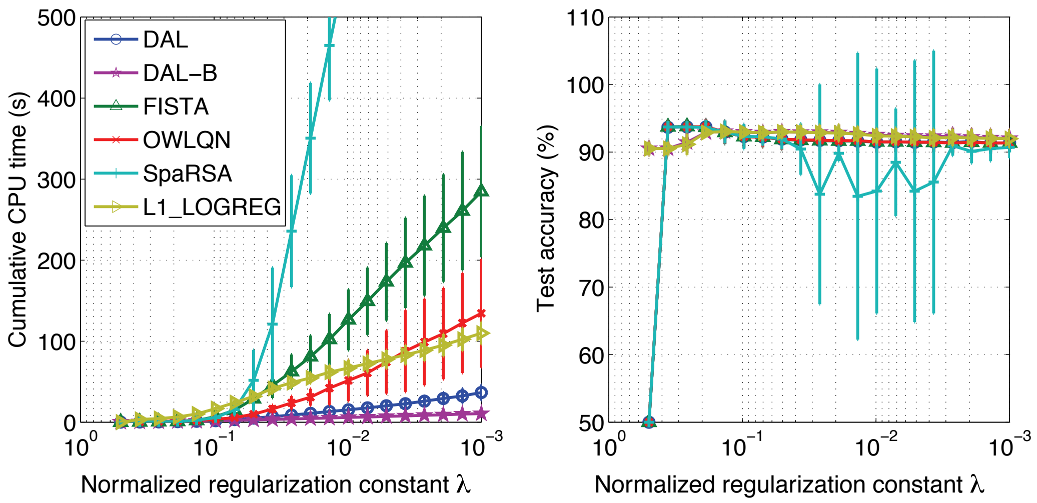


Figure 7: **Dorothea** data set ($m = 767, n = 100{,}000$). DAL is efficient in this case ($m \ll n$).

8. Conclusion

In this paper, we have extended DAL algorithm (Tomioka and Sugiyama, 2009) for general regularized minimization problems, and provided it with a new view based on the proximal minimization

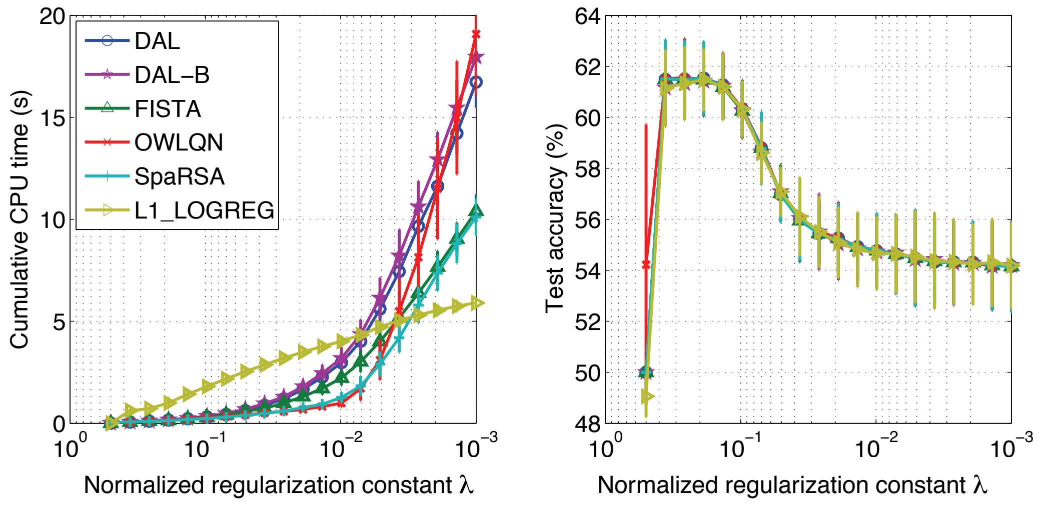


Figure 8: **Madelon** data set ($m = 1{,}733$, $n = 500$). DAL is not very efficient in this case ($m \gg n$).

framework in Rockafellar (1976b). Generalizing the recent result from Beck and Teboulle (2009), we improved the convergence results on super-linear convergence of augmented Lagrangian methods in literature for the case of sparse estimation.

Importantly, most assumptions that we made in our analysis can be checked independent of data. Instead of assuming that the problem is strongly convex we assume that the loss function has a Lipschitz continuous gradient, which can be checked before receiving data. Another assumption we have made is that the proximation with respect to the regularizer can be computed analytically, which can also be checked without looking at data. Moreover, we have shown that such assumption is valid for the ℓ_1-regularizer, group lasso regularizer, and any other support function of some convex set for which the projection onto the set can be analytically obtained.

Compared to the general result in Rockafellar (1976b), our result is stronger when the inner minimization is solved approximately. Compared to Kort and Bertsekas (1976), we do not need to assume the strong convexity of the objective function, which is obviously violated for the dual of many sparsity regularized estimation problems; instead we assume that the loss function has Lipschitz continuous gradient. Note that we use no asymptotic arguments as in Rockafellar (1976b) and Kort and Bertsekas (1976). Currently, our results does not apply to primal-based augmented Lagrangian method discussed in Goldstein and Osher (2009) for loss functions that are not strongly convex (e.g., logistic loss). The extension of our analysis to these methods is a future work.

The theoretically predicted rapid convergence of DAL algorithm is also empirically confirmed in simulated ℓ_1-regularized logistic regression problems. Moreover, we have compared six recently proposed algorithms for ℓ_1-regularized logistic regression, namely DAL, FISTA, OWLQN, SpaRSA, L1_LOGREG, and IRS on synthetic and benchmark data sets. On the synthetic data sets, we have shown that DAL has the mildest dependence on the number of parameters among the methods compared. On the benchmark data sets, we have shown that DAL is the fastest among the methods compared when the number of parameters is larger than the number of observations on both sparse and dense data sets.

Furthermore, we have empirically investigated the relationship between the choice of the initial proximity parameter η_0 and the number of (inner/outer) iterations as well as the computation time. We found that the computation can be sped up by choosing a large value for η_0; however the improvement is often small compared to the change in η_0 and choosing large value for η_0 can make the inner minimization unstable by making the problem poorly conditioned.

There are basically two strategies to make an efficient optimization algorithm. One is to use many iterations that are very light. FISTA, SpaRSA, and OWLQN (and also stochastic approaches Shalev-Shwartz and Srebro, 2008; Duchi and Singer, 2009) fall into this category. Theoretical convergence guarantee is often weak for these methods, for example, $O(1/k^2)$ for FISTA. Another strategy is to use a small number of heavier iterations. Interior point methods, such as L1_LOGREG, are prominent examples of this class. DAL can be considered as a member of the second class. We have theoretically and empirically shown that DAL requires a small number of outer iterations. At the same time, DAL inherits good properties of iterative shrinkage/thresholding algorithms from the first class. For example, it effectively uses the fact that the proximal operation can be computed analytically, and it can maintain the sparsity of the parameter vector during optimization. Furthermore, we have shown that the dual formulation of DAL makes the inner minimization efficient, because (i) typically the number of observations m is smaller than the number of parameters n, and (ii) the gradient and Hessian of the inner objective can be computed efficiently for sparse estimation problems.

Future work includes the extension of our analysis to the primal-based augmented Lagrangian methods (Yin et al., 2008; Goldstein and Osher, 2009; Yang and Zhang, 2009; Lin et al., 2009), application of approximate augmented Lagrangian methods and operator splitting methods to machine learning problems (see Zhang et al., 2010; Boyd et al., 2011; Tomioka et al., 2011b), and application of DAL to more advanced sparse estimation problems (e.g., Cai et al., 2008; Wipf and Nagarajan, 2008; Tomioka et al., 2011a).

Acknowledgments

We would like to thank Masakazu Kojima, Masao Fukushima, and Hisashi Kashima for helpful discussions. This work was partially supported by the Global COE program ”The Research and Training Center for New Development in Mathematics”, MEXT KAKENHI 22700138, 22700289, and the FIRST program.

Appendix A. Preliminaries on Proximal Operation

This section contains some basic results on proximal operation, which we use in later sections and is based on Moreau (1965), Rockafellar (1970), and Combettes and Wajs (2005).

A.1 Proximal Operation

Let f be a closed proper convex function over $\mathbb{R}^n$ that takes values in $\mathbb{R} \cup \{+\infty\}$. The *proximal operator* with respect to f is defined as follows:

$$\mathrm{prox}_f(\mathbf{z}) = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} \left(f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2 \right). \tag{55}$$

Note that because of the strong convexity of the minimand in the right-hand side, the above minimizer is unique. Similarly we define the proximal operator with respect to the convex conjugate function f^* of f as follows:

$$\mathrm{prox}_{f^*}(\mathbf{z}) = \underset{\mathbf{x}\in\mathbb{R}^n}{\mathrm{argmin}} \left(f^*(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2 \right).$$

The following elegant result is well known.

Lemma 8 (Moreau's decomposition) *The proximation of a vector $\mathbf{z}\in\mathbb{R}^n$ with respect to a convex function f and that with respect to its convex conjugate f^* is complementary in the following sense:*

$$\mathrm{prox}_f(\mathbf{z}) + \mathrm{prox}_{f^*}(\mathbf{z}) = \mathbf{z}.$$

Proof The proof can be found in Moreau (1965) and Rockafellar (1970, Theorem 31.5). Here, we present a slightly more simple proof.

Let $\mathbf{x} = \mathrm{prox}_f(\mathbf{z})$ and $\mathbf{y} = \mathrm{prox}_{f^*}(\mathbf{z})$. By definition we have $\partial f(\mathbf{x}) + \mathbf{x} - \mathbf{z} \ni 0$ and $\partial f^*(\mathbf{y}) + \mathbf{y} - \mathbf{z} \ni 0$, which imply

$$\partial f(\mathbf{x}) \ni \mathbf{z}-\mathbf{x}, \tag{56}$$

$$\partial f^*(\mathbf{z}-\mathbf{x}) \ni \mathbf{x}, \tag{57}$$

and

$$\partial f^*(\mathbf{y}) \ni \mathbf{z}-\mathbf{y}, \tag{58}$$

$$\partial f(\mathbf{z}-\mathbf{y}) \ni \mathbf{y}, \tag{59}$$

respectively, because $(\partial f)^{-1} = \partial f^*$ (Rockafellar, 1970, Corollary 23.5.1).

From Equations (56) and (58), we have

$$f(\mathbf{z}-\mathbf{y}) \geq f(\mathbf{x}) + (\mathbf{z}-\mathbf{y}-\mathbf{x})^\top(\mathbf{z}-\mathbf{x}), \tag{60}$$

$$f^*(\mathbf{z}-\mathbf{x}) \geq f^*(\mathbf{y}) + (\mathbf{z}-\mathbf{x}-\mathbf{y})^\top(\mathbf{z}-\mathbf{y}). \tag{61}$$

Similarly, Equations (57) and (59) give

$$f^*(\mathbf{y}) \geq f^*(\mathbf{z}-\mathbf{x}) + (\mathbf{y}-\mathbf{z}+\mathbf{x})^\top\mathbf{x}, \tag{62}$$

$$f(\mathbf{x}) \geq f(\mathbf{z}-\mathbf{y}) + (\mathbf{x}-\mathbf{z}+\mathbf{y})^\top\mathbf{y}. \tag{63}$$

Summing both sides of Equations (60)–(63), we have

$$0 \geq 2\|\mathbf{z}-\mathbf{x}-\mathbf{y}\|^2,$$

from which we conclude that $\mathbf{x}+\mathbf{y}=\mathbf{z}$. ■

Proximal operation can be considered as a generalization of the projection onto a convex set. For example, if we take f as the indicator function of the ℓ_∞ ball of radius λ, that is, $f(\mathbf{z}) = \delta_\lambda^\infty(\mathbf{z})$ (see Equation 5), then the proximal operation with respect to f is the projection onto the ℓ_∞-ball (8). On the other hand, the proximal operation with respect to the ℓ_1-regularizer is the soft-thresholding operator (9). Therefore, we notice that

$$\mathrm{proj}_{[-\lambda,\lambda]}(\mathbf{z}) + \mathrm{prox}_\lambda^{\ell_1}(\mathbf{z}) = \mathbf{z},$$

which is a special case of Lemma 8, because the ℓ_1-regularizer is the convex conjugate of the indicator function of the ℓ_∞-ball of radius λ; see Figure 9.

A.2 Moreau's Envelope

The minimum attained in Equation (55) is called the Moreau envelope of f:

$$F(\mathbf{z}) = \min_{\mathbf{x}\in\mathbb{R}^n} \left(f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2 \right). \tag{64}$$

The decomposition in Lemma 8 can be expressed in terms of envelope functions as follows.

Lemma 9 (Decomposition and envelope functions) *Let f and f^* be a pair of convex conjugate functions, and let F and F^* be the Moreau envelopes of f and f^*, respectively. Then we have*

$$F(\mathbf{z}) + F^*(\mathbf{z}) = \frac{1}{2}\|\mathbf{z}\|^2.$$

Proof Let $\mathbf{x} = \mathrm{prox}_f(\mathbf{z})$ and $\mathbf{y} = \mathrm{prox}_{f^*}(\mathbf{z})$ as in the proof of Lemma 8. From the definition of convex conjugate f^*, we have

$$f(\mathbf{x}) + f^*(\mathbf{y}) = \mathbf{y}^\top \mathbf{x},$$

because $\mathbf{y} = \mathbf{z} - \mathbf{x} \in \partial f(\mathbf{x})$ (Rockafellar, 1970, Theorem 23.5). Therefore, we have

$$\begin{aligned} F(\mathbf{z}) + F^*(\mathbf{z}) &= f(\mathbf{x}) + \frac{1}{2}\|\mathbf{y}\|^2 + f^*(\mathbf{y}) + \frac{1}{2}\|\mathbf{x}\|^2 \\ &= \mathbf{y}^\top\mathbf{x} + \frac{1}{2}\|\mathbf{y}\|^2 + \frac{1}{2}\|\mathbf{x}\|^2 \\ &= \frac{1}{2}\|\mathbf{x}+\mathbf{y}\|^2 = \frac{1}{2}\|\mathbf{z}\|^2, \end{aligned}$$

where we used $\mathbf{x}+\mathbf{y} = \mathbf{z}$ from Lemma 8 in the last line. ■

Note that F^* is the Moreau envelope of f^* and *not* the convex conjugate of F.

Moreau's envelope can be considered as a inf-convolution (see Rockafellar, 1970) of f and a quadratic function $\|\cdot\|^2/2$. Accordingly it is differentiable and the derivative is given in the following lemma.

Lemma 10 (Derivative of Moreau's envelope) *Moreau's envelope function F in Equation* (64) *is continuously differentiable (even if f is not differentiable) and the derivative can be written as follows:*

$$\nabla F(\mathbf{z}) = \mathrm{prox}_{f^*}(\mathbf{z}).$$

Proof The proof can be found in Moreau (1965) and Rockafellar (1970, Theorem 31.5). We repeat the proof below for completeness. The proof consists of two parts. We first show that for all $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^n$

$$F(\mathbf{z}') \geq F(\mathbf{z}) + (\mathbf{z}'-\mathbf{z})^\top \mathbf{y}, \tag{65}$$

where $\mathbf{y} = \mathrm{prox}_{f^*}(\mathbf{z})$, which implies that $\mathbf{y} = \mathrm{prox}_{f^*}(\mathbf{z}) \in \partial F(\mathbf{z})$. Second, we show that

$$F(\mathbf{z}') \leq F(\mathbf{z}) + (\mathbf{z}'-\mathbf{z})^\top \mathbf{y} + \frac{\|\mathbf{z}'-\mathbf{z}\|^2}{2}, \tag{66}$$

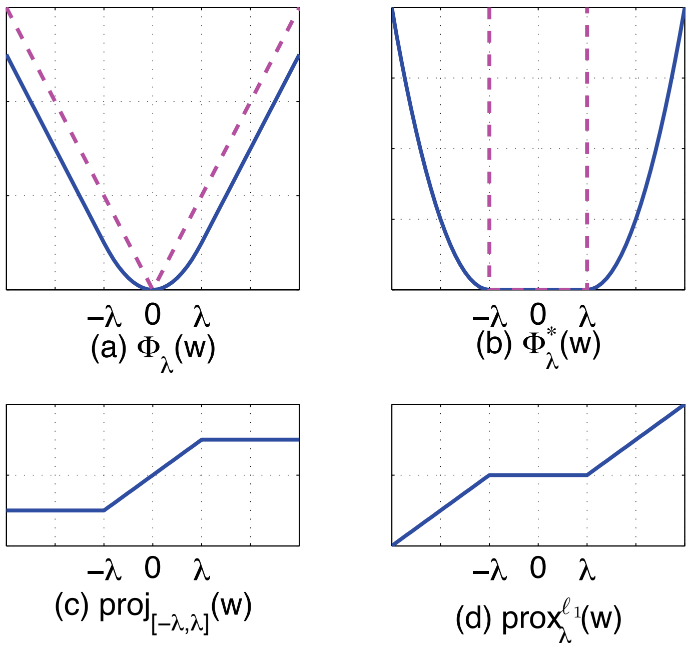


Figure 9: (a) The ℓ_1-regularizer (dashed) and its envelope function Φ_λ (solid). (b) The indicator function δ_λ^∞ (dashed) and its envelope function Φ_λ^* (solid). (c) The derivative of Φ_λ, which is the projection onto the interval $[-\lambda, \lambda]$; see Equation (8). (d) The derivative of Φ_λ^*, which is called the soft-threshold function (9). Note that the ℓ_1-regularizer and the indicator function δ_λ^∞ are conjugate to each other.

which implies the uniqueness of the subgradient of $F(\mathbf{z})$ for all $\mathbf{z}$.

Inequality (65) follows easily from the definition of the envelope function F and Lemma 8 as follows:

$$\begin{aligned}F(\mathbf{z}') - F(\mathbf{z}) &= f(\mathbf{x}') + \frac{1}{2}\|\mathbf{y}'\|^2 - f(\mathbf{x}) - \frac{1}{2}\|\mathbf{y}\|^2 \\ &= \Big(f(\mathbf{x}') - f(\mathbf{x})\Big) + \Big(\frac{1}{2}\|\mathbf{y}'\|^2 - \frac{1}{2}\|\mathbf{y}\|^2\Big) \\ &\geq (\mathbf{x}' - \mathbf{x})^\top \mathbf{y} + (\mathbf{y}' - \mathbf{y})^\top \mathbf{y} \\ &= (\mathbf{z}' - \mathbf{z})^\top \mathbf{y},\end{aligned}$$

where $\mathbf{x} = \mathrm{prox}_f(\mathbf{z})$, $\mathbf{y} = \mathrm{prox}_{f^*}(\mathbf{z})$, and $\mathbf{x}'$ and $\mathbf{y}'$ are similarly defined. We used the convexity of f with $\mathbf{y} \in \partial f(\mathbf{x})$ and the convexity of $\|\cdot\|^2/2$ in the third line.

Second, we obtain inequality (66) by upper-bounding $F(\mathbf{z}')$ as follows:

$$\begin{aligned}
F(\mathbf{z}') &= \min_{\xi\in\mathbb{R}^n}\left(f(\xi)+\frac{1}{2}\|\xi-\mathbf{z}'\|^2\right)\\
&\leq f(\mathbf{x})+\frac{1}{2}\|\mathbf{x}-\mathbf{z}'\|^2\\
&= F(\mathbf{z})-\frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2+\frac{1}{2}\|\mathbf{x}-\mathbf{z}'\|^2\\
&= F(\mathbf{z})-\frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2+\frac{1}{2}\|\mathbf{x}-\mathbf{z}+\mathbf{z}-\mathbf{z}'\|^2\\
&= F(\mathbf{z})+(\mathbf{x}-\mathbf{z})^\top(\mathbf{z}-\mathbf{z}')+\frac{1}{2}\|\mathbf{z}'-\mathbf{z}\|^2\\
&= F(\mathbf{z})+\mathbf{y}^\top(\mathbf{z}'-\mathbf{z})+\frac{1}{2}\|\mathbf{z}'-\mathbf{z}\|^2.
\end{aligned}$$

■

The envelope functions of two convex functions $\phi_\lambda(w)=\lambda|w|$ and $\phi_\lambda^*(w)=\delta_\lambda^\infty(w)$, and their derivatives (the projection (8) and the soft-threshold function (9), respectively) are schematically shown in Figure 9.

Appendix B. Derivation of Equation (20)

$$\begin{aligned}
\text{Equation (18)} &= \max_{\alpha\in\mathbb{R}^m}\Bigg\{-f_\ell^*(-\alpha)+\min_{\mathbf{w}\in\mathbb{R}^n}\left(\phi_\lambda(\mathbf{w})+\frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}^t-\eta_t\mathbf{A}^\top\alpha\|^2\right)\\
&\qquad -\frac{1}{2\eta_t}\|\mathbf{w}^t+\eta_t\mathbf{A}^\top\alpha\|^2\Bigg\}+\frac{\|\mathbf{w}^t\|^2}{2\eta_t}\\
&= \max_{\alpha\in\mathbb{R}^m}\left(-f_\ell^*(-\alpha)+\frac{1}{\eta_t}\Phi_{\lambda\eta_t}(\mathbf{w}^t+\eta_t\mathbf{A}^\top\alpha)-\frac{1}{2\eta_t}\|\mathbf{w}^t+\eta_t\mathbf{A}^\top\alpha\|^2\right)+\frac{\|\mathbf{w}^t\|^2}{2\eta_t}\\
&= \max_{\alpha\in\mathbb{R}^m}\left(-f_\ell^*(-\alpha)-\frac{1}{\eta_t}\Phi^*_{\lambda\eta_t}(\mathbf{w}^t+\eta_t\mathbf{A}^\top\alpha)\right)+\frac{\|\mathbf{w}^t\|^2}{2\eta_t},
\end{aligned}$$

where we used the definition of the Moreau envelope in the second line and Lemma 9 in the third line. Finally omitting the constant term $\|\mathbf{w}^t\|^2/(2\eta_t)$ in the last line and reversing the sign we obtain Equation (20).

Appendix C. Proofs

In this section, we present the proofs of Theorem 3, Lemma 4, Theorem 6, and Theorem 7.

C.1 Proof of Theorem 3

Proof The first step of the proof is to generalize Lemma 1 in two ways: first we allow a point $\mathbf{w}^*$ in the set of minimizers W^* to be chosen for each time step, and second, we introduce a parameter μ to tighten the bound. Let $\mathbf{w}^t$ be the closest point from $\mathbf{w}^t$ in W^*, namely $\mathbf{w}^t := \mathrm{argmin}_{\mathbf{w}^* \in W^*} \|\mathbf{w}^t - \mathbf{w}^*\|$. From the proof of Lemma 1, we have

$$\begin{aligned}
\eta_t(f(\mathbf{w}^{t+1}) - f(\mathbf{w}^{t+1})) &\geq \left\langle \mathbf{w}^{t+1} - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^{t+1} \right\rangle \\
&= \left\langle \mathbf{w}^{t+1} - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^t + \mathbf{w}^t - \mathbf{w}^{t+1} + \mathbf{w}^{t+1} - \mathbf{w}^{t+1} \right\rangle \\
&= \|\mathbf{w}^{t+1} - \mathbf{w}^{t+1}\|^2 + \left\langle \mathbf{w}^{t+1} - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^t \right\rangle + \left\langle \mathbf{w}^{t+1} - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^{t+1} \right\rangle \\
&\geq \|\mathbf{w}^{t+1} - W^*\|^2 - \|\mathbf{w}^{t+1} - W^*\| \|\mathbf{w}^t - W^*\| \\
&\geq \|\mathbf{w}^{t+1} - W^*\|^2 - \left(\frac{\mu}{2} \|\mathbf{w}^{t+1} - W^*\|^2 + \frac{1}{2\mu} \|\mathbf{w}^t - W^*\|^2 \right) \quad (\forall \mu > 0) \\
&= \left(1 - \frac{\mu}{2}\right) \|\mathbf{w}^{t+1} - W^*\|^2 - \frac{1}{2\mu} \|\mathbf{w}^t - W^*\|^2, \qquad (\star)
\end{aligned}$$

where the last inner product $\left\langle \mathbf{w}^{t+1} - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^{t+1} \right\rangle$ in the third line is non-negative because the set of minimizers W^* is a convex set, and $\mathbf{w}^{t+1}$ is the projection of $\mathbf{w}^{t+1}$ onto W^*; see Bertsekas (1999, Proposition B.11). In addition, the fifth line follows from the inequality of arithmetic and geometric means.

Note that by setting $\mu = 1$ in $(\star)$ and $\mathbf{w}^t = \mathbf{w}^{t+1}$, we recover Lemma 1. Now using assumption **(A1)**, we obtain the following expression:

$$\left(2\mu - \mu^2\right) \|\mathbf{w}^{t+1} - W^*\|^2 + 2\mu\sigma\eta_t \|\mathbf{w}^{t+1} - W^*\|^\alpha \leq \|\mathbf{w}^t - W^*\|^2.$$

Maximizing the left hand side with respect to μ, we have $\mu = 1 + \sigma\eta_t \|\mathbf{w}^{t+1} - W^*\|^{\alpha-2}$ and accordingly,

$$\left(1 + \sigma\eta_t \|\mathbf{w}^{t+1} - W^*\|^{\alpha-2}\right)^2 \|\mathbf{w}^{t+1} - W^*\|^2 \leq \|\mathbf{w}^t - W^*\|^2.$$

Taking the square-root of both sides we obtain

$$\|\mathbf{w}^{t+1} - W^*\| + \sigma\eta_t \|\mathbf{w}^{t+1} - W^*\|^{\alpha-1} \leq \|\mathbf{w}^t - W^*\|. \tag{67}$$

The last part of the theorem is obtained by lower-bounding the left-hand side of the above inequality using Young's inequality as follows:

$$\begin{aligned}
&\|\mathbf{w}^{t+1} - W^*\| + \sigma\eta_t \|\mathbf{w}^{t+1} - W^*\|^{\alpha-1} \\
&= (1 + \sigma\eta_t) \left(\frac{1}{1 + \sigma\eta_t} \|\mathbf{w}^{t+1} - W^*\| + \frac{\sigma\eta_t}{1 + \sigma\eta_t} \|\mathbf{w}^{t+1} - W^*\|^{\alpha-1} \right) \\
&\geq (1 + \sigma\eta_t) \|\mathbf{w}^{t+1} - W^*\|^{\frac{1}{1+\sigma\eta_t}} \cdot \|\mathbf{w}^{t+1} - W^*\|^{\frac{(\alpha-1)\sigma\eta_t}{1+\sigma\eta_t}} \\
&= (1 + \sigma\eta_t) \|\mathbf{w}^{t+1} - W^*\|^{\frac{1+(\alpha-1)\sigma\eta_t}{1+\sigma\eta_t}}.
\end{aligned}$$

Substituting this relation back into inequality (67) completes the proof of the theorem. ■

C.2 Proof of Lemma 4

Proof First let us define $\delta^t \in \mathbb{R}^m$ as the gradient of the AL function (22) at the approximate minimizer α^t follows:

$$\delta^t := \nabla \varphi_t(\alpha^t) = -\nabla f_\ell^*(-\alpha^t) + \mathbf{A}\mathbf{w}^{t+1},$$

where $\mathbf{w}^{t+1} := \mathrm{prox}_{\phi_{\lambda\eta_t}}(\mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha^t)$. Note that $\|\delta^t\| \leq \sqrt{\frac{\gamma}{\eta_t}}\|\mathbf{w}^{t+1} - \mathbf{w}^t\|$ from Assumption **(A4)**. Using Corollary 23.5.1 from Rockafellar (1970), we have

$$\nabla f_\ell(\mathbf{A}\mathbf{w}^{t+1} - \delta^t) = \nabla f_\ell\left(\nabla f_\ell^*(-\alpha^t)\right) = -\alpha^t, \tag{68}$$

which implies that if δ^t is small, $-\alpha^t$ is approximately the gradient of the loss term at $\mathbf{w}^{t+1}$.

Moreover, let $\mathbf{q}^t = \mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha^t$. Since $\mathbf{w}^{t+1} = \mathrm{prox}_{\phi_{\lambda\eta_t}}(\mathbf{q}^t)$ (Assumption **A3**), we have

$$\partial \phi_{\lambda\eta_t}(\mathbf{w}^{t+1}) + (\mathbf{w}^{t+1} - \mathbf{q}^t) \ni 0,$$

which implies

$$(\mathbf{q}^t - \mathbf{w}^{t+1})/\eta_t \in \partial \phi_\lambda(\mathbf{w}^{t+1}), \tag{69}$$

because $\phi_{\lambda\eta_t} = \eta_t \phi_\lambda$.

Now we are ready to derive an analogue of inequality (39) in the proof of Lemma 1. Let $\mathbf{w} \in \mathbb{R}^n$ be an arbitrary vector. We can decompose the residual value in the left hand side of inequality (39) as follows:

$$\begin{aligned}\eta_t(f(\mathbf{w}) - f(\mathbf{w}^{t+1})) = \eta_t(&\underbrace{f_\ell(\mathbf{A}\mathbf{w}) - f_\ell(\mathbf{A}\mathbf{w}^{t+1} - \delta^t)}_{\text{(A)}}) \\ &+ \eta_t(\underbrace{f_\ell(\mathbf{A}\mathbf{w}^{t+1} - \delta^t) - f_\ell(\mathbf{A}\mathbf{w}^{t+1})}_{\text{(B)}}) \\ &+ \eta_t(\underbrace{\phi_\lambda(\mathbf{w}) - \phi_\lambda(\mathbf{w}^{t+1})}_{\text{(C)}}).\end{aligned}$$

The above terms (A), (B), and (C) can be separately bounded using the convexity of f_ℓ and ϕ_λ as follows:

$$\text{(A)}: \quad f_\ell(\mathbf{A}\mathbf{w}) - f_\ell(\mathbf{A}\mathbf{w}^{t+1} - \delta^t) \geq \left\langle \mathbf{A}(\mathbf{w} - \mathbf{w}^{t+1}) + \delta^t, -\alpha^t \right\rangle,$$

$$\text{(B)}: \quad f_\ell(\mathbf{A}\mathbf{w}^{t+1} - \delta^t) - f_\ell(\mathbf{A}\mathbf{w}^{t+1}) \geq -\left\langle \delta^t, -\alpha^t \right\rangle - \frac{1}{2\gamma}\|\delta^t\|^2,$$

$$\text{(C)}: \quad \phi_\lambda(\mathbf{w}) - \phi_\lambda(\mathbf{w}^{t+1}) \geq \left\langle \mathbf{w} - \mathbf{w}^{t+1}, \frac{\mathbf{w}^t + \eta_t \mathbf{A}^\top \alpha^t - \mathbf{w}^{t+1}}{\eta_t} \right\rangle,$$

where (A) is due to Equation (68), (B) is due to assumption **(A2)** and Hiriart-Urruty and Lemaréchal (1993, Theorem X.4.2.2), and (C) is due to Equation (69). Combining (A), (B), and (C), we have the following expression:

$$\eta_t(f(\mathbf{w}) - f(\mathbf{w}^{t+1})) \geq \left\langle \mathbf{w}^t - \mathbf{w}^{t+1}, \mathbf{w} - \mathbf{w}^{t+1} \right\rangle - \frac{\eta_t}{2\gamma}\|\delta^t\|^2.$$

Note that the above inequality reduces to Equation (39) if $\|\delta_t\| = 0$ (exact minimization). Using assumption **(A4)**, we obtain

$$\begin{aligned}\eta_t(f(\mathbf{w}) - f(\mathbf{w}^{t+1})) &\geq \left\langle \mathbf{w}^t - \mathbf{w} + \mathbf{w} - \mathbf{w}^{t+1}, \mathbf{w} - \mathbf{w}^{t+1} \right\rangle - \frac{1}{2}\|\mathbf{w}^t - \mathbf{w}^{t+1}\|^2 \\ &= \frac{1}{2}\|\mathbf{w} - \mathbf{w}^{t+1}\|^2 - \frac{1}{2}\|\mathbf{w} - \mathbf{w}^t\|^2,\end{aligned}$$

which completes the proof. ■

C.3 Proof of Theorem 6

Proof Let $\delta = (1-\varepsilon)/(\sigma\eta_t)$. Note that $\delta \leq 3/4 < 1$ from the assumption. Following the proof of Lemma 4 with $\mathbf{w} = \mathbf{w}^t$, we have

$$\begin{aligned}&\eta_t(f(\mathbf{w}^{t+1}) - f(\mathbf{w}^{t+1})) \\ &= \eta_t(f(\mathbf{w}^t) - f(\mathbf{w}^{t+1})) \\ &\geq \left\langle \mathbf{w}^t - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^{t+1} \right\rangle - \frac{\delta}{2}\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 \\ &= \left\langle \mathbf{w}^t - \mathbf{w}^t + \mathbf{w}^t - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^{t+1} \right\rangle - \frac{1}{2}\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 + \frac{1-\delta}{2}\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 \\ &= \frac{1}{2}\|\mathbf{w}^t - \mathbf{w}^{t+1}\|^2 - \frac{1}{2}\|\mathbf{w}^t - \mathbf{w}^t\|^2 + \frac{1-\delta}{2}\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 \\ &\geq \frac{1}{2}\|\mathbf{w}^{t+1} - W^*\|^2 - \frac{1}{2}\|\mathbf{w}^t - W^*\|^2 \\ &\qquad + \frac{1-\delta}{2}\left(\|\mathbf{w}^{t+1} - W^*\|^2 + \|\mathbf{w}^t - W^*\|^2 + 2\left\langle \mathbf{w}^{t+1} - \mathbf{w}^{t+1}, \mathbf{w}^t - \mathbf{w}^t \right\rangle\right) \\ &\geq -(1-\delta)\|\mathbf{w}^{t+1} - W^*\|\|\mathbf{w}^t - W^*\| + \left(1 - \frac{\delta}{2}\right)\|\mathbf{w}^{t+1} - W^*\|^2 - \frac{\delta}{2}\|\mathbf{w}^t - W^*\|^2 \\ &\geq -(1-\delta)\left(\frac{1}{2\mu}\|\mathbf{w}^t - W^*\|^2 + \frac{\mu}{2}\|\mathbf{w}^{t+1} - W^*\|^2\right) \\ &\qquad + \left(1 - \frac{\delta}{2}\right)\|\mathbf{w}^{t+1} - W^*\|^2 - \frac{\delta}{2}\|\mathbf{w}^t - W^*\|^2 \quad (\forall \mu > 0),\end{aligned}$$

where we used $\|\mathbf{w}^{t+1} - \mathbf{w}^t\|^2 \geq 0$, $\left\langle \mathbf{w}^{t+1} - \mathbf{w}^{t+1}, \mathbf{w}^{t+1} - \mathbf{w}^t \right\rangle \geq 0$ and $\left\langle \mathbf{w}^{t+1} - \mathbf{w}^t, \mathbf{w}^t - \mathbf{w}^t \right\rangle \geq 0$ in the sixth line; the seventh line follows from Cauchy-Schwartz inequality; the eighth line follows from the inequality of arithmetic and geometric means.

Applying assumption **(A1)** with $\alpha = 2$ to the above expression, we have

$$\frac{1-\delta}{2\mu}\|\mathbf{w}^t - W^*\|^2 \geq -\frac{1-\delta}{2}\mu\|\mathbf{w}^{t+1} - W^*\|^2 + \left(\left(1 - \tfrac{\delta}{2}\right) + \sigma\eta_t\right)\|\mathbf{w}^{t+1} - W^*\|^2 - \frac{\delta}{2}\|\mathbf{w}^t - W^*\|^2.$$

Multiplying both sides with $\mu/\|\mathbf{w}^{t+1} - W^*\|^2$, we have

$$\frac{1-\delta}{2}\frac{\|\mathbf{w}^t - W^*\|^2}{\|\mathbf{w}^{t+1} - W^*\|^2} \geq -\frac{1-\delta}{2}\mu^2 + \left\{\left(1 - \frac{\delta}{2}\right) + \sigma\eta_t - \frac{\delta}{2}\frac{\|\mathbf{w}^t - W^*\|^2}{\|\mathbf{w}^{t+1} - W^*\|^2}\right\}\mu. \tag{70}$$

Now we have to consider two cases depending on the sign inside the curly brackets. If the sign is negative or zero, we have

$$1-\frac{\delta}{2}+\sigma\eta_t-\frac{\delta}{2}\frac{\|\mathbf{w}^t-W^*\|^2}{\|\mathbf{w}^{t+1}-W^*\|^2}\leq 0,$$

which implies

$$\|\mathbf{w}^{t+1}-W^*\|^2\leq\frac{\delta}{2-\delta+2\sigma\eta_t}\|\mathbf{w}^t-W^*\|^2. \tag{71}$$

Since $\delta\leq 3/4$, the factor in front of $\|\mathbf{w}^t-W^*\|^2$ in the right-hand side is clearly smaller than one. We further show that this factor is smaller than $1/(1+\varepsilon\sigma\eta_t)^2$. First we upper bound δ by $1/(1+\varepsilon\sigma\eta_t)$ as follows:

$$\delta=\frac{1-\varepsilon}{\sigma\eta_t}=\frac{(1-\varepsilon)(\frac{1}{\sigma\eta_t}+\varepsilon)}{1+\varepsilon\sigma\eta_t}\leq\frac{(1-\varepsilon)\varepsilon+3/4}{1+\varepsilon\sigma\eta_t}\leq\frac{1}{1+\varepsilon\sigma\eta_t}.$$

Plugging the above upper bound into inequality (71), we have

$$\begin{aligned}\|\mathbf{w}^{t+1}-W^*\|^2&\leq\frac{\delta}{1+2\sigma\eta_t}\|\mathbf{w}^t-W^*\|^2\\&\leq\frac{1}{(1+\varepsilon\sigma\eta_t)(1+2\sigma\eta_t)}\|\mathbf{w}^t-W^*\|^2\leq\frac{1}{(1+\varepsilon\sigma\eta_t)^2}\|\mathbf{w}^t-W^*\|^2,\end{aligned}$$

which completes the proof for the first case.

If on the other hand, the term inside the curly brackets is positive in Equation (70), maximizing the right-hand side of Equation (70) with respect to μ gives the following expression:

$$(1-\delta)r_t\geq 1-\frac{\delta}{2}+\sigma\eta_t-\frac{\delta}{2}r_t^2,$$

where we defined $r_t:=\|\mathbf{w}^t-W^*\|/\|\mathbf{w}^{t+1}-W^*\|$. Because $r_t>0$, the above inequality translates into

$$\begin{aligned}r_t&\geq\frac{\sqrt{1+2\sigma\eta_t\delta}-1+\delta}{\delta}\\&\geq\frac{1+\sigma\eta_t\delta-\sigma^2\eta_t^2\delta^2-1+\delta}{\delta}\\&\geq 1+\sigma\eta_t(1-\sigma\eta_t\delta)\\&=1+\varepsilon\sigma\eta_t.\end{aligned}$$

The second line is true because for $x\geq 0$, $\sqrt{1+x}\geq 1+x/2-x^2/4$; the last line follows from the definition $\delta=(1-\varepsilon)/(\sigma\eta_t)$. ■

C.4 Proof of Theorem 7

Proof Since the minimizer is unique, we denote the minimizer by $\mathbf{w}^*$ and show that the following is true:

$$f(\mathbf{w}) - f(\mathbf{w}^*) \geq \sigma\|\mathbf{w}-\mathbf{w}^*\|^2 \qquad (\forall \mathbf{w} : \|\mathbf{w}-\mathbf{w}^*\| \leq c\tau L).$$

Using Theorem X.4.2.2 in Hiriart-Urruty and Lemaréchal (1993), for $\|\beta\| \leq \tau$, we have

$$\begin{aligned} f^*(\beta) &\leq f^*(0) + \beta^\top \nabla f^*(0) + \frac{L}{2}\|\beta\|^2 \\ &= -f(\mathbf{w}^*) + \beta^\top \mathbf{w}^* + \frac{L}{2}\|\beta\|^2, \end{aligned} \tag{72}$$

where $\mathbf{w}^* := \mathrm{argmin}_{\mathbf{w}\in\mathbb{R}^n} f(\mathbf{w}) = \nabla f^*(0)$ and $f^*(0) = -f(\mathbf{w}^*)$.

On the other hand, we have

$$\begin{aligned} f(\mathbf{w}) &= \sup_{\beta\in\mathbb{R}^n} \left(\beta^\top \mathbf{w} - f^*(\beta)\right) \\ &\geq \sup_{\|\beta\|\leq\tau} \left(\beta^\top \mathbf{w} - f^*(\beta)\right) \\ &\geq \sup_{\|\beta\|\leq\tau} \left(\beta^\top (\mathbf{w}-\mathbf{w}^*) - \frac{L}{2}\|\beta\|^2\right) + f(\mathbf{w}^*) \\ &\begin{cases} = f(\mathbf{w}^*) + \frac{1}{2L}\|\mathbf{w}-\mathbf{w}^*\|^2 & (\text{if } c \leq 1), \\ \geq f(\mathbf{w}^*) + \frac{2c-1}{2c^2L}\|\mathbf{w}-\mathbf{w}^*\|^2 & (\text{otherwise}), \end{cases} \end{aligned}$$

where we used inequality (72) in the third line; the last line follows because if $c \leq 1$, the maximum is attained at $\beta = (\mathbf{w}-\mathbf{w}^*)/L$, and otherwise we can lower bound the value at the maximum by the value at $\beta = (\mathbf{w}-\mathbf{w}^*)/(cL)$. Combining the above two cases, we have Theorem 7. ■

References

G. Andrew and J. Gao. Scalable training of L1-regularized log-linear models. In *Proc. of the 24th international conference on Machine learning*, pages 33–40, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: http://doi.acm.org/10.1145/1273496.1273501.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in NIPS 19*, pages 41–48. MIT Press, Cambridge, MA, 2007.

A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 25–32. MIT Press, Cambridge, MA, 2008.

S. E Baranzini, P. Mousavi, J. Rio, S. J. Caillier, A. Stillman, P. Villoslada, M. M. Wyatt, M. Comabella, L. D. Greller, R. Somogyi, X. Montalban, and J. R. Oksenberg. Transcription-based prediction of response to ifnβ using supervised computational methods. *PLoS Biol.*, 3(1):e2, 2004.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999. 2nd edition.

P.J. Bickel, Y. Ritov, and A.B. Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009. ISSN 0090-5364.

J. M. Bioucas-Dias. Bayesian wavelet-based image deconvolution: A GEM algorithm exploiting a class of heavy-tailed priors. *IEEE Trans. Image Process.*, 15:937–951, 2006.

J. M. Bioucas-Dias and M. A. T. Figueiredo. A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Trans. Image Process.*, 16(12):2992–3004, 2007.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers, 2011. Unfinished working draft.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. ISSN 1064-8275.

J.-F. Cai, E. J. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. Technical report, arXiv:0810.3286, 2008. URL `http://arxiv.org/abs/0810.3286`.

P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pur. Appl. Math.*, LVII:1413–1457, 2004.

D. L. Donoho. De-noising by soft-thresholding. *IEEE Trans. Inform. Theory*, 41(3):613–627, 1995.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *J. Mach. Learn. Res.*, 10:2899–2934, 2009. ISSN 1532-4435.

B. Efron, T. Hastie, R. Tibshirani, and I. Johnstone. Least angle regression. *Annals of Statistics*, 32 (2):407–499, 2004.

M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proc. of the American Control Conference*, 2001.

M. A. T. Figueiredo and R. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Trans. Image Process.*, 12:906–916, 2003.

M. A. T. Figueiredo, J. M. Bioucas-Dias, and R. D. Nowak. Majorization-minimization algorithm for wavelet-based image restoration. *IEEE Trans. Image Process.*, 16(12), 2007a.

M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal on selected topics in Signal Processing*, 1(4):586–597, 2007b.

M. Girolami. A variational method for learning sparse and overcomplete representations. *Neural Computation*, 13(11):2517–2532, 2001.

T. Goldstein and S. Osher. The split Bregman method for L1 regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009. ISSN 1936-4954.

I. F. Gorodnitsky and B. D. Rao. Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm. *IEEE Trans. Signal Process.*, 45(3), 1997.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors. *Feature Extraction: Foundations and Applications*. Springer Verlag, 2006.

S. Haufe, R. Tomioka, G. Nolte, K.-R. Müller, and M. Kawanabe. Modeling sparse connectivity between underlying brain sources for EEG/MEG. *IEEE Trans. Biomed. Eng.*, 57(8):1954–1963, 2010. ISSN 0018-9294.

M. R. Hestenes. Multiplier and gradient methods. *J. Optim. Theory Appl.*, 4:303–320, 1969.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods*. Springer, 1993.

T. S. Jaakkola. *Variational methods for inference and estimation in graphical models*. PhD thesis, Massachusetts Institute of Technology, 1997.

S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinvesky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE journal of selected topics in signal processing*, 1:606–617, 2007.

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

B. W. Kort and D. P. Bertsekas. Combined primal–dual and penalty methods for convex programming. *SIAM Journal on Control and Optimization*, 14(2):268–294, 1976.

G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 2010. Under revision.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *J. Mach. Learn. Res.*, 10:777–801, 2009. ISSN 1532-4435.

Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of a corrupted low-rank matrices. *Mathematical Programming*, 2009. submitted.

P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006. ISSN 0090-5364.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

J. J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la S. M. F.*, 93:273–299, 1965.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical Report 2007/76, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, 2007.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

J. Palmer, D. Wipf, K. Kreutz-Delgado, and B. Rao. Variational EM algorithms for non-gaussian latent variable models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1059–1066. MIT Press, Cambridge, MA, 2006.

M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, London, New York, 1969.

A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *JMLR*, 9:2491–2521, 2008.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14:877–898, 1976a.

R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Math. of Oper. Res.*, 1:97–116, 1976b.

S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *Proc. of the 25th International Conference on Machine Learning*, pages 928–935. ACM, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proc. of the 24th International Conference on Machine Learning*, pages 807–814. ACM, 2007.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in NIPS 17*, pages 1329–1336. MIT Press, Cambridge, MA, 2005.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc. B*, 58(1):267–288, 1996.

R. Tomioka and K.-R. Müller. A regularized discriminative framework for EEG analysis with application to brain-computer interface. *Neuroimage*, 49(1):415–432, 2010.

R. Tomioka and M. Sugiyama. Dual augmented Lagrangian method for efficient sparse reconstruction. *IEEE Signal Processing Letters*, 16(12):1067–1070, 2009.

R. Tomioka and T. Suzuki. Regularization strategies and empirical Bayesian learning for MKL. Technical report, arXiv:1011.3090v1, 2010.

R. Tomioka, T. Suzuki, M. Sugiyama, and H. Kashima. A fast augmented Lagrangian algorithm for learning low-rank matrices. In *Proc. of the 27th International Conference on Machine Learning*. Omnipress, 2010.

R. Tomioka, K. Hayashi, and H. Kashima. Estimation of low-rank tensors via convex optimization. *SIAM J. Matrix Anal. A.*, 2011a. Submitted.

R. Tomioka, T. Suzuki, and M. Sugiyama. Augmented Lagrangian methods for learning, selecting, and combining features. In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for Machine Learning*. MIT Press, 2011b.

D. Wipf and S. Nagarajan. A new view of automatic relevance determination. In *Advances in NIPS 20*, pages 1625–1632. MIT Press, 2008.

S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Trans. Signal Process.*, 2009.

J. Yang and Y. Zhang. Alternating direction algorithms for L1-problems in compressive sensing. Technical Report TR09-37, Dept. of Computational & Applied Mathematics, Rice University, 2009.

W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for l1-minimization with applications to compressed sensing. *SIAM J. Imaging Sciences*, 1(1):143–168, 2008.

J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-newton approach to nonsmooth convex optimization problems in machine learning. *The Journal of Machine Learning Research*, 11:1145–1200, 2010.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Soc. B*, 68(1):49–67, 2006.

X. Zhang, M. Burger, and S. Osher. A unified primal-dual algorithm framework based on bregman iteration. *Journal of Scientific Computing*, 46(1):20–46, 2010.

P. Zhao and B. Yu. On model selection consistency of lasso. *J. Mach. Learn. Res.*, 7:2541–2563, 2006. ISSN 1532-4435.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B(Statistical Methodology)*, 67(2):301–320, 2005.

Journal of Machine Learning Research 12 (2011) 1587-1615 Submitted 9/09; Revised 9/10; Published 5/11

Double Updating Online Learning

Peilin Zhao ZHAO0106@NTU.EDU.SG
Steven C.H. Hoi CHHOI@NTU.EDU.SG
School of Computer Engineering
Nanyang Technological University
Singapore 639798

Rong Jin RONGJIN@CSE.MSU.EDU
Department of Computer Science & Engineering
Michigan State University
East Lansing, MI, 48824

Editor: Nicolò Cesa-Bianchi

Abstract

In most kernel based online learning algorithms, when an incoming instance is misclassified, it will be added into the pool of support vectors and assigned with a weight, which often remains unchanged during the rest of the learning process. This is clearly insufficient since when a new support vector is added, we generally expect the weights of the other existing support vectors to be updated in order to reflect the influence of the added support vector. In this paper, we propose a new online learning method, termed **Double Updating Online Learning**, or **DUOL** for short, that explicitly addresses this problem. Instead of only assigning a fixed weight to the misclassified example received at the current trial, the proposed online learning algorithm also tries to update the weight for one of the existing support vectors. We show that the mistake bound can be improved by the proposed online learning method. We conduct an extensive set of empirical evaluations for both binary and multi-class online learning tasks. The experimental results show that the proposed technique is considerably more effective than the state-of-the-art online learning algorithms. The source code is available to public at `http://www.cais.ntu.edu.sg/~chhoi/DUOL/`.

Keywords: online learning, kernel method, support vector machines, maximum margin learning, classification

1. Introduction

Online learning has been studied extensively in the machine learning community (Rosenblatt, 1958; Freund and Schapire, 1999; Kivinen et al., 2001; Crammer et al., 2006; Cesa-Bianchi and Lugosi, 2006). In general, for a misclassified example, most of the kernel based online learning algorithms will simply assign to it a fixed weight that remains unchanged during the whole learning process. Although such an approach is advantageous in computational efficiency, it has significant limitations. This is because when a new example is added to the pool of support vectors, the weights assigned to the existing support vectors may no longer be optimal, and should be updated to reflect the influence of the new support vector. We emphasize that although several online algorithms are proposed to update the example weights as the learning process proceeds, most of them are not designed to improve the classification accuracy. For instance, in Orabona et al. (2008) and Crammer et al. (2003); Dekel et al. (2008), online learning algorithms are proposed to adjust the example weights in order to fit in the constraint on the number of support vectors; in Kivinen et al. (2001), example weights are adjusted to deal with the drifting concepts.

Motivated by the above observations, we propose a new strategy for online learning that explicitly addresses this problem. It is designed to dynamically tune the weights of support vectors in order to improve the classification performance. In some trials of online learning, besides assigning a weight to the misclassified example, the proposed online learning algorithm also updates the weight for one of the existing support vectors, referred to as *auxiliary example*. We refer to the proposed approach as **Double Updating Online Learning** (Zhao et al., 2009), or **DUOL** for short.

The key challenge in the proposed online learning approach is to decide which existing support vector should be selected for updating weight. An intuitive choice is to select the existing support vector that "conflicts" with the new misclassified example, that is the existing support vector which on the one hand shares similar input pattern as the new example and on the other hand belongs to a class different from that of the new example. In order to quantitatively analyze the impact of updating the weight for such an existing support vector, we employ an analysis that is based on the work of online convex programming by incremental dual ascent (Shalev-Shwartz and Singer, 2006, 2007). Our analysis shows that under certain conditions, the proposed online learning algorithm can significantly reduce the mistake bound of the existing online algorithms. Besides binary classification, we extend the double updating online learning algorithm to multi-class learning. Extensive experiments show promising performance of the proposed online learning algorithm compared to the state-of-the-art algorithms for online learning.

The rest of this paper is organized as follows. Section 2 reviews the related work for online learning. Section 3 presents the proposed "double updating" approach for online learning of binary classification problems. Section 4 extends the double updating method to online multi-class learning. Section 5 gives our experimental results. Section 6 discusses the possible directions to explore in the future. Section 7 concludes this work.

2. Related Work

Online learning has been extensively studied in machine learning (Rosenblatt, 1958; Crammer and Singer, 2003; Cesa-Bianchi et al., 2004; Crammer et al., 2006; Fink et al., 2006). One of the most well-known online approaches is the Perceptron algorithm (Rosenblatt, 1958; Freund and Schapire, 1999), which updates the learning function by adding the misclassified example with a constant weight to the current set of support vectors. Recently a number of online learning algorithms have been developed based on the criterion of maximum margin (Crammer and Singer, 2003; Gentile, 2001; Kivinen et al., 2001; Crammer et al., 2006; Li and Long, 1999). One example is the Relaxed Online Maximum Margin algorithm (ROMMA) (Li and Long, 1999), which repeatedly chooses the hyper-planes that correctly classify the existing training examples with a large margin. Another representative example is the Passive-Aggressive (PA) algorithm (Crammer et al., 2006). It updates the classification function when a new example is misclassified or its classification score does not exceed the predefined margin. Empirical studies showed that the maximum margin based online learning algorithms are generally more effective than the Perceptron algorithm. Despite the difference, most online learningalgorithms only update the weight of the newly added support vector, and keep the weights of the existing support vectors unchanged. This constraint could significantly limit the performance of online learning.

The proposed online learning algorithm is closely related to the recent work of online convex programming by incremental dual ascent (Shalev-Shwartz and Singer, 2006, 2007). Although the idea of simultaneously updating the weights of multiple support vectors was mentioned in Shalev-Shwartz and Singer (2006, 2007), neither efficient algorithm nor theoretical result was given explicitly in their work. Besides, our work is also related to budget online learning (Weston and Bordes, 2005; Crammer et al., 2003; Cavallanti et al., 2007; Dekel et al., 2008) and online learning for drifting concepts. Although these online learning algorithms are capable of dynamically adjusting the weights of support vectors, they are designed to either fit in the budget for the number of support vectors or to handle drifting concepts, but not to reduce the number of classification mistakes in online learning.

Finally, several algorithms were proposed for online training of SVM that update the weights of more than one support vectors simultaneously (Cauwenberghs and Poggio, 2000; Bordes et al., 2005, 2007; Dredze et al., 2008; Crammer et al., 2008, 2009). In particular, in Bordes et al. (2005, 2007), the authors proposed to update the weights of two support vectors simultaneously at each iteration, similar to the proposed algorithm. These algorithms differ from the proposed one in that they are designed for efficiently learning an SVM classification model, not for online learning, and therefore do not provide guarantee for mistake bound.

3. Double Updating Online Learning for Binary Classification

In this section, we present the proposed double updating online learning method for solving online binary classification tasks. Below we start by introducing some preliminaries and notations.

3.1 Preliminaries and Notations

We consider the problem of online classification. Our goal is to learn a function $f : \mathbb{R}^d \to \mathbb{R}$ based on a sequence of training examples $\{(x_1, y_1), \ldots, (x_T, y_T)\}$, where $x_t \in \mathbb{R}^d$ is a d-dimensional instance and $y_t \in \mathcal{Y} = \{-1, +1\}$ is the class label assigned to x_t. We use $sign(f(x))$ to predict the class assignment for any x, and $|f(x)|$ to measure the classification confidence. Let $\ell(f(x), y) : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ be the loss function that penalizes the deviation of estimates $f(x)$ from observed labels y. We refer to the output f of the learning algorithm as a *hypothesis* and denote the set of all possible hypotheses by $\mathcal{H} = \{f | f : \mathbb{R}^d \to \mathbb{R}\}$.

In this paper, we consider $\mathcal{H}$ a Reproducing Kernel Hilbert Space (**RKHS**) endowed with a kernel function $\kappa(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (Vapnik, 1998) implementing the inner product$\langle \cdot, \cdot \rangle$ such that: 1) κ has the reproducing property $\langle f, \kappa(x, \cdot) \rangle = f(x)$ for $x \in \mathbb{R}^d$; 2) $\mathcal{H}$ is the closure of the span of all $\kappa(x, \cdot)$ with $x \in \mathbb{R}^d$, that is, $\kappa(x, \cdot) \in \mathcal{H}$ for every $x \in \mathcal{X}$. The inner product $\langle \cdot, \cdot \rangle$ induces a norm on $f \in H$ in the usual way: $\|f\|_{\mathcal{H}} := \langle f, f \rangle^{\frac{1}{2}}$. To make it clear, we use $\mathcal{H}_\kappa$ to denote an RKHS with explicit dependence on kernel function κ. Throughout the analysis, we assume $\kappa(x,x) \leq 1$ for any $x \in \mathbb{R}^d$.

3.2 Motivation

We consider trial t in an online learning task where the training example (x_a, y_a) is misclassified (i.e., $y_a f(x_a) \leq 0$)). Let $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, n\}$ be the collection of n misclassified examples received before the trial t. We also refer to these misclassified training examples as "support vectors". We denote by $\alpha = (\alpha_1, \ldots, \alpha_n) \in (0, C]^n$ the weights assigned to the support vectors in $\mathcal{D}$, where C is a predefined constant. The resulting classifier, denoted by $f(x)$, is given by

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \kappa(x, x_i).$$

In the conventional approach for online learning, we simply assign a constant weight, denoted by $\beta \in (0, C]$, to (x_a, y_a), and the resulting classifier becomes

$$f'(x) = \beta y_a \kappa(x, x_a) + \sum_{i=1}^{n} \alpha_i y_i \kappa(x, x_i) = \beta y_a \kappa(x, x_a) + f(x).$$

The shortcoming of the conventional online learning approach is that the introduction of the new support vector (x_a, y_a) may harm the classification of existing support vectors in $\mathcal{D}$, which is revealed by the following proposition.

Proposition 1 *Let (x_a, y_a) be an example misclassified by the current classifier $f(x) = \sum_{i=1}^{n} \alpha_i y_i \kappa(x, x_i)$ with $\alpha_i \geq 0, i = 1, \ldots, n$, that is, $y_a f(x_a) < 0$. Let $f'(x) = \beta y_a \kappa(x, x_a) + f(x)$ be the updated classifier with $\beta > 0$. There exists at least one support vector $x_i \in \mathcal{D}$ such that $y_i f(x_i) > y_i f'(x_i)$.*

Proof It follows from the fact that: $\exists\, x_i \in \mathcal{D}, y_i y_a \kappa(x_i, x_a) < 0$ when $y_a f(x_a) < 0$. ■

As indicated by Proposition 1, when a misclassified example (x_a, y_a) is added to the classifier, the classification confidence of at least one existing support vector will be reduced. When $y_a f(x_a) \leq -\gamma$, there exists one support vector $(x_b, y_b) \in \mathcal{D}$ that satisfies $\beta y_a y_b k(x_a, x_b) \leq -\beta\gamma/n$. This support vector will be misclassified by the updated classifier $f'(x)$ if $y_b f(x_b) \leq \beta\gamma/n$. In order to alleviate this problem, we propose to update the weight for the existing support vector whose classification confidence is significantly affected by the new misclassified example. In particular, we consider a support vector $(x_b, y_b) \in \mathcal{D}$ for weight updating if it satisfies the following two conditions:

- $y_b f(x_b) \leq 0$, that is, support vector (x_b, y_b) is misclassified by the current classifier $f(x)$;

- $k(x_b, x_a) y_a y_b \leq -\rho$ where $\rho \in (0, 1)$ is a predetermined threshold, that is, support vector (x_b, y_b) "**conflicts**" with the new misclassified example (x_a, y_a).

We refer to the support vector satisfying the above conditions as an **auxiliary example**. It is clear that by adding the misclassified example (x_a, y_a) to classifier $f(x)$ with weight β, the classification score of (x_b, y_b) will be reduced by at least $\beta\rho$, which could lead to a significant misclassification of the auxiliary example (x_b, y_b). To avoid such a mistake, we propose to update the weights for both (x_a, y_a) and (x_b, y_b) simultaneously. In the next section, we show the details of the double updating algorithm for online learning, and the analysis for mistake bound.

Our analysis follows closely the previous work on the relationship between online learning and the dual formulation of SVM (Shalev-Shwartz and Singer, 2006, 2007), in which the online learning is interpreted as an efficient updating rule for maximizing the objective function in the dual form of SVM. We denote by Δ_t the improvement of the objective function in dual SVM when adding a misclassified example to the classification function at the t-th trial. According to Theorem 1 in Shalev-Shwartz and Singer (2006), if an online learning algorithm $\mathcal{A}$ is designed to ensure that for all t, Δ_t is bounded from below by a **bounding constant** Δ, then the number of mistakes made by $\mathcal{A}$ when trained over a sequence of trials $(x_1,y_1),\ldots,(x_T,y_T)$, denoted by M, is upper bounded by

$$M \leq \frac{1}{\Delta}\left(\min_{f\in\mathcal{H}_\kappa}\frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa}+C\sum_{i=1}^{T}\ell(y_i f(x_i))\right),$$

where $\ell(y_i f(x_i)) = \max(0, 1-y_i f(x_i))$ is the hinge loss function. According to Shalev-Shwartz and Singer (2006, 2007), the bounding constant $\Delta = 1/2$ when we only update the classifier with the newly misclassified example. In our analysis, we will show that Δ can be significantly improved when updating the weights for both the misclassified example and the auxiliary example.

For the remaining part of this section, we denote by (x_b,y_b) an auxiliary example that satisfies the two conditions specified before. We define

$$k_a = \kappa(x_a,x_a),\ k_b = \kappa(x_b,x_b),\ k_{ab} = \kappa(x_a,x_b),\ w_{ab} = y_a y_b k_{ab}.$$

According to the assumption of auxiliary example, we have $w_{ab} = k_{ab}y_a y_b \leq -\rho$. Finally, we denote by $\widehat{\gamma}_b$ the weight for the auxiliary example (x_b,y_b) that is used in the current classifier $f(x)$, by γ_a and γ_b the updated weights for (x_a,y_a) and (x_b,y_b), respectively, and by d_{γ_b} the difference $\gamma_b - \widehat{\gamma}_b$.

3.3 Double Updating Online Learning for Binary Classification

Recall an auxiliary example (x_b,y_b) should satisfy two conditions (I) $y_b f(x_b) \leq 0$, and (II) $w_{ab} \leq -\rho$. In addition, the example (x_a,y_a) received in the current iteration t is misclassified, that is, $y_a f(x_a) \leq 0$. Following the framework of dual formulation for online learning, the following lemma shows how to compute Δ_t, that is, the improvement in the objective function of dual SVM by adjusting weights for (x_a,y_a) and (x_b,y_b).

Lemma 1 *The maximal improvement in the objective function of dual SVM by adjusting weights for (x_a,y_a) and (x_b,y_b), denoted by Δ_t, is computed by solving the following optimization problem(which is a special case of the optimization problem (28) in Shalev-Shwartz and Singer, 2006):*

$$\Delta_t \quad = \quad \max_{\gamma_a, d_{\gamma_b}}\left\{h(\gamma_a,d_{\gamma_b}) : 0\leq\gamma_a\leq C,\ -\widehat{\gamma}_b\leq d_{\gamma_b}\leq C-\widehat{\gamma}_b\right\} \tag{1}$$

where

$$h(\gamma_a,d_{\gamma_b}) = \gamma_a(1-y_a f(x_a)) + d_{\gamma_b}(1-y_b f(x_b)) - \frac{k_a}{2}\gamma_a^2 - \frac{k_b}{2}d_{\gamma_b}^2 - w_{ab}\gamma_a d_{\gamma_b}.$$

The lemma follows directly the dual formulation of SVM. The theorem below bounds the bounding constant Δ when C is sufficiently large.

Theorem 1 *Assume $C \geq \widehat{\gamma}_b + 1/(1-\rho)$ with $\rho\in[0,1)$ for the selected auxiliary example (x_b,y_b), we have the following bound for the bounding constant Δ:*

$$\Delta \geq \frac{1}{1-\rho}.$$

Proof First, we show $d_{\gamma_b} \geq 0$. This is because for given $\gamma_a \geq 0$, the optimal solution for d_{γ_b}, given by

$$d_{\gamma_b} = \frac{1-y_b f(x_b) - w_{ab}\gamma_a}{k_b},$$

is positive because $y_b f(x_b) \leq 0$ and $w_{ab} \leq -\rho$. Using the fact $k_a, k_b \leq 1$, $\gamma_a, d_{\gamma_b} \geq 0$, $y_a f(x_a) \leq 0$, $y_b f(x_b) \leq 0$, and $w_{a,b} \leq -\rho$, we have

$$h(\gamma_a, d_{\gamma_b}) \geq \gamma_a + d_{\gamma_b} - \frac{1}{2}\gamma_a^2 - \frac{1}{2}d_{\gamma_b}^2 + \rho\gamma_a d_{\gamma_b}.$$

Thus, Δ is bounded as

$$\Delta \geq \max_{\gamma_b \in [0,C], d_{\gamma_b} \in [0, C-\widehat{\gamma}_b]} \gamma_a + d_{\gamma_b} - \frac{1}{2}(\gamma_a^2 + d_{\gamma_b}^2) + \rho\gamma_a d_{\gamma_b}.$$

Under the condition that $C \geq \hat{\gamma}_b + 1/(1-\rho)$, it is easy to verify that the optimal solution for the above problem is $\gamma_a = d_{\gamma_b} = 1/(1-\rho)$, which leads to the result in the theorem. ■

We refer to the case as a **strong double update** when the condition of Theorem 1 is satisfied. We have the following theorem for the general case when we only have $C \geq 1$.

Theorem 2 *Assume $C \geq 1$. We have the following bound for Δ when updating the weight for the misclassified example (x_a, y_a) and the auxiliary example (x_b, y_b):*

$$\Delta \geq \frac{1}{2} + \frac{1}{2}\min\left((1+\rho)^2, (C-\widehat{\gamma})^2\right).$$

Proof By setting $\gamma_a = 1$, we have $h(\gamma_a, d_{\gamma_b})$ computed as

$$h(\gamma_a = 1, d_{\gamma_b}) \geq \frac{1}{2} + (1+\rho)d_{\gamma_b} - \frac{1}{2}d_{\gamma_b}^2.$$

Hence, Δ is lower bounded by

$$\Delta \geq \frac{1}{2} + \max_{d_{\gamma_b} \in [0, C-\widehat{\gamma}]} \left((1+\rho)d_{\gamma_b} - \frac{1}{2}d_{\gamma_b}^2\right) \geq \frac{1}{2} + \frac{1}{2}\min((1+\rho)^2, (C-\widehat{\gamma})^2).$$

■

Although Theorem 1 and 2 show that the double update strategy could significantly improve the bounding constant Δ over $1/2$ and consequentially reduce the mistake bound, it is applicable only when there exists an auxiliary example. Below, we extend the double update strategy to the cases when there is no auxiliary example. Specifically, we relax the condition for performing double update as follows: there exists $(x_b, y_b) \in \mathcal{D}$ that (i) $w_{ab} \leq -\rho$, (ii) $y_b f_{t-1}(x_b) \leq 1$, and (iii) $C \geq \hat{\gamma}_b + \rho$. We refer to these cases as **weak double update**.

Theorem 3 *Assume $w_{ab} \leq -\rho$, $y_b f_{t-1}(x_b) \leq 1$ and $C \geq \hat{\gamma}_b + \rho$, we have the following bound for the bounding constant*

$$\Delta \geq \frac{1+\rho^2}{2}.$$

Proof Following the definitions and assumptions, we have

$$\Delta = \max_{\gamma_a, d_{\gamma_b}} h(\gamma_a, d_{\gamma_b}) \geq h(1, \rho) \geq 1 - \frac{1}{2} + 0 - \frac{\rho^2}{2} + \rho^2 = \frac{1+\rho^2}{2}.$$

Algorithm 1 The Double Updating Online Learning Algorithm (**DUOL**)

```
PROCEDURE
1:  Initialize $S_0 = \emptyset, f_0 = 0$;
2:  for t=1,2,...,T do
3:      Receive a new instance $\mathbf{x}_t$
4:      Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$;
5:      Receive its label $y_t$;
6:      $l_t = max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$
7:      if $l_t > 0$ then
8:          $w_{min} = \infty$;
9:          for $\forall i \in S_{t-1}$ do
10:             if $(f^i_{t-1} \leq 1)$ then
11:                 if $(y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) \leq w_{min})$ then
12:                     $w_{min} = y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
13:                     $(\mathbf{x}_b, y_b) = (\mathbf{x}_i, y_i)$;
14:                 end if
15:             end if
16:         end for
17:         $f^t_{t-1} = y_t f_{t-1}(\mathbf{x}_t)$;
18:         $S_t = S_{t-1} \cup \{t\}$;
19:         if $(w_{min} \leq -\rho)$ then
20:             Compute $\gamma_t$ and $\gamma_b$ by solving
                the optimization (1)
21:             for $\forall i \in S_t$ do
22:                 $f^i_t \leftarrow f^i_{t-1} + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$
                        $+ y_i(\gamma_b - \hat{\gamma}_b) y_b \kappa(\mathbf{x}_i, \mathbf{x}_b)$;
23:             end for
24:             $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$
                    $+ (\gamma_b - \hat{\gamma}_b) y_b \kappa(\mathbf{x}_b, \cdot)$;
25:         else /* no auxiliary example found */
26:             $\gamma_t = min(C, \ell_t / \kappa(x_t, x_t))$;
27:             for $\forall i \in S_t$ do
28:                 $f^i_t \leftarrow f^i_{t-1} + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
29:             end for
30:             $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$;
31:         end if
32:     else
33:         $f_t = f_{t-1}$; $S_t = S_{t-1}$;
34:         for $\forall i \in S_t$ do
35:             $f^i_t \leftarrow f^i_{t-1}$;
36:         end for
37:     end if
38: end for
return $f_T, S_T$
END
```

Figure 1: The Algorithms of Double Updating Online Learning (DUOL).

■

Solving the optimization problem (1) is the key to the double update. The following proposition provides the optimal solution to the problem (1).

Proposition 2 *Denote $\ell_a := 1 - y_a f(x_a)$ and $\ell_b := 1 - y_b f(x_b)$. Assume $\ell_a, \ell_b \geq 0$, $k_a, k_b > 0$ and $w_{ab} \leq 0$, then the solution of optimization problem (1) is as follows:*

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C - \hat{\gamma}_b) & \text{if } (k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a) < 0 \text{ and } (k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0 \text{ and } \frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a}, C - \hat{\gamma}_b) & \text{if } \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} \in [0, C] \text{ and } \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab}\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} > 0 \\ (\frac{k_b \ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2}) & \text{if } (\frac{k_b \ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2}) \in [0, C] \times [-\hat{\gamma}_b, C - \hat{\gamma}_b] \end{cases}.$$

The detailed proof for Proposition 2 can be found in Appendix A. Figure 1 summarizes the proposed Double Updating Online Learning (DUOL) algorithm. In this algorithm, to efficiently find the auxiliary example (x_b, y_b), we introduce a variable f^i_t for each support vector to keep track of its classification score. Parameter ρ is used to trade off between efficiency and efficacy for DUOL: the smaller ρ the more double updates will be performed.

Finally, we give the mistake bound for the DUOL algorithm. We denote by $\mathcal{M}$ the set of indexes that correspond to the trials of misclassification, that is,

$$\mathcal{M} = \{t \,|y_t \neq sign(f_{t-1}(x_t)), \forall t \in [T]\}.$$

In addition, we denote by $\mathcal{M}_d^s(\rho)$ and $\mathcal{M}_d^w(\rho)$ the sets of indexes for the cases of *strong* and *weak* double updating, respectively, that is,

$$\begin{aligned}\mathcal{M}_d^s(\rho) &= \{t \mid \exists \text{ auxiliary example } (x_b, y_b) \text{ s.t. } C \geq \widehat{\gamma}_b + \frac{1}{1-\rho} \text{ for } (x_t, y_t),\ t \in \mathcal{M}\},\\ \mathcal{M}_d^w(\rho) &= \{t \mid \exists\ (x_b, y_b) \text{ s.t. } w_{ab} \leq -\rho,\ y_b f_{t-1}(x_b) \leq 1 \text{ and } C \geq \widehat{\gamma}_b + \rho, t \in \mathcal{M}/\mathcal{M}_d^s(\rho)\}.\end{aligned}$$

Note that in set $\mathcal{M}_d^s(\rho)$, for the convenience of analysis, we only consider the subset of strong updates when the condition $C \geq \widehat{\gamma}_b + 1/(1-\rho)$ is satisfied. Finally, we denote the cardinalities of sets $\mathcal{M}$, $\mathcal{M}_d^s$, and $\mathcal{M}_d^w$ by $M = |\mathcal{M}|$, $M_d^s(\rho) = |\mathcal{M}_d^s(\rho)|$, $M_d^w(\rho) = |\mathcal{M}_d^w(\rho)|$, and $M_s = M - M_d^s(\rho) - M_d^w(\rho)$, respectively.

Theorem 4 *Let $(x_1, y_1), \ldots, (x_T, y_T)$ be a sequence of examples, where $x_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ and $\kappa(x_t, x_t) \leq 1$ for all t, and assume $C \geq 1$. Then for any function f in $\mathcal{H}_\kappa$, the number of prediction mistakes M made by DUOL on this sequence of examples is bounded by:*

$$2\left(\min_{f\in\mathcal{H}_\kappa} \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + C\sum_{i=1}^T \ell(y_i f(x_i))\right) - \frac{\rho^2}{2} M_d^w(\rho) - \frac{1+\rho}{1-\rho} M_d^s(\rho),$$

where $\rho \in [0, 1)$.

Proof According to Theorem 1 and 3, we have

$$\min_{t\in\mathcal{M}_d^s(\rho)} \Delta_t \geq \frac{1}{1-\rho}, \quad \min_{t\in\mathcal{M}_d^w(\rho)} \Delta_t \geq \frac{1+\rho^2}{2}.$$

Moreover, according to Theorem 2, we have $\Delta_t \geq 1/2, \forall t \in \mathcal{M}$. Putting them together, we have

$$\frac{1}{2} M_s + \frac{1+\rho^2}{2} M_d^w(\rho) + \frac{1}{1-\rho} M_d^s(\rho) \leq \left(\min_{f\in\mathcal{H}_\kappa} \frac{1}{2}\|f\|^2_{\mathcal{H}_\kappa} + C\sum_{i=1}^T \ell(y_i f(x_i))\right).$$

We complete the proof using $M = M_s + M_d^w(\rho) + M_d^s(\rho)$. ■

As revealed by the above theorem, the number of mistakes made by the proposed double updating online learning algorithm will be smaller than the online learning algorithm that only performs a single update in each trial. The difference in the mistake bound is essentially due to the double updating, that is, the more the number of double updates, the more advantageous the proposed algorithm will be. Besides, the above bound also indicates that a strong double update is more powerful than a weak double update given that the associated weight of a strong double update $(1+\rho)/(1-\rho)$ is always much larger than that of a weak double update $\rho^2/2$. It is worthwhile pointing out that although according to Theorem 4, it seems that the larger the value of ρ the smaller the mistake bound will be. This however may not be true because $M_d^s(\rho)$ in general decreases as ρ increases. Finally, we note that Theorem 4 bounds the number of mistakes made by the proposed DUOL algorithm for $C \geq 1$. When $C < 1$, the mistake bound for the proposed algorithm follows Theorem 2, 3 and Corollary 2 in Shalev-Shwartz and Singer (2007).

4. Multiclass Double Updating Online Learning

In this section, we extend the proposed double updating online learning algorithm to *multiclass* learning where each instance can be assigned to multiple classes.

4.1 Online Multiclass Learning

Similar to online binary classification, online *multiclass* learning is performed over a sequence of training examples $(x_1, Y_1), \ldots, (x_T, Y_T)$. Unlike binary classification where $y_t \in \{-1, +1\}$, in multi-class learning, each class assignment $Y_t \subseteq \mathcal{Y} = \{1, \ldots, k\}$ could contain multiple class labels, making it a more challenging problem. We use $\hat{Y}_t$ to represent the class set predicted by the online learning algorithm. Before presenting our algorithm, we first review online multiclass learning (Crammer and Singer, 2003; Fink et al., 2006) based on the framework of label ranking (Crammer and Singer, 2005).

4.1.1 LABEL RANKING FOR MULTICLASS LEARNING

Given an instance x, the label ranking approach first computes a score for every class label in $\mathcal{Y}$, and ranks the classes in the descending order of their scores. The predicted class set $\hat{Y}_t$ is formed by the classes with the highest scores. The objective of label ranking is to ensure that the score of class r is significantly larger than that of class s if $r \in Y_t$ is a true class assignment while $s \in \mathcal{Y} \setminus Y_t$ is not. An instance x is classified incorrectly if that above condition is NOT satisfied.

We follow the protocol of *multi-prototype* (Vapnik, 1998; Crammer and Singer, 2001; Crammer et al., 2006) for the design of multiclass multilabel learning algorithm. It learns multiple hypotheses/classifiers, one classifier for each class in $\mathcal{Y}$, leading to a total of k classifiers that are trained for the classification task. Specifically, for trial t, upon receiving an instance x_t, the scores of k classes output by the set of k hypotheses are given by

$$f_{t-1}(x_t) = (f_{t-1,1}(x_t), \cdots, f_{t-1,k}(x_t))^T,$$

where $f_{t-1,i} \in \mathcal{H}_K, i = 1, \ldots, k$. We introduce two variables r_t and s_t that are defined as follows:

$$r_t = \arg\min_{r \in Y_t} f_{t-1,r}(x_t) \quad \text{and} \quad s_t = \arg\max_{s \notin Y_t} f_{t-1,s}(x_t), \tag{2}$$

here, r_t and s_t represent the class of the smallest score among all relevant classes and the class of the largest score among the irrelevant classes, respectively. Using the notation of r_t and s_t, the *margin* with respect to the hypothesis set f_{t-1} at trial t is defined as follows:

$$\Gamma\big(f_{t-1}; (x_t, Y_t)\big) = f_{t-1,r_t}(x_t) - f_{t-1,s_t}(x_t).$$

Based on the notation of classification margin, we define the loss function of hypotheses $f_{t-1}(x)$ for training example (x_t, Y_t) as follows:

$$\ell\big(f_{t-1}; (x_t, Y_t)\big) = \max_{r \in Y_t, s \notin Y_t} \big[1 - (f_{t-1,r}(x_t) - f_{t-1,s}(x_t))\big]_+,$$

where $[x]_+ = \max(0, x)$.

4.1.2 A PERCEPTRON ALGORITHM FOR ONLINE MULTICLASS LEARNING

According to Crammer and Singer (2003), when an example is misclassified at trial t, we update each component of the classifier f_{t-1} as follows:

$$f_{t,i}(x) = f_{t-1,i}(x) + \sigma_{Y_t}(i,t)\gamma_t\kappa(x_t,x),\ \forall i \in \mathcal{Y}, \tag{3}$$

where $\gamma_t \in (0,C]$, and function $\sigma_{Y_t}(i,t)$, which is simplified as $\sigma(i,t)$, is defined below:

$$\sigma(i,t) = \begin{cases} 1 & \text{if } i = r_t \\ -1 & \text{if } i = s_t \\ 0 & \text{otherwise} \end{cases}.$$

Using notation $H(Y_t) = \big(\sigma(1,t),\cdots,\sigma(k,t)\big)^T$, we rewrite Equation (3) as $f_t(x) = f_{t-1}(x) + \gamma_t H(Y_t)\kappa(x_t,x)$, or equivalently

$$f(x) = \sum_{i=1}^{n} \gamma_i H(Y_i)\kappa(x_i,x),$$

where n is the number of support vectors received so far.

4.2 Multiclass DUOL Algorithm

We extend the DUOL algorithm to multiclass learning. We denote by (x_a,Y_a) the misclassified example received at the current trial, that is, $(f(x_a))_{r_a} - (f(x_a))_{s_a} \leq 0$. Similar to DUOL for binary classification, we introduce an auxiliary example (x_b,Y_b) from the existing support vectors that obey the following conditions:

1. $(f(x_b))_{r_b} - (f(x_b))_{s_b} \leq 0$, that is, (x_b,Y_b) is misclassified by current classifier f;
2. $(H(Y_a)\cdot H(Y_b))\kappa(x_a,x_b) \leq -2\rho$ where $\rho \in (0,1)$ is a threshold. This property indicates that example (x_a,Y_a) **conflicts** with example (x_b,Y_b).

Compared to auxiliary example defined for binary classification, we introduce $H(Y_a)\cdot H(Y_b)$ in above when defining two conflicting instances. Given $\kappa(x_a,x_b) \geq 0$, the second condition of auxiliary example implies $H(Y_a)\cdot H(Y_b) \leq 0$, which further indicates that two examples (x_a,Y_a) and (x_b,Y_b) have the opposite prediction, that is, ($r_a = s_b$) or ($s_a = r_b$). This result is revealed by the following proposition.

Proposition 3 *The inequality $H(Y_a)\cdot H(Y_b) < 0$ holds if and only if ($r_a = s_b$) or ($s_a = r_b$).*

The proof of Proposition 3 is given in the appendix.

Similar to the DUOL algorithm for binary classification, our analysis aims to show that by updating weights for both misclassified example and the auxiliary example, we may be able to significantly improve the bounding constant Δ, which is defined as follows:

$$M \times \Delta \leq \Big(\min_{f\in\mathcal{H}_\kappa} F(f) + C\sum_{i=1}^{T} \ell\big(f;(x_i,Y_i)\big)\Big), \tag{4}$$

where $\mathcal{H}_\kappa = \prod_{i=1}^{k}\mathcal{H}_\kappa$ and $F(f) = \sum_{i=1}^{k}\frac{1}{2}\|f_i\|^2_{\mathcal{H}_\kappa}$. To ease our further discussions, we define $k_a = \kappa(x_a,x_a), k_b = \kappa(x_b,x_b), w_{ab} = (H(Y_a)\cdot H(Y_b))\kappa(x_a,x_b)$.

The following proposition shows the optimization problem related to the multiclass double updating online learning algorithm, which forms the basis for deriving the bounding constant Δ.

Proposition 4 *With the double updating, that is, adjusting the weight of some auxiliary support vector (x_b, Y_b) from $\hat{\gamma}_b$ to γ_b (denoted by $d_{\gamma_b} = \gamma_b - \hat{\gamma}_b$) and assigning weight γ_a to the current misclassified example (x_a, Y_a), the improvement in the objective function of dual SVM, denoted by Δ_t, is computed by the following optimization problem:*

$$\begin{aligned} \max_{\gamma_a, d_{\gamma_b}} \quad & \gamma_a\Big(1-\big(f_{t-1,r_a}(x_a)-f_{t-1,s_a}(x_a)\big)\Big)+d_{\gamma_b}\Big(1-\big(f_{t-1,r_b}(x_b)-f_{t-1,s_b}(x_b)\big)\Big) \\ & -k_a\gamma_a^2-k_b d_{\gamma_b}^2-w_{ab}\gamma_a d_{\gamma_b}, \qquad (5) \\ s.t. \quad & 0\le\gamma_a\le C, -\hat{\gamma}_b\le d_{\gamma_b}\le C-\hat{\gamma}_b. \end{aligned}$$

Theorem 5 *Assume $\kappa(x,x)\le 1$ for any x and $C\ge\hat{\gamma}_b+\frac{1}{2(1-\rho)}$ for the selected auxiliary example (x_b, Y_b), we have the following bound for Δ:*

$$\Delta \ge \frac{1}{2(1-\rho)}.$$

We refer to the case as a strong double update when there exists a auxiliary example (x_b, Y_b) s.t. $C\ge\hat{\gamma}_b+\frac{1}{2(1-\rho)}$. Similar to double updating for binary classification, we introduce *weak* double update when there exists (x_b, Y_b) s.t. $w_{ab}\le -2\rho$, $f_{t-1,r_b}(x_b)-f_{t-1,s_b}(x_b)\le 1$, and $C\ge\hat{\gamma}_b+\frac{\rho}{2}$.

Theorem 6 *Assume there exists (x_b, Y_b) s.t. $w_{ab}\le-2\rho$, $f_{t-1,r_b}(x_b)-f_{t-1,s_b}(x_b)\le 1$, $C\ge\hat{\gamma}_b+\frac{\rho}{2}$ and the current instance is misclassified, then we have the following bounding constant*

$$\Delta \ge \frac{1+\rho^2}{4}.$$

The exact solution to the Quadratic Programming (QP) problem in (5) is given by the following proposition.

Proposition 5 *Denote $\ell_a := 1-(f_{t-1,r_a}(x_a)-f_{t-1,s_a}(x_a))$ and $\ell_b := 1-(f_{t-1,r_b}(x_b)-f_{t-1,s_b}(x_b))$. Assume $\ell_a, \ell_b\ge 0$, $k_a, k_b>0$ and $w_{ab}\le 0$, then the solution of optimization (5) is as follows:*

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C-\hat{\gamma}_b) & \textit{if } (2k_aC+w_{ab}(C-\hat{\gamma}_b)-\ell_a)<0 \textit{ and } (2k_b(C-\hat{\gamma}_b)+w_{ab}C-\ell_b)<0 \\ (C, \frac{\ell_b-w_{ab}C}{2k_b}) & \textit{if } \frac{w_{ab}^2C-w_{ab}\ell_b-4k_ak_bC+2k_b\ell_a}{2k_b}>0 \textit{ and } \frac{\ell_b-w_{ab}C}{2k_b}\in[-\hat{\gamma}_b, C-\hat{\gamma}_b] \\ (\frac{\ell_a-w_{ab}(C-\hat{\gamma}_b)}{2k_a}, C-\hat{\gamma}_b) & \textit{if } \frac{\ell_a-w_{ab}(C-\hat{\gamma}_b)}{2k_a}\in[0,C] \textit{ and } \ell_b-2k_b(C-\hat{\gamma}_b)-w_{ab}\frac{\ell_a-w_{ab}(C-\hat{\gamma}_b)}{2k_a}>0 \\ (\frac{2k_b\ell_a-w_{ab}\ell_b}{4k_ak_b-w_{ab}^2}, \frac{2k_a\ell_b-w_{ab}\ell_a}{4k_ak_b-w_{ab}^2}) & \textit{if } (\frac{2k_b\ell_a-w_{ab}\ell_b}{4k_ak_b-w_{ab}^2}, \frac{2k_a\ell_b-w_{ab}\ell_a}{4k_ak_b-w_{ab}^2})\in[0,C]\times[-\hat{\gamma}_b, C-\hat{\gamma}_b] \end{cases}.$$

We skip the proof due to its high similarity to that of Proposition 2. Figure 2 summarizes the steps of the multiclass DUOL (M-DUOL) algorithm. Note that we replace the conditions for auxiliary example with the margin error in order to make more double updates.

A mistake bound for the M-DUOL algorithm, similar to Theorem 4, is given by the following theorem.

Theorem 7 *Let $(x_1, Y_1), \ldots, (x_T, Y_T)$ be a sequence of examples, where $x_t\in\mathbb{R}^n$, $Y_t\subseteq\mathcal{Y}$ and $\kappa(x_i, x_j)\in[0,1]$ for all i, j. And assume $C\ge 1$. Then for any function $f\in\prod_{i=1}^k\mathcal{H}_\kappa$, the number of prediction mistakes M made by M-DUOL on this sequence of examples is bounded by:*

$$4\Big(\min_{f\in\mathcal{H}_\kappa} F(f)+C\sum_{i=1}^T\ell\big(f;(x_i, Y_i)\big)\Big)-\frac{\rho^2}{2}M_d^w(\rho)-\frac{1+\rho}{1-\rho}M_d^s(\rho).$$

Algorithm 2: The Multiclass DUOL Algorithm (**M-DUOL**)

```
PROCEDURE
1:  Initialize $H_0 = \emptyset, S_0 = \emptyset, f_0 = 0$;
2:  for t=1,2,...,T do
3:      Receive a new instance $\mathbf{x}_t$
4:      Predict $W_{t-1} = f_{t-1}(\mathbf{x}_t)$;
5:      Receive its label set $Y_t$
6:      $\ell_t = \left[1 - W_{t-1} \cdot H(Y_t)\right]_+$
7:      if $l_t > 0$ then
8:          $w_{min} = \infty$;
9:          for $\forall i \in S_{t-1}$ do
10:             if $f^i_{t-1} \le 1$ then
11:                 if $(Hk_{ti} < w_{min}$ then
12:                     $w_{min} = Hk_{ti}$;
13:                     $(\mathbf{x}_b, Y_b) = (\mathbf{x}_i, Y_i)$;
14:                 end if
15:             end if
16:         end for
17:         $f^t_{t-1} = W_{t-1} \cdot H(Y_t)$;
18:         $S_t = S_{t-1} \cup \{t\}; H_t = H_{t-1} \cup \{H(Y_t)\}$;
19:         if $(w_{min} \le -2\rho)$ then
20:             Compute $\gamma_t$ and $\gamma_b$ by solving
                 the optimization (5)
21:             for $\forall i \in S_t$ do
22:                 $f^i_t \leftarrow f^i_{t-1} + [\gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \mathbf{x}_i)] \cdot H(Y_i)$
                        $+ [(\gamma_b - \hat{\gamma}_b) * H(Y_b) * \kappa(\mathbf{x}_b, \mathbf{x}_i)] \cdot H(Y_i)$;
23:             end for
24:             $f_t = f_{t-1} + \gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \cdot)$
                    $+ (\gamma_b - \hat{\gamma}_b) * H(Y_b) * \kappa(\mathbf{x}_b, \cdot)$;
25:         else /* no auxiliary example found */
26:             $\gamma_t = min(C, \frac{\ell_t}{2\kappa(x_t, x_t)})$;
27:             for $\forall i \in S_t$ do
28:                 $f^i_t \leftarrow f^i_{t-1} + [\gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \mathbf{x}_i)] \cdot H(Y_i)$;
29:             end for
30:             $f_t = f_{t-1} + \gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \cdot)$;
31:         end if
32:     else
33:         $f_t = f_{t-1}; S_t = S_{t-1}; H_t = H_{t-1}$;
34:         for $\forall i \in S_t$ do
35:             $f^i_t \leftarrow f^i_{t-1}$;
36:         end for
37:     end if
38: end for
return $f_T, S_T, H_T$
END
```

Figure 2: Algorithms of multiclass double-updating online learning (M-DUOL).

5. Experimental Results

In this section, we evaluate the empirical performance of the proposed double updating online learning algorithms for online learning tasks. We first evaluate the performance of DUOL for binary classification, followed by the evaluation of multiclass double updating online learning.

5.1 Testbeds and Experimental Setup for Binary-class Online Learning

We compare our technique with a number of state-of-the-art techniques, including the kernel Perceptron algorithm (Kivinen et al., 2001), the "ROMMA" algorithm and its aggressive version "agg-ROMMA" (Li and Long, 1999), the ALMA$_p(\alpha)$ algorithm (Gentile, 2001), and the Passive-Aggressive algorithms ("PA") (Crammer et al., 2006). For PA, two versions of algorithms (PA-I and PA-II) are implemented as described in Crammer et al. (2006). Note that one may also compare with the online SVM algorithm (Shalev-Shwartz and Singer, 2006), which updates the weights for all support vectors in each trial. However, we do not include this baseline for comparison because it is too computationally intensive to run on some large data sets.

For the proposed DUOL algorithms, we implement three variants based on different solvers to the problem in (1): (i) "DUOL$_{appr}$" that employs an approximate solution to (1), that is, $\gamma_t = \frac{1}{1-\rho}$ and $\gamma_b = \hat{\gamma}_b + \frac{1}{1-\rho}$, (ii)"DUOL" that uses the exact solution to (1) given in Proposition 2, and (iii) "DUOL$_{iter}$" that first updates the weight for the misclassified example and then the weight for auxiliary example, as suggested in Shalev-Shwartz and Singer (2007)

We test all the algorithms on eight benchmark data sets from web machine learning repositories, which are listed in table 1. All of the data sets can be downloaded from LIBSVM website,[1] UCI machine learning repository,[2] and MIT CBCL face data sets.[3]

Data Set	# examples	# features
sonar	208	60
splice	1,000	60
german	1,000	24
mushrooms	8,124	112
dorothea	1,150	100,000
spambase	4,601	57
MITFace	6,977	361
w7a	24,692	300

Table 1: Binary-class data sets used in the experiments.

To make a fair comparison, for all algorithms in comparison, we set $C = 5$ and use the same Gaussian kernel with $\sigma = 8$. For the $\text{ALMA}_p(\alpha)$ algorithm, parameter p and α are set to 2 and 0.9, respectively, based on our experience. For the proposed DUOL algorithm, we fix ρ to be 0 for all cases. All the experiments are repeated 20 times, each with an independent random permutation of the data points. All the results are reported by averaging over the 20 runs. We evaluate the online learning performance by measuring the *mistake rate*, that is, the percentage of examples that are misclassified by the online learning algorithm. We measure the sparsity of the learned classifiers by the number of support vectors. We evaluate computational efficiencies of all the algorithms in terms of their CPU running time (in seconds). All the experiments are run in Matlab over a windows machine of 2.3GHz CPU.

5.2 Performance Evaluation for Binary-Class Online Learning

Table 2 summarizes the performance of all the compared online learning algorithms over the binary data sets. We can draw several observations from the results.

First, among the six baseline algorithms in comparison, we observe that the agg-ROMMA and two PA algorithms (PA-I and PA-II) perform considerably better than the other three algorithms (i.e., Perceptron, ROMMA, and ALMA) in most cases. We also notice that the agg-ROMMA and the two PA algorithms consume considerably larger numbers of support vectors than the other three algorithms. We believe this is because the agg-ROMMA and the two PA algorithms adopt more aggressive strategies than the other three algorithms, resulting in more updates and better classification performance. For the convenience of discussion, we refer to agg-ROMMA and two PA algorithms as *aggressive* algorithms, and the other three online learning algorithms as *non-aggressive* ones.

Second, we observe that among the three variants of double updating online learning, the DUOL approach, which solves the optimization problem exactly, yields the least *mistake rate* with the smallest number of support vectors for most of the cases. Comparing with the baseline algorithms,

1. LIBSVM website is `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.
2. UCI ML repository is at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.
3. MIT CBCL face data sets can be found at `http://cbcl.mit.edu/software-datasets`.

Algorithm	sonar			splice		
	Mistake (%)	Support Vectors (#)	Time (s)	Mistakes (%)	Support Vectors (#)	Time (s)
Perceptron	38.125 ± 3.815	79.30 ± 7.93	**0.004**	27.120 ± 0.975	271.20 ± 9.75	**0.017**
ROMMA	36.587 ± 2.976	**76.10 ± 6.19**	0.006	25.560 ± 0.814	**255.60 ± 8.14**	0.032
agg-ROMMA	34.928 ± 2.860	130.05 ± 7.51	0.009	22.980 ± 0.780	602.90 ± 7.42	0.044
$\text{ALMA}_2(0.9)$	36.370 ± 3.572	86.25 ± 6.43	0.006	26.040 ± 0.965	314.95 ± 9.41	0.032
PA-I	40.986 ± 2.837	154.15 ± 6.95	**0.004**	23.815 ± 1.042	665.60 ± 5.60	0.029
PA-II	40.481 ± 3.023	162.40 ± 6.26	**0.004**	23.515 ± 1.005	689.00 ± 7.85	0.029
DUOL_{iter}	39.495 ± 3.299	149.85 ± 3.42	0.014	23.205 ± 0.932	566.85 ±13.08	0.097
DUOL_{appr}	41.010 ± 2.335	162.25 ± 5.01	0.013	21.945 ± 1.134	721.85 ± 9.10	0.095
DUOL	**34.255 ± 2.811**	137.60 ± 6.99	0.017	**20.875 ± 0.868**	577.15 ±10.81	0.087

Algorithm	german			mushrooms		
	Mistake (%)	Support Vectors (#)	Time (s)	Mistakes (%)	Support Vectors (#)	Time (s)
Perceptron	34.760 ± 0.947	347.60 ± 9.47	**0.019**	2.083 ± 0.278	**169.25 ± 22.58**	**0.148**
ROMMA	34.725 ± 1.009	**347.25 ± 10.09**	0.037	2.429 ± 0.101	197.35 ± 8.24	0.264
agg-ROMMA	32.925 ± 1.184	633.40 ± 14.02	0.049	1.568 ± 0.096	1307.90 ± 39.59	0.576
$\text{ALMA}_2(0.9)$	33.480 ± 0.681	394.75 ± 9.24	0.036	2.538 ± 0.297	304.80 ± 38.02	0.267
PA-I	33.010 ± 1.025	721.10 ± 12.99	0.031	1.661 ± 0.089	1221.55 ± 22.80	0.454
PA-II	32.630 ± 1.016	749.50 ± 11.84	0.032	1.657 ± 0.088	1326.20 ± 22.85	0.483
DUOL_{iter}	35.985 ± 1.077	714.35 ± 12.75	0.125	1.537 ± 0.101	860.05 ± 23.00	0.521
DUOL_{appr}	**30.275 ± 0.937**	716.10 ± 10.44	0.096	1.459 ± 0.101	1291.35 ± 32.03	0.658
DUOL	31.810 ± 1.090	656.30 ± 14.36	0.108	**0.596 ± 0.053**	453.70 ± 19.40	0.341

Algorithm	dorothea			spambase		
	Mistake (%)	Support Vectors (#)	Time (s)	Mistakes (%)	Support Vectors (#)	Time (s)
Perceptron	13.257 ± 0.973	**152.45 ± 11.18**	**0.016**	24.987 ± 0.525	1149.65 ± 24.16	**0.215**
ROMMA	17.461 ± 0.537	200.80 ± 6.18	0.035	23.953 ± 0.510	**1102.10 ± 23.44**	0.275
agg-ROMMA	17.435 ± 0.500	438.30 ± 13.83	0.044	21.242 ± 0.384	2550.70 ± 27.28	0.515
$\text{ALMA}_2(0.9)$	14.478 ± 0.378	210.25 ± 5.68	0.035	23.579 ± 0.411	1550.15 ± 15.65	0.348
PA-I	17.500 ± 0.491	461.30 ± 15.80	0.026	22.112 ± 0.374	2861.50 ± 24.36	0.479
PA-II	17.500 ± 0.491	461.30 ± 15.80	0.027	21.907 ± 0.340	3029.10 ± 24.69	0.504
DUOL_{iter}	21.109 ± 0.796	559.20 ± 19.44	0.080	21.907 ± 0.432	2511.20 ± 34.14	1.215
DUOL_{appr}	17.500 ± 0.491	461.30 ± 15.80	0.054	20.185 ± 0.351	2981.00 ± 26.95	1.091
DUOL	**11.757 ± 0.237**	407.50 ± 12.80	0.080	**19.438 ± 0.282**	2494.95 ± 26.19	1.069

Algorithm	MITFace			w7a		
	Mistake (%)	Support Vectors (#)	Time (s)	Mistakes (%)	Support Vectors (#)	Time (s)
Perceptron	4.665 ± 0.192	325.50 ± 13.37	**0.207**	4.027 ± 0.095	**994.40 ± 23.57**	3.392
ROMMA	4.114 ± 0.155	**287.05 ± 10.84**	0.285	4.158 ± 0.087	1026.75 ± 21.51	1.875
agg-ROMMA	3.137 ± 0.093	1121.15 ± 24.18	0.555	3.500 ± 0.061	2318.65 ± 60.49	3.257
$\text{ALMA}_2(0.9)$	4.467 ± 0.169	400.10 ± 10.53	0.297	3.518 ± 0.071	1031.05 ± 15.33	**1.314**
PA-I	3.190 ± 0.128	1155.45 ± 14.53	0.439	3.701 ± 0.057	2839.60 ± 41.57	2.691
PA-II	3.108 ± 0.112	1222.05 ± 13.73	0.463	3.571 ± 0.053	3391.50 ± 51.94	3.311
DUOL_{iter}	2.551 ± 0.128	963.45 ± 23.80	0.572	4.456 ± 0.073	3048.85 ± 54.53	4.566
DUOL_{appr}	2.687 ± 0.140	1262.50 ± 20.68	0.656	3.116 ± 0.104	2908.95 ± 28.65	3.679
DUOL	**2.151 ± 0.106**	697.95 ± 13.17	0.445	**2.914 ± 0.045**	2402.55 ± 39.88	6.470

Table 2: Evaluation of online learning algorithms on the binary-class data sets.

we observe that DUOL achieves significantly smaller *mistake rates* than the other single-updating algorithms in all cases. This shows that the proposed double updating approach is effective in improving the performance of online prediction. By examining the number of support vectors, we observed that DUOL results in sparser classifiers than the three aggressive online learning algorithms, and denser classifiers than the three non-aggressive algorithms.

Third, according to the results of running time, we observe that DUOL is overall efficient as compared with the state-of-the-art online learning algorithms. Among all the algorithms in comparison, Perceptron, due to its simplicity nature, is clearly the most efficient algorithm. Since DUOL requires double updates, it is less efficient than PA, ROMMA and ALMA algorithms, but is comparable to the agg-ROMMA algorithm. Note that the comparisons of running time costs are slightly different compared with the results in our previous conference paper (Zhao et al., 2009) because we did some improvements of efficiency for the implementations of some existing algorithms in this journal article.

5.3 Evaluation of Different Auxiliary Example Selection Strategies and the Sensitivity to Parameter C for DUOL

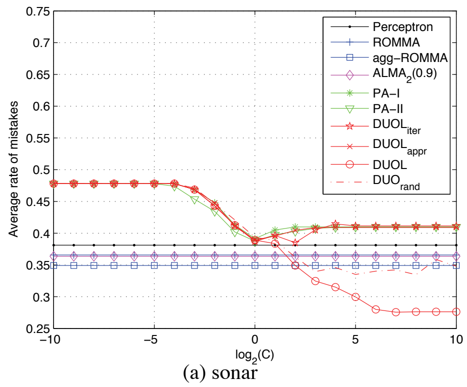
As the performance of DUOL quite relies on the choice of auxiliary examples, in this section, we evaluate different auxiliary example selection strategies. Specifically, we compare the proposed strategy to a random selection approach, referred to as "DUOL$_{rand}$", which randomly chooses an auxiliary example from the existing support vectors. The exact solution to the problem in (1), given by Proposition 2, is used for updating the weights of both examples. We set $\rho = 0$ and $\sigma = 8$ for all the data sets, same as the previous experiments.

Figure 3 compares the online prediction performance between DUOL and DUOL$_{rand}$ as well as the other competing algorithms with varied C values across eight different data sets. Several observations can be drawn from the results.

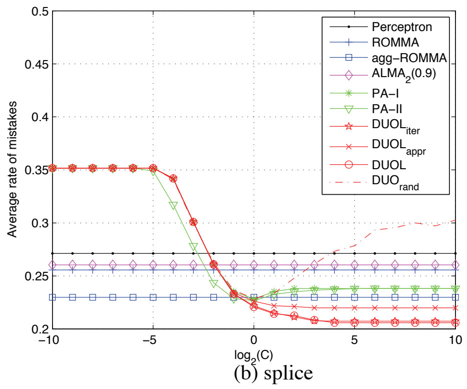
First, it is clear to see that the proposed strategy for selecting auxiliary examples is more effective than the random selection strategy for most cases. Second, among all the compared algorithms, we observe that DUOL always achieves the best performance when C is sufficiently large (e.g., $C > 10$), except for data sets "german" and "w7a" where a smaller C value tends to produce a better result. This observation is consistent to our previous theoretical result, which indicates setting a large C value usually implies more strong updates and consequently a better mistake bound. Third, we observe that the proposed DUOL algorithm is significantly more accurate than the other two variants of double updating online learning algorithms (DUOL$_{iter}$ and DUOL$_{appr}$) for varied C values, as we expected. We observe that DUOL$_{iter}$, the iterative updating approach, performs unstably, which might be due to local optimum suffered from its heuristic update. This observation validates the importance of performing the optimal double updates by the proposed DUOL algorithm.

5.4 Empirical Evaluation of Mistake Bounds

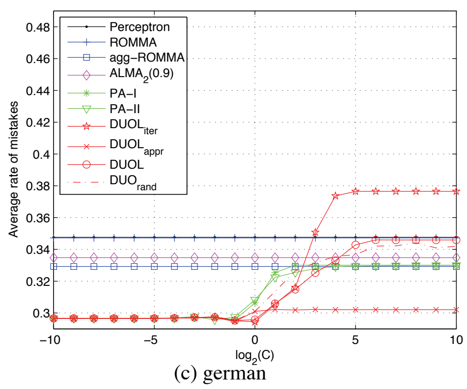
To examine how the double updating strategy affects the mistake bound, we empirically compare M, the total number of mistakes made by the DUOL algorithm, $M_d^w(\rho)$, the number of mistake cases where the weak double updates are applied, and $M_d^s(\rho)$, the number of mistake cases where the strong double updates are applied. Figure 4 shows the comparison between M, $M_d^w(\rho)$, and $M_d^s(\rho)$ by varying ρ from 0 to 1.



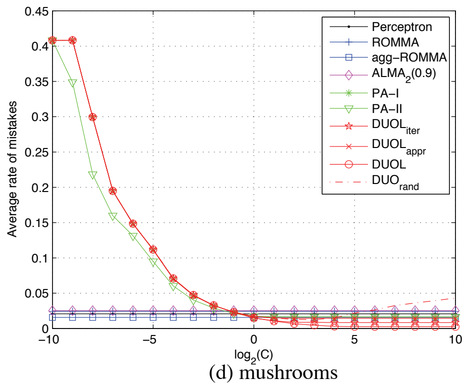
(a) sonar



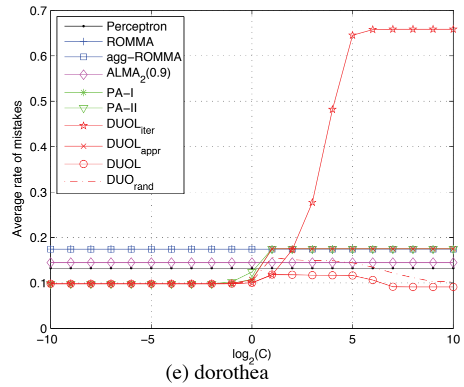
(b) splice



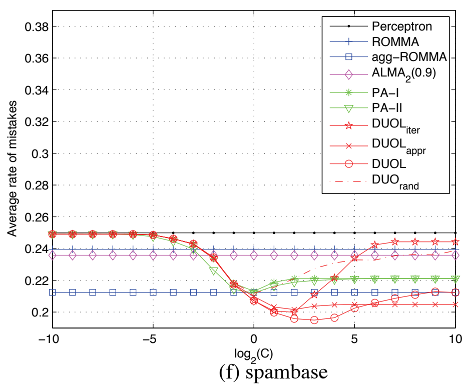
(c) german



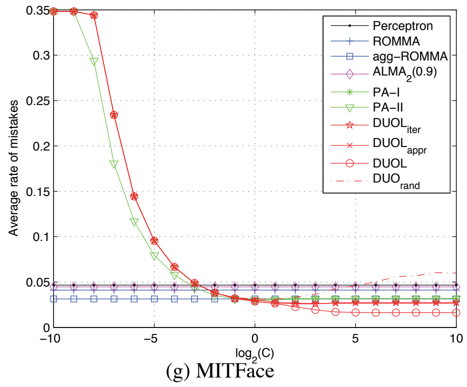
(d) mushrooms



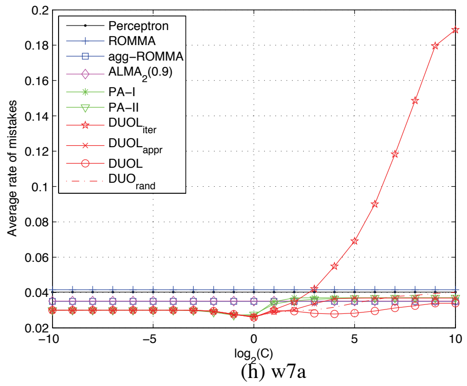
(e) dorothea



(f) spambase



(g) MITFace



(h) w7a

Figure 3: Comparison between DUOL and DUOL$_{rand}$ with varied C values.

First, we observe that double updates are frequently applied when ρ is small. This is because it is easier to find an auxiliary example for double updating when ρ is small. Further, we find that setting ρ close to 0 by default often leads to the best or close to the best results. Second, we observe that the number of weak updates is significantly larger than that of strong updates. This is because the condition of conducting a strong double update is significantly more difficult to be satisfied that that for a weak double update. Third, we observe that both $M_d^w(\rho)$ and $M_d^s(\rho)$ monotonically decrease when increasing the value of ρ. In the extreme case, when ρ is close to 1, their value often drops to zero, indicating that no double update was applied. In the meantime, we find that the total number of mistakes often reaches the maximum, as ρ approaches 1. These results again validate the importance and effectiveness of the proposed double updating algorithm.

5.5 Testbeds and Experimental Setup for Multiclass Online Learning

Table 3 shows the multiclass data sets from Web machine learning repository used in our experiments. We compare the proposed M-DUOL algorithm with six state-of-the-art online learning algorithms. The first three algorithms are variants of Perceptron-based on methods studied in Crammer and Singer (2003). They are: (i) "Max", the perceptron method based on the *max-score* multiclass update, (ii) "Uniform", the perceptron method based on the *Uniform* multiclass update, and (iii) "Prop", the perceptron method based on the *proportion* multiclass update. We also compare the proposed algorithm with the other three state-of-the-art online multi-class learning algorithms, including the MIRA algorithm proposed by Crammer and Singer (2003), and the Passive-Aggressive (PA) algorithms, "PA-I" and "PA-II" proposed by Crammer et al. (2006). Similar to the experiments of binary classification, we implement three variants of the proposed M-DUOL algorithm based on different solvers to the problem in (5), that is, "M-DUOL$_{appr}$", "M-DUOL", and "M-DUOL$_{iter}$". For all experiments, we use the Gaussian kernel with $\sigma = 8$ and set $C = 10$. The threshold ρ in the proposed algorithms is set to 0 for all experiments. All the experiments were repeated 20 time and the final results are averaged over 20 runs.

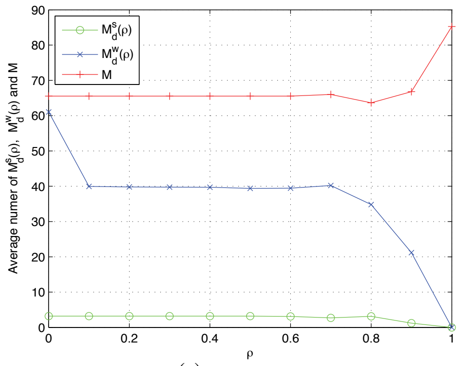
data set	# training examples	# classes	# features
vehicle	846	4	18
dna	2,000	3	180
segment	2,310	7	19
satimage	4,435	6	36
usps	7,291	10	256
mnist	10,000	10	780
letter	15,000	26	16
protein	17,766	3	357

Table 3: Multiclass data sets used in the experiments.

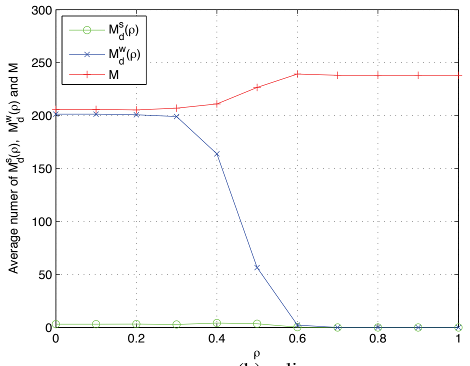
5.6 Performance Evaluation for Multi-class Online Learning

Table 4 summarizes the empirical performance for multi-class online learning. Several observations can be drawn from the experimental results.

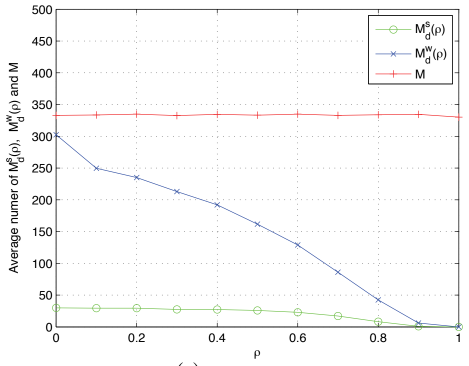
First, by comparing all the baseline algorithms, we find that the two PA algorithms yield considerably lower mistake rates than the other single-updating online learning algorithms. On the other hand, the classifiers learned by the three Perceptron-based algorithms (Max, Uniform, and



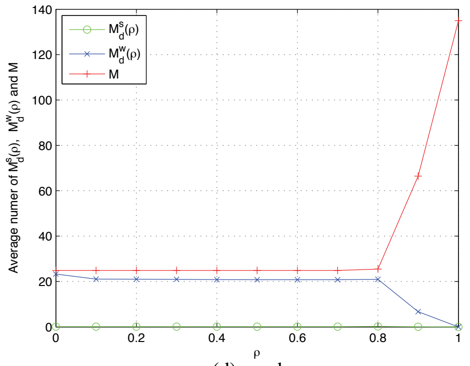
(a) sonar



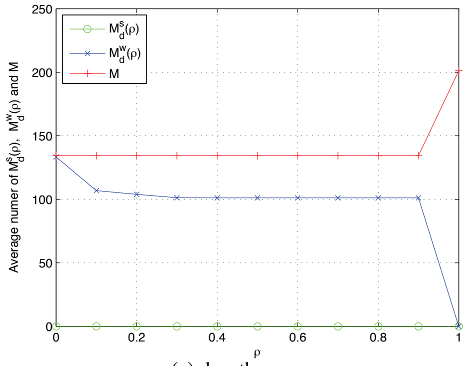
(b) splice



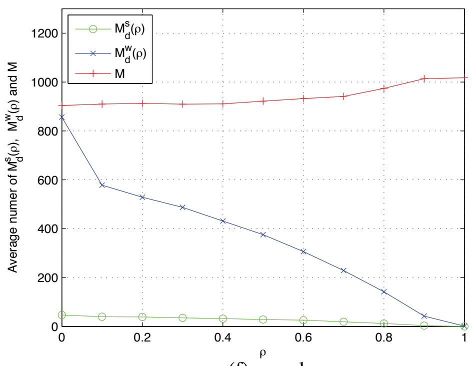
(c) german



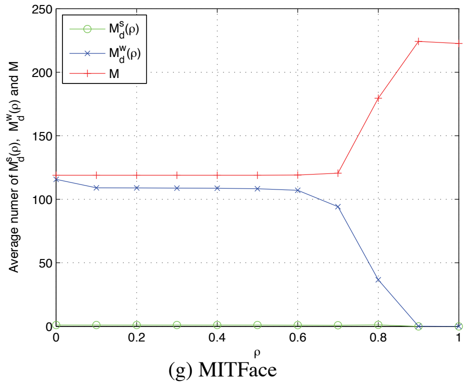
(d) mushrooms



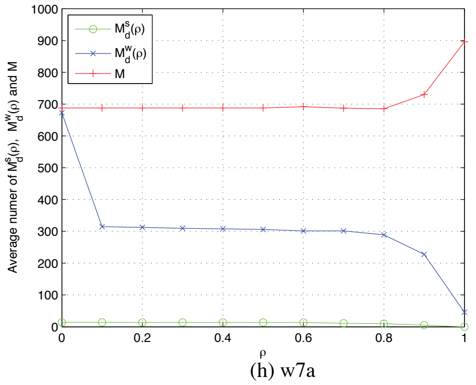
(e) dorothea



(f) spambase



(g) MITFace



(h) w7a

Figure 4: Empirical comparison of M, $M_d^w(\rho)$ and $M_d^s(\rho)$ w.r.t. varied $\rho \in [0,1]$ values.

Algorithm	vehicle				dna		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Max	64.882 ± 1.643	548.90 ± 13.90	**0.079**		20.460 ± 0.770	409.20 ± 15.41	**0.192**
Uniform	65.934 ± 1.554	557.80 ± 13.15	0.109		19.875 ± 0.427	**397.50 ± 8.54**	0.264
Prop	66.678 ± 1.757	564.10 ± 14.86	0.116		20.268 ± 0.555	405.35 ± 11.10	0.267
MIRA	62.252 ± 2.114	**526.65 ± 17.89**	1.821		26.920 ± 0.880	538.40 ± 17.61	5.304
PA-I	67.086 ± 1.479	781.70 ± 12.42	0.091		15.503 ± 0.474	1224.35 ± 13.48	0.326
PA-II	66.909 ± 1.475	789.30 ± 10.73	0.089		15.398 ± 0.467	1237.50 ± 13.12	0.325
M-DUOL$_{iter}$	70.674 ± 1.194	758.05 ± 8.65	0.162		11.668 ± 0.599	1086.00 ± 16.39	0.502
M-DUOL$_{appr}$	69.634 ± 1.463	828.05 ± 4.48	0.158		14.105 ± 0.611	1281.75 ± 14.44	0.495
M-DUOL	**51.950 ± 1.948**	719.25 ± 10.95	0.172		**10.340 ± 0.513**	869.80 ± 12.61	0.438

Algorithm	segment				satimage		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Max	41.342 ± 1.013	955.00 ± 23.40	**0.414**		29.628 ± 0.561	1314.00 ± 24.89	**0.826**
Uniform	41.468 ± 0.550	957.90 ± 12.71	0.566		28.440 ± 0.398	1261.30 ± 17.64	1.071
Prop	41.589 ± 0.714	960.70 ± 16.48	0.565		28.878 ± 0.467	1280.75 ± 20.72	1.087
MIRA	35.784 ± 3.770	**826.55 ± 87.08**	9.193		27.536 ± 2.228	**1221.20 ± 98.80**	15.229
PA-I	39.775 ± 0.665	1852.75 ± 19.90	0.573		27.377 ± 0.361	2676.40 ± 24.88	1.296
PA-II	39.842 ± 0.655	1870.70 ± 18.97	0.577		27.258 ± 0.429	2709.50 ± 23.77	1.307
M-DUOL$_{iter}$	41.416 ± 1.084	1787.90 ± 31.00	0.903		33.894 ± 0.567	2787.45 ± 43.18	2.024
M-DUOL$_{appr}$	39.314 ± 0.791	1923.60 ± 14.31	0.871		26.222 ± 0.464	3052.50 ± 31.39	2.024
M-DUOL	**20.580 ± 0.705**	1265.15 ± 28.39	0.693		**22.524 ± 0.482**	2066.85 ± 32.99	1.505

Algorithm	usps				mnist		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Max	10.025 ± 0.195	730.90 ± 14.21	**1.459**		15.318 ± 0.168	1531.80 ± 16.80	**2.744**
Uniform	9.445 ± 0.150	**688.60 ± 10.91**	1.858		14.603 ± 0.201	**1460.25 ± 20.15**	3.631
Prop	9.614 ± 0.176	700.95 ± 12.86	1.868		14.763 ± 0.228	1476.30 ± 22.78	3.635
MIRA	11.572 ± 0.403	843.75 ± 29.39	44.663		18.037 ± 0.539	1803.70 ± 53.93	67.168
PA-I	6.641 ± 0.158	2528.45 ± 23.48	2.669		11.026 ± 0.208	4773.70 ± 32.84	5.771
PA-II	6.568 ± 0.116	2561.95 ± 27.94	2.606		10.959 ± 0.238	4830.40 ± 27.06	5.824
M-DUOL$_{iter}$	5.743 ± 0.158	2284.15 ± 40.06	3.160		8.947 ± 0.182	4398.95 ± 46.46	9.031
M-DUOL$_{appr}$	6.002 ± 0.132	2725.40 ± 23.55	3.541		9.640 ± 0.164	5163.05 ± 37.34	10.386
M-DUOL	**5.162 ± 0.149**	1759.30 ± 23.44	2.408		**8.282 ± 0.183**	3557.15 ± 25.17	7.050

Algorithm	letter				protein		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Max	71.562 ± 0.538	10734.35 ± 80.63	**18.749**		47.657 ± 0.221	8466.75 ± 39.21	**12.842**
Uniform	71.973 ± 0.280	10795.90 ± 41.99	47.031		46.828 ± 0.272	**8319.45 ± 48.36**	14.342
Prop	72.033 ± 0.273	10804.95 ± 40.89	43.683		47.260 ± 0.260	8396.15 ± 46.13	14.620
MIRA	67.709 ± 1.196	**10156.35 ±179.54**	467.019		47.905 ± 0.922	8510.80 ±163.74	42.174
PA-I	72.283 ± 0.338	14708.55 ± 15.27	24.848		47.657 ± 0.230	14153.25 ± 49.06	23.409
PA-II	72.339 ± 0.380	14735.55 ± 15.86	24.131		47.550 ± 0.285	14285.85 ± 44.94	23.602
M-DUOL$_{iter}$	73.066 ± 0.326	14614.65 ± 22.26	210.684		50.070 ± 0.392	14191.85 ± 64.80	55.622
M-DUOL$_{appr}$	69.992 ± 0.331	14892.70 ± 11.77	215.587		51.459 ± 0.582	16000.55 ± 72.07	63.065
M-DUOL	**54.068 ± 0.351**	13140.40 ± 37.33	186.452		**46.281 ± 0.418**	12550.10 ± 87.27	43.774

Table 4: Evaluation of multiclass online learning algorithms on the multiclass data sets.

Prop) and MIRA are considerably sparser than those learned by the two PA algorithms. We believe that this can be attributed to the aggressive updating strategies used by the PA algorithms. Second, among the three variants of double updating for multi-label learning, it is not surprising to observe that M-DUOL yields the lowest mistake rates for all data sets. Further, among all the algorithms, we observe that the M-DUOL algorithm makes the least number of mistakes for all data sets, and significantly outperforms all the baseline algorithms.

Second, by examining the sparsity of classifiers learned by the proposed algorithms, we observe that the number of support vectors identified by M-DUOL is usually smaller than that of the PA algorithms (except for data set "vehicle"), but is significantly larger than those of the four non-aggressive algorithms (i.e., Max, Uniform, Prop, and MIRA).

Finally, comparing the running time cost, we observe that the Max algorithm is the most efficient one, while MIRA is the least efficient approach for all the data sets. Despite the additional time needed for double updates, overall we found that the running time of the proposed M-DUOL algorithm is comparable to those of the two PA algorithms (except for the "letter" data set where the time costs of the M-DUOL algorithms are considerably greater than those of the PA algorithms).

6. Discussions and Future Directions

Although encouraging results have been achieved by the proposed novel DUOL algorithms, we should address the limitations of our current work and discuss some research directions for future improvements. First of all, the proposed DUOL algorithm is based on the Passive Aggressive online learning algorithms (Crammer et al., 2006). For the future work, it is possible to extend other single update online learning methods, such as EG (Kivinen and Warmuth, 1995), for double updating. Second, the approach for choosing an auxiliary example from existing support vectors may be further improved by exploring the heuristics for measuring the informativeness of an example. Finally, we plan to extend the proposed double updating framework for budget online learning to make sparse classifiers.

7. Conclusions

This paper presented a novel "double updating" approach to online learning named as "DUOL", which not only updates the weight of the misclassified example, but also adjusts the weight of one existing support vector that the most seriously conflicts with the new support vector. We show that the mistake bound for an online classification task can be significantly reduced by the proposed DUOL algorithms. We have conducted an extensive set of experiments by comparing with a number of algorithms for both binary and multiclass online classifications. Promising empirical results showed that the proposed double updating online learning algorithms consistently outperform the single-update online learning algorithms.

Acknowledgments

The work described in this paper was supported in part by Singapore MOE ARC Tier-2 Grant (T208B2203), Singapore MOE ARC Tier-1 Grant (RG67/07), National Science Foundation (IIS-0643494), and US Office of Naval Research (ONR N00014-09-1-0663).

Appendix A. The Proof for Proposition 2

Proof The optimization (1) can be rewritten to the following equivalent optimization:

$$\begin{aligned}
\min_{\gamma_a, d_{\gamma_b}} \quad & \frac{k_a}{2}\gamma_a^2 + \frac{k_b}{2}d_{\gamma_b}^2 + w_{ab}\gamma_a d_{\gamma_b} - \ell_a\gamma_a - \ell_b d_{\gamma_b}, \\
\text{s.t.} \quad & \gamma_a - C \le 0, && (6)\\
& -\gamma_a \le 0, && (7)\\
& d_{\gamma_b} - C + \hat{\gamma}_b \le 0, && (8)\\
& -d_{\gamma_b} - \hat{\gamma}_b \le 0, && (9)
\end{aligned}$$

where $k_a, k_b > 0$, $w_{ab} \le 0$, $\ell_a = 1 - y_a f(x_a) \ge 0$, $\ell_b = 1 - y_b f(x_b) \ge 0$ and $\hat{\gamma}_b > 0$. With λ_1, λ_2, λ_3 and λ_4 as Lagrange multipliers, the KKT conditions for this problem consist of the constraints (6)-(9), the nonnegativity constraints $\lambda_i \ge 0$, $\forall i$, the complementary slackness conditions

$$\lambda_1(\gamma_a - C) = 0,\ \lambda_2(-\gamma_a) = 0,\ \lambda_3(d_{\gamma_b} - C + \hat{\gamma}_b) = 0,\ \lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$$

and zero gradient conditions:

$$k_a\gamma_a + w_{ab}d_{\gamma_b} - \ell_a + \lambda_1 - \lambda_2 = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab}\gamma_a - \ell_b + \lambda_3 - \lambda_4 = 0.$$

We will discuss every possible condition to compute the closed-form solution. Firstly, we will discuss the case $\lambda_1 \neq 0$:

A.1 Case 1. If $\lambda_1 \neq 0$

Since $\lambda_1(\gamma_a - C) = 0$, we have $\gamma_a = C$; further, because $\lambda_2(-\gamma_a) = 0$, we have $\lambda_2 = 0$. Under the condition $\lambda_1 \neq 0$, we will discuss $\lambda_3 \neq 0$ and $\lambda_3 = 0$ separately as follows:

A.1.1 SUB-CASE 1.1. IF $\lambda_3 \neq 0$

Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, we have $d_{\gamma_b} = C - \hat{\gamma}_b$, as a result $\lambda_4(-C) = 0$, so $\lambda_4 = 0$. Plugging the results $\gamma_a = C$, $\lambda_2 = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and $\lambda_4 = 0$ into the zero gradient condition, we have

$$k_aC + w_{ab}(C - \hat{\gamma}_b) - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b + \lambda_3 = 0.$$

Thus, we have

$$\lambda_1 = -[k_aC + w_{ab}(C - \hat{\gamma}_b) - \ell_a] \quad \text{and} \quad \lambda_3 = -[k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b].$$

As a result, if

$$-(k_aC + w_{ab}(C - \hat{\gamma}_b) - \ell_a) > 0 \quad \text{and} \quad -(k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) > 0,$$

then KKT conditions are satisfied, $(\gamma_a, d_{\gamma_b}) = (C, C - \hat{\gamma}_b)$ is the unique solution.

A.1.2 SUB-CASE 1.2. IF $\lambda_3 = 0$

When $\lambda_3 = 0$, we only conclude $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$.
Under the conditions $\lambda_1 \neq 0$ and $\lambda_3 = 0$, we will discuss the two cases $\lambda_4 \neq 0$ and $\lambda_4 = 0$, respectively as follows.
Sub-case 1.2.1. If $\lambda_4 \neq 0$. Since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, we have $d_{\gamma_b} = -\hat{\gamma}_b$. Plugging the results $\lambda_2 = 0$, $\gamma_a = C$, $\lambda_3 = 0$ and $d_{\gamma_b} = -\hat{\gamma}_b$ in to the zero gradient conditions:

$$k_aC + w_{ab}(-\hat{\gamma}_b) - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) + w_{ab}C - \ell_b - \lambda_4 = 0.$$

But since $k_b(-\hat{\gamma}_b) < 0$ $w_{ab}C \leq 0$ and $\ell_b, \lambda_4 \geq 0$, $k_b(-\hat{\gamma}_b) + w_{ab}C - \ell_b - \lambda_4 < 0$, which contradicts the equation above.
Sub-case 1.2.2. If $\lambda_4 = 0$. Plugging the conditions $\gamma_a = C$, $\lambda_2 = 0$, $\lambda_3 = 0$ and $\lambda_4 = 0$ into the zero gradient equations:

$$k_aC + w_{ab}d_{\gamma_b} - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab}C - \ell_b = 0.$$

Solving the above equations leads to the following:

$$\lambda_1 = \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} \quad \text{and} \quad d_{\gamma_b} = \frac{\ell_b - w_{ab}C}{k_b}.$$

If $\frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0$ and $\frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$, then the KKT conditions are all satisfied; as a result, $(\gamma_a, d_{\gamma_b}) = (C, \frac{\ell_b - w_{ab}C}{k_b})$ is the unique optimal solution.

Next we will discuss the situation with the condition $\lambda_1 = 0$.

A.2 Case 2. If $\lambda_1 = 0$

Under the condition $\lambda_1 = 0$, we only conclude $\gamma_a \in [0, C]$. We will discuss the cases $\lambda_2 \neq 0$ and $\lambda_2 = 0$ under the condition $\lambda_1 = 0$, respectively.

A.2.1 SUB-CASE 2.1. IF $\lambda_2 \neq 0$

Since $\lambda_2(-\gamma_a) = 0$, we conclude $\gamma_a = 0$. Under the conditions $\lambda_1 = 0$ and $\lambda_2 \neq 0$, we will discuss the cases $\lambda_3 \neq 0$ and $\lambda_3 = 0$:
Sub-case 2.1.1. If $\lambda_3 \neq 0$. Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, plugging the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and $\lambda_4 = 0$ into the zero gradient conditions:

$$w_{ab}(C - \hat{\gamma}_b) - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) - \ell_b + \lambda_3 = 0.$$

Since $w_{ab} \leq 0, C - \hat{\gamma}_b \geq 0$ and $\ell_a \geq 0$, we conclude

$$\lambda_2 = w_{ab}(C - \hat{\gamma}_b) - \ell_a \leq 0.$$

But $\lambda_2 \geq 0$ and $\lambda_2 \neq 0$, conclude $\lambda_2 > 0$, which contradicts the inequality above.
Sub-case 2.1.2. If $\lambda_3 = 0$. Under these known conditions, we only know $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$. Below, we will discuss the cases $\lambda_4 \neq 0$ and $\lambda_4 = 0$, under the conditions $\lambda_1 = 0, \lambda_2 \neq 0$ and $\lambda_3 = 0$.

- If $\lambda_4 \neq 0$, since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$. From the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $\lambda_3 = 0$ and $d_{\gamma_b} = -\hat{\gamma}_b$ and the zero gradient conditions, we have

  $$w_{ab}(-\hat{\gamma}_b) - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) - \ell_b - \lambda_4 = 0.$$

  Since $k_b, \hat{\gamma}_b > 0$ and $\ell_b \geq 0$, we conclude

  $$\lambda_4 = k_b(-\hat{\gamma}_b) - \ell_b < 0.$$

  But the equation above contradicts $\lambda_4 > 0$.

- Else if $\lambda_4 = 0$, from the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $\lambda_3 = 0$ and $\lambda_4 = 0$ and the zero gradient conditions, we have

  $$w_{ab} d_{\gamma_b} - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b d_{\gamma_b} - \ell_b = 0.$$

  Since $w_{ab} \leq 0$, $\ell_b, \ell_a \geq 0$ and $k_b > 0$,

  $$\lambda_2 = w_{ab}\frac{\ell_b}{k_b} - \ell_a \leq 0,$$

  which contradicts $\lambda_2 > 0$ (Since $\lambda_2 \neq 0$).

A.2.2 SUB-CASE 2.2. IF $\lambda_2 = 0$

Under the conditions $\lambda_1 = \lambda_2 = 0$, we only know $\gamma_a \in [0, C]$. Below, we will discuss the two cases $\lambda_3 \neq 0$ and $\lambda_3 = 0$, under the conditions $\lambda_1 = \lambda_2 = 0$.

Sub-case 2.2.1. If $\lambda_3 \neq 0$. Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$, as a result $\lambda_4(-C) = 0$, so $\lambda_4 = 0$. From the conditions $\lambda_1 = \lambda_2 = \lambda_4 = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and zero gradient conditions:

$$k_a\gamma_a + w_{ab}(C - \hat{\gamma}_b) - \ell_a = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) + w_{ab}\gamma_a - \ell_b + \lambda_3 = 0.$$

As a result, if

$$\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} \in [0, C] \quad \text{and} \quad \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab}\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} > 0,$$

the unique optimal solution is $(\gamma_a, d_{\gamma_b}) = (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a}, C - \hat{\gamma}_b)$.

Sub-case 2.2.2. If $\lambda_3 = 0$. According to $\lambda_1 = \lambda_2 = \lambda_3 = 0$, we only conclude $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$.

- If $\lambda_4 \neq 0$, since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$. From $\lambda_1 = \lambda_2 = \lambda_3 = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$ and zero gradient conditions:

  $$k_a\gamma_a + w_{ab}(-\hat{\gamma}_b) - \ell_a = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) + w_{ab}\gamma_a - \ell_b - \lambda_4 = 0,$$

  since $\lambda_4 = k_b(-\hat{\gamma}_b) + w_{ab}\gamma_a - \ell_b < 0$ which contradicts with the condition $\lambda_4 > 0$.

- If $\lambda_4 = 0$, from $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$ and zero gradient conditions:

  $$k_a\gamma_a + w_{ab}d_{\gamma_b} - \ell_a = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab}\gamma_a - \ell_b = 0.$$

  As a result, if γ_a and d_{γ_b} satisfy the following:

  $$\gamma_a = \frac{k_b\ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2} \in [0, C] \quad \text{and} \quad d_{\gamma_b} = \frac{k_a\ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b],$$

  then $(\gamma_a, d_{\gamma_b}) = (\frac{k_b\ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a\ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2})$ is the unique optimal solution.

Summary: The final closed-form solution to the optimization is summarized as:

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C-\hat{\gamma}_b) & \text{if } (k_a C + w_{ab}(C-\hat{\gamma}_b) - \ell_a) < 0 \text{ and } (k_b(C-\hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0 \text{ and } \frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C-\hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{k_a}, C-\hat{\gamma}_b) & \text{if } \frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{k_a} \in [0,C] \text{ and } \ell_b - k_b(C-\hat{\gamma}_b) - w_{ab}\frac{\ell_a - w_{ab}(C-\hat{\gamma}_b)}{k_a} > 0 \\ (\frac{k_b\ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a\ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2}) & \text{if } (\frac{k_b\ell_a - w_{ab}\ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a\ell_b - w_{ab}\ell_a}{k_a k_b - w_{ab}^2}) \in [0,C] \times [-\hat{\gamma}_b, C-\hat{\gamma}_b] \end{cases}.$$

■

Appendix B. The Proof for Proposition 3

Proof First of all, the product $H(Y_a) \cdot H(Y_b)$ can be simplified as:

$$H(Y_a) \cdot H(Y_b) = \sum_{i=1}^{k} \sigma(i,a)\sigma(i,b) = \sigma(r_a,a)\sigma(r_a,b) + \sigma(s_a,a)\sigma(s_a,b) = \sigma(r_a,b) - \sigma(s_a,b).$$

We can check the value of $\sigma(r_a,b) - \sigma(s_a,b)$ by examining all possible cases as follows:

1 If $r_a = r_b$ that implies that x_a and x_b have the same relevant labels, then we should have $H(Y_a) \cdot H(Y_b) = 1 - \sigma(s_a,b) \geq 1$ (either 1 or 2);

2 If $r_a \neq r_b$, then:

2.1 If $r_a = s_b$, then $H_{Y_a} \cdot H_{Y_b} = \sigma(s_b,b) - \sigma(s_a,b) = -1 - \sigma(s_a,b) \leq -1$;

2.2 If $r_a \neq s_b$, then $H_{Y_a} \cdot H_{Y_b} = \sigma(r_a,b) - \sigma(s_a,b) = 0 - \sigma(s_a,b)$:

2.2.1 If $s_a = s_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(s_b,b) = 1$;

2.2.2 If $s_a = r_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(r_b,b) = -1$;

2.2.3 If $s_a \neq s_b$ and $s_a \neq r_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(s_a,b) = 0$.

We thus have the fact that $H(Y_a) \cdot H(Y_b) < 0$ holds if and only if $(r_a = s_b)$ or $(s_a = r_b)$. ■

Appendix C. The Proof of Proposition 4

In this appendix, we will derive the dual ascent by the multiclass double updating approach. Our approach to the proofs is mainly inspired by the study in Shalev-Shwartz (2007), but our problem is different from their study.

For the convenience of our presentation, we introduce the following notation for our derivation. We denote the loss function for a training example (x,Y) as follows:

$$g(f) = \ell\big(f;(x,Y)\big) = \max_{r \in Y, s \notin Y} \Big[1 - \big(f_r(x) - f_s(x)\big)\Big]_+.$$

We order all the classes r in the assigned set Y as $r_1, \cdots, r_{\|Y\|}$, and the class s in the unassigned set $\mathcal{Y} \setminus Y$ as $s_1, \cdots, s_{\|[k]/Y\|}$. We slightly abuse our notations by simplifying $\langle f,g \rangle_{\mathcal{H}_K}$ as $\langle f,g \rangle$ and $\|f\|_{\mathcal{H}_K}$ as $\|f\|$ when there is no ambiguity about the space for computing dot product and norm.

We first give a lemma that shows the Fenchel conjugate of the above loss function g.

Lemma 2 *Let $\mathcal{Y} = [k]$ be the possible labels set. $Y \subseteq \mathcal{Y}$ is relevant labels set for $x \in R^n$. $f = (f_1, \cdots, f_k)^T$, where $\forall i \in [k]$, $f_i \in \mathcal{H}_\kappa$. And the loss function is defined as follows:*

$$g(f) = \max_{r \in Y, s \notin Y} \Big[1 - \big(f_r(x) - f_s(x)\big)\Big]_+.$$

Then for any $\lambda = (\lambda_1, \cdots, \lambda_k)^T$, where $\forall i\ \lambda_i \in \mathcal{H}_\kappa$, we have g's Fenchel conjugate as:

$$g^*(\lambda) = \begin{cases} -\sum_{i,j} \alpha_{ij} & \text{if } \lambda_{r_i} + \sum_j \alpha_{ij}\kappa(x,\cdot) = 0 \text{ and } \lambda_{s_j} - \sum_i \alpha_{ij}\kappa(x,\cdot) = 0 \\ \infty & \text{otherwise} \end{cases},$$

where $\alpha = (\alpha_{ij}) \in \mathcal{A} = [A | A \in R_+^{\|Y\|} \times R_+^{(k-\|Y\|)}, \|A\|_1 \le 1]$ and $(r_i \times s_j) \in \mathcal{B} = Y \times ([k]/Y)$.

Proof The approach of our proof is similar to the method for proving the "Max-of-hinge" in Shalev-Shwartz (2007). First of all, it is not difficult to show that the loss function can be re-formulated as follows:

$$\begin{aligned} g(f) &= \max_{\alpha \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(f_{r_i}(x) - f_{s_j}(x)\big)\Big] \\ &= \max_{\alpha \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(\langle f_{r_i}(\cdot), \kappa(x,\cdot)\rangle - \langle f_{s_j}(\cdot), \kappa(x,\cdot)\rangle\big)\Big]. \end{aligned}$$

As a result, we have:

$$\begin{aligned} g^*(\lambda) &= \max_f \Big\{ \langle \lambda, f \rangle - g(f) \Big\} \\ &= \max_f \Big\{ \sum_{n=1}^k \langle \lambda_n, f_n \rangle - \max_{\alpha \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(\langle f_{r_i}(\cdot), \kappa(x,\cdot)\rangle - \langle f_{s_j}(\cdot), \kappa(x,\cdot)\rangle\big)\Big]\Big\}. \end{aligned}$$

For any $f_n, \lambda_n \in \mathcal{H}_\kappa$, they can be written as: $f_n = \beta_n \kappa(x,\cdot) + f_n^\perp, \lambda_n = \gamma_n \kappa(x,\cdot) + \lambda_n^\perp$, where $f_n^\perp, \lambda_n^\perp \in \mathcal{V}^\perp$, $\mathcal{V} = \overline{span\{\kappa(x,\cdot)\}}$. As a result, we have

$$g^*(\lambda) = \max_f \Big\{ \sum_{n=1}^k \Big(\langle \lambda_n^\perp, f_n^\perp \rangle + \beta_n \gamma_n \kappa(x,x)\Big) - \max_{\alpha \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(\beta_{r_i}\kappa(x,x) - \beta_{s_j}\kappa(x,x)\big)\Big]\Big\}.$$

When $\lambda_n^\perp \neq 0$, the $\max_{f^\perp} \langle \lambda_n^\perp, f_n^\perp \rangle$ will be ∞, resulting $g^*(\lambda) = \infty$. Otherwise, if $\lambda_n^\perp = 0$, $\forall n$, the term $f_n^\perp$ does not take effect for the objective; as a result, the optimal f_n can be written in the form of $\beta_n \kappa(x,\cdot)$ and the conjugate is computed as follows:

$$\begin{aligned} g^*(\lambda) &= \max_{\beta_n} \Big\{ \sum_{n=1}^k \beta_n \gamma_n \kappa(x,x) - \max_{\alpha \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \Big[1 - \big(\beta_{r_i}\kappa(x,x) - \beta_{s_j}\kappa(x,x)\big)\Big]\Big\} \\ &= \max_{\beta_n} \min_{\alpha \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \Big\{ \sum_{n=1}^k \langle \lambda_n, \beta_n \kappa(x,\cdot)\rangle - \sum_{i,j} \alpha_{ij} \Big[1 - \big(\langle \beta_{r_i}\kappa(x,\cdot), \kappa(x,\cdot)\rangle - \langle \beta_{s_j}\kappa(x,\cdot), \kappa(x,\cdot)\rangle\big)\Big]\Big\} \\ &= \min_{\alpha \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \max_{\beta_n} \Big\{ \sum_{n=1}^k \langle \lambda_n, \beta_n \kappa(x,\cdot)\rangle - \sum_{i,j} \alpha_{ij} \Big[1 - \big(\langle \beta_{r_i}\kappa(x,\cdot), \kappa(x,\cdot)\rangle - \langle \beta_{s_j}\kappa(x,\cdot), \kappa(x,\cdot)\rangle\big)\Big]\Big\} \\ &= \min_{\alpha \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \Big\{ -\sum_{i,j} \alpha_{ij} + \max_{\beta_n} \Big[\sum_{r_i} \langle \beta_{r_i}\kappa(x,\cdot), \lambda_{r_i} + \sum_j \alpha_{ij}\kappa(x,\cdot)\rangle + \sum_{s_j} \langle \beta_{s_j}\kappa(x,\cdot), \lambda_{s_j} - \sum_i \alpha_{ij}\kappa(x,\cdot)\rangle \Big]\Big\}. \end{aligned}$$

The fourth equality is guaranteed by the strong max-min property (Boyd and Vandenberghe, 2004), and more importantly, we can see that only when α satisfies $\lambda_{r_i} + \sum_j \alpha_{ij}\kappa(x,\cdot) = 0$ and $\lambda_{s_j} - \sum_i \alpha_{ij}\kappa(x,\cdot) = 0$, the second term in the equation above will be zero; otherwise, it will be ∞. Therefore, we have the resulting Fenchel conjugate of $g(f)$ as follows:

$$g^*(\lambda) = \begin{cases} -\sum_{i,j}\alpha_{ij} & \lambda_{r_i} + \sum_j \alpha_{ij}\kappa(x,\cdot) = 0 \text{ and } \lambda_{s_j} - \sum_i \alpha_{ij}\kappa(x,\cdot) = 0 \\ \infty & \text{otherwise} \end{cases}.$$

■

Given the above Fenchel dual of loss function, we can derive the dual for the optimization problem given on the right-hand side of Equation (4), as given in the following lemma.

Lemma 3 *Suppose the complexity measure function is given as $F(f) = \sum_{i=1}^k \frac{1}{2}\|f_i\|^2_{\mathcal{H}_\kappa}$, and we set α_{ij} to zeros for $\forall(i,j) \in \{[Y_t \times ([k]/Y_t)]/(r_t,s_t)\}$, where (r_t,s_t) is defined in Equation (2). Then the dual objective function for optimization given on the right-hand side of Equation (4) can be expressed as follows:*

$$D(\gamma_1,\cdots,\gamma_T) = -\sum_{i=1}^k \frac{1}{2}\|\sum_{t=1}^T \sigma(i,t)\gamma_t\kappa(x_t,\cdot)\|^2 + \sum_{t=1}^T \gamma_t,$$

where $\gamma_t \in [0,C]$ *and* $\sigma(i,t) = \begin{cases} 1 & \textit{if } i = r_t \\ -1 & \textit{if } i = s_t \\ 0 & \textit{otherwise} \end{cases}.$

Proof The proof here resembles the one in the section 3.2 of Shalev-Shwartz (2007). Firstly, we note that the problem (4) is equivalent to the following:

$$\inf_{f_0,f_1,\cdots,f_T} \left(F(f_0) + \sum_{t=1}^T Cg_t(f_t)\right) \text{ s.t. } f_0,f_t \in \mathcal{H}_\kappa \text{ and } \forall t \in [T], f_t = f_0.$$

By introducing T function vectors $\lambda_1,\cdots,\lambda_T$, in which each $\lambda_t = (\lambda_{t,1},\cdots,\lambda_{t,k}) \in H_\kappa$ is a Lagrange multipliers for the constraint $f_t = f_0$, we can obtain the following Lagrangian:

$$\mathcal{L}(f_0,\cdots,f_T,\lambda_1,\cdots,\lambda_T) = F(f_0) + \sum_{t=1}^T Cg_t(f_t) + \sum_{t=1}^T \langle\lambda_t, f_0 - f_t\rangle.$$

The dual objective function can be derived as follows:

$$\begin{aligned} D(\lambda_1,\cdots,\lambda_T) &= \inf_{f_0,f_1,\cdots,f_T} \mathcal{L}(f_0,\cdots,f_T,\lambda_1,\cdots,\lambda_T) \\ &= -\sup_{f_0}\left[\langle f_0, -\sum_{t=1}^T \lambda_t\rangle - F(f_0)\right] - \sum_{t=1}^T \sup_{f_t}\left[\langle f_t,\lambda_t\rangle - Cg_t(f_t)\right] \\ &= -F^*(-\sum_{t=1}^T \lambda_t) - \sum_{t=1}^T (Cg_t)^*(\lambda_t) = -F^*(-\sum_{t=1}^T \lambda_t) - \sum_{t=1}^T Cg_t^*(\frac{\lambda_t}{C}). \end{aligned}$$

Because $F(f) = \sum_{i=1}^k \frac{1}{2}\|f_i\|^2_{\mathcal{H}_\kappa}$, we have $F^* = F$. The dual problem thus becomes:

$$D(\lambda_1,\cdots,\lambda_T) = -\sum_{i=1}^k \frac{1}{2}\|-\sum_{t=1}^T \lambda_{t,i}\|^2_{\mathcal{H}_\kappa} - \sum_{t=1}^T Cg_t^*(\frac{\lambda_t}{C}).$$

Because we want to maximize the dual objective, according to Lemma 2, we should set

$$\frac{\lambda_{t,r_i^t}}{C} + \sum_j \alpha_{ij}^t k(x_t,\cdot) = 0, \quad \frac{\lambda_{t,s_j^t}}{C} - \sum_t \alpha_{ij}^t k(x_t,\cdot) = 0,$$

where $(\alpha_{ij}^t) \in \mathcal{A}_t, (r_i^t \times s_j^t) \in \mathcal{B}_t$, $\mathcal{A}_t = [A|A \in R_+^{\|Y_t\|} \times R_+^{(k-\|Y_t\|)}, \|A\|_1 \leq 1]$ and $\mathcal{B}_t = Y_t \times ([k]/Y_t)$. Furthermore, we set α_{ij} to zeros for $\forall (i,j) \in \{[Y_t \times ([k]/Y_t)]/(r_t,s_t)\}$. For simplicity, we denote α_{r_t,s_t}^t as $\frac{\gamma_t}{C}$. As a result, the dual objective function becomes

$$D(\gamma_1,\cdots,\gamma_T) = -\sum_{i=1}^k \frac{1}{2}\|\sum_{t=1}^T \sigma(i,t)\gamma_t\kappa(x_t,\cdot)\|^2 + \sum_{t=1}^T \gamma_t,$$

where $\gamma_t \in [0,C]$ and $\sigma(i,t) = \begin{cases} 1 & \text{if } i = r_t \\ -1 & \text{if } i = s_t \\ 0 & \text{otherwise} \end{cases}$. ■

By applying Lemma 3, we thus have the dual objective function for the t-th step as:

$$D_t(\gamma_1,\cdots,\gamma_t) = -\sum_{i=1}^k \frac{1}{2}\|\sum_{j=1}^t \sigma(i,j)\gamma_j k(x_j,\cdot)\|^2 + \sum_{j=1}^t \gamma_j. \tag{10}$$

Now our goal is to derive the dual ascent guaranteed by the proposed double updating scheme. When pair (x_a,Y_a) is misclassified by the prediction function $f_t = (f_{t,1},\cdots,f_{t,k})$, we will perform the update on the prediction function. Assume we conduct a double updating for (x_a,Y_a) and some auxiliary example (x_b,Y_b), we can prove Proposition 4 as follows.

Proof According to Equation (10) obtained by Lemma 3, before performing the double updating, the value of the dual function is expressed as:

$$D_{t-1} = -\sum_{i=1}^k \frac{1}{2}\|\sum_{j=1}^{t-1} \sigma(i,j)\hat{\gamma}_j k(x_j,\cdot)\|^2 + \sum_{j=1}^{t-1}\hat{\gamma}_j = -\sum_{i=1}^k \frac{1}{2}\|f_{t-1,i}\|^2 + \sum_{j=1}^{t-1}\hat{\gamma}_j,$$

where $\hat{\gamma}_j$'s denote the weights of the prediction function f_{t-1} before the updating. After performing the dual update, the value of the new dual function can be written as:

$$D_t = -\sum_{i=1}^k \frac{1}{2}\|f_{t-1,i} + \sigma(i,a)\gamma_a k(x_a,\cdot) + \sigma(i,b)d_{\gamma_b}k(x_b,\cdot)\|^2 + \sum_{j=1}^{t-1}\hat{\gamma}_j + \gamma_a + d_{\gamma_b}.$$

Hence, the dual ascent is computed as follows:

$$\begin{aligned}\Delta D = D_t - D_{t-1} \quad &= \quad \gamma_a\Big(1 - \big(f_{t-1,r_a}(x_a) - f_{t-1,s_a}(x_a)\big)\Big) + d_{\gamma_b}\Big(1 - \big(f_{t-1,r_b}(x_b) - f_{t-1,s_b}(x_b)\big)\Big) \\ &\qquad -\gamma_a^2 s_a - d_{\gamma_b}^2 s_b - \sum_{i=1}^k \sigma(i,a)\sigma(i,b)\gamma_a d_{\gamma_b} k(x_a,x_b)\ .\end{aligned}$$

■

References

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

Antoine Bordes, Léon Bottou, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with larank. In *Proceedings of the 24th International Conference on Machine learning (ICML'07)*, pages 89–96, 2007.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 13 (NIPS)*, pages 409–415, 2000.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. on Inf. Theory*, 50(9):2050–2057, 2004.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

Koby Crammer and Yoram Singer. Loss bounds for online category ranking. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT'05)*, pages 48–62, 2005.

Koby Crammer, Jaz S. Kandola, and Yoram Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Koby Crammer, Mark Dredze, and Fernando Pereira. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 345–352, 2008.

Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.*, 37(5):1342–1372, 2008. ISSN 0097-5397.

Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*, pages 264–271, 2008.

Michael Fink, Shai Shalev-Shwartz, Yoram Singer, and Shimon Ullman. Online multiclass learning by interclass hypothesis sharing. In *Proceedings of the 25th International Conference on Machine learning (ICML'06)*, pages 313–320, 2006.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296, 1999.

Claudio Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

Jyrki Kivinen and Manfred K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC'95)*, pages 209–218, 1995.

Jyrki Kivinen, Alex J. Smola, and Robert C. Williamson. Online learning with kernels. In *Advances in Neural Information Processing Systems (NIPS)*, pages 785–792, 2001.

Yi Li and Philip M. Long. The relaxed online maximum margin algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 498–504, 1999.

Francesco Orabona, Joseph Keshet, and Barbara Caputo. The projectron: a bounded kernel-based perceptron. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML'08)*, pages 720–727, 2008.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

Shai Shalev-Shwartz. Online learning:theory, algorithms, and applications. In *Ph.D thesis*, 2007.

Shai Shalev-Shwartz and Yoram Singer. Online learning meets optimization in the dual. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT'06)*, pages 423–437, 2006.

Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Jason Weston and Antoine Bordes. Online (and offline) on an even tighter budget. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS'05)*, pages 413–420, 2005.

Peilin Zhao, Steven C. H. Hoi, and Rong Jin. Duol: A double updating approach for online learning. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 2259–2267, 2009.

Journal of Machine Learning Research 12 (2011) 1617-1653 Submitted 5/10; Revised 2/11; Published 5/11

Learning High-Dimensional Markov Forest Distributions: Analysis of Error Rates

Vincent Y. F. Tan VTAN@WISC.EDU
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706

Animashree Anandkumar A.ANANDKUMAR@UCI.EDU
Center for Pervasive Communications and Computing
Electrical Engineering and Computer Science
University of California, Irvine
Irvine, CA 92697

Alan S. Willsky WILLSKY@MIT.EDU
Stochastic Systems Group
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Editor: Marina Meilă

Abstract

The problem of learning forest-structured discrete graphical models from i.i.d. samples is considered. An algorithm based on pruning of the Chow-Liu tree through adaptive thresholding is proposed. It is shown that this algorithm is both structurally consistent and risk consistent and the error probability of structure learning decays faster than any polynomial in the number of samples under fixed model size. For the high-dimensional scenario where the size of the model d and the number of edges k scale with the number of samples n, sufficient conditions on (n,d,k) are given for the algorithm to satisfy structural and risk consistencies. In addition, the extremal structures for learning are identified; we prove that the independent (resp., tree) model is the hardest (resp., easiest) to learn using the proposed algorithm in terms of error rates for structure learning.

Keywords: graphical models, forest distributions, structural consistency, risk consistency, method of types

1. Introduction

Graphical models (also known as Markov random fields) have a wide range of applications in diverse fields such as signal processing, coding theory and bioinformatics. See Lauritzen (1996), Wainwright and Jordan (2003) and references therein for examples. Inferring the structure and parameters of graphical models from samples is a starting point in all these applications. The structure of the model provides a quantitative interpretation of relationships amongst the given collection of random variables by specifying a set of conditional independence relationships. The parameters of the model quantify the strength of these interactions among the variables.

The challenge in learning graphical models is often compounded by the fact that typically only a small number of samples are available relative to the size of the model (dimension of data). This is referred to as the high-dimensional learning regime, which differs from classical statistics where a large number of samples of fixed dimensionality are available. As a concrete example, in order to analyze the effect of environmental and genetic factors on childhood asthma, clinician scientists in Manchester, UK have been conducting a longitudinal birth-cohort study since 1997 (Custovic et al., 2002; Simpson et al., 2010). The number of variables collected is of the order of $d \approx 10^6$ (dominated by the genetic data) but the number of children in the study is small ($n \approx 10^3$). The paucity of subjects in the study is due in part to the prohibitive cost of collecting high-quality clinical data from willing participants.

In order to learn high-dimensional graphical models, it is imperative to strike the right balance between data fidelity and overfitting. To ameliorate the effect of overfitting, the samples are often fitted to a *sparse graphical model* (Wainwright and Jordan, 2003), with a small number of edges. One popular and tractable class of sparse graphical models is the set of tree[1] models. When restricted to trees, the Chow-Liu algorithm (Chow and Liu, 1968; Chow and Wagner, 1973) provides an efficient implementation of the maximum-likelihood (ML) procedure to learn the structure from independent samples. However, in the high-dimensional regime, even a tree may overfit the data (Liu et al., 2011). In this paper, we consider learning high-dimensional, forest-structured (discrete) graphical models from a given set of samples.

For learning the forest structure, the ML (Chow-Liu) algorithm does not produce a consistent estimate since ML favors richer model classes and hence, outputs a tree in general. We propose a consistent algorithm called CLThres, which has a thresholding mechanism to prune "weak" edges from the Chow-Liu tree. We provide tight bounds on the *overestimation* and *underestimation* errors, that is, the error probability that the output of the algorithm has more or fewer edges than the true model.

1.1 Main Contributions

This paper contains three main contributions. Firstly, we propose an algorithm named CLThres and prove that it is structurally consistent when the true distribution is forest-structured. Secondly, we prove that CLThres is risk consistent, meaning that the risk under the estimated model converges to the risk of the *forest projection*[2] of the underlying distribution, which may not be a forest. We also provide precise convergence rates for structural and risk consistencies. Thirdly, we provide conditions for the consistency of CLThres in the high-dimensional setting.

We first prove that CLThres is structurally consistent, i.e., as the number of samples grows for a fixed model size, the probability of learning the incorrect structure (set of edges), decays to zero for a fixed model size. We show that the error rate is in fact, dominated by the rate of decay of the overestimation error probability.[3] We use an information-theoretic technique known as the *method of types* (Cover and Thomas, 2006, Ch. 11) as well as a recently-developed technique known as Euclidean information theory (Borade and Zheng, 2008). We provide an upper bound on the error probability by using convex duality to find a surprising connection between the overestimation error

1. A *tree* is a *connected*, acyclic graph. We use the term *proper forest* to denote the set of *disconnected*, acyclic graphs.
2. The forest projection is the forest-structured graphical model that is closest in the KL-divergence sense to the true distribution. We define this distribution formally in (12).
3. The overestimation error probability is the probability that the number of edges learned exceeds the true number of edges. The underestimation error is defined analogously.

rate and a semidefinite program (Vandenberghe and Boyd, 1996) and show that the overestimation error in structure learning decays faster than any polynomial in n for a fixed data dimension d.

We then consider the high-dimensional scenario and provide sufficient conditions on the growth of (n,d) (and also the true number of edges k) to ensure that CLThres is structurally consistent. We prove that even if d grows faster than any polynomial in n (and in fact close to exponential in n), structure estimation remains consistent. As a corollary from our analyses, we also show that for CLThres, independent models (resp., tree models) are the "hardest" (resp., "easiest") to learn in the sense that the asymptotic error rate is the highest (resp., lowest), over all models with the same scaling of (n,d). Thus, the empty graph and connected trees are the extremal forest structures for learning. We also prove that CLThres is risk consistent, i.e., the risk of the estimated forest distribution converges to the risk of the forest projection of the true model at a rate of $O_p(d \log d / n^{1-\gamma})$ for any $\gamma > 0$. We compare and contrast this rate to existing results such as Liu et al. (2011). Note that for this result, the true probability model does not need to be a forest-structured distribution. Finally, we use CLThres to learn forest-structured distributions given synthetic and real-world data sets and show that in the finite-sample case, there exists an inevitable trade-off between the underestimation and overestimation errors.

1.2 Related Work

There are many papers that discuss the learning of graphical models from data. See Dudik et al. (2004), Lee et al. (2006), Abbeel et al. (2006), Wainwright et al. (2006), Meinshausen and Buehlmann (2006), Johnson et al. (2007), and references therein. Most of these methods pose the learning problem as a parameterized convex optimization problem, typically with a regularization term to enforce sparsity in the learned graph. Consistency guarantees in terms of n and d (and possibly the maximum degree) are provided. Information-theoretic limits for learning graphical models have also been derived in Santhanam and Wainwright (2008). In Zuk et al. (2006), bounds on the error rate for learning the structure of Bayesian networks using the Bayesian Information Criterion (BIC) were provided. Bach and Jordan (2003) learned tree-structured models for solving the independent component analysis (ICA) problem. A PAC analysis for learning thin junction trees was given in Chechetka and Guestrin (2007). Meilă and Jordan (2000) discussed the learning of graphical models from a different perspective; namely that of learning mixtures of trees via an expectation-maximization procedure.

By using the theory of large-deviations (Dembo and Zeitouni, 1998; Den Hollander, 2000), we derived and analyzed the error exponent for learning trees for discrete (Tan et al., 2011) and Gaussian (Tan et al., 2010a) graphical models. The error exponent is a quantitative measure of performance of the learning algorithm since a larger exponent implies a faster decay of the error probability. However, the analysis does not readily extend to learning forest models and furthermore it was for the scenario when number of variables d does not grow with the number of samples n. In addition, we also posed the structure learning problem for trees as a composite hypothesis testing problem (Tan et al., 2010b) and derived a closed-form expression for the Chernoff-Stein exponent in terms of the mutual information on the bottleneck edge.

In a paper that is closely related to ours, Liu et al. (2011) derived consistency (and sparsistency) guarantees for learning tree and forest models. The pairwise joint distributions are modeled using kernel density estimates, where the kernels are Hölder continuous. This differs from our approach since we assume that each variable can only take finitely many values, leading to stronger results on error rates for structure learning via the method of types, a powerful proof technique in information theory and statistics. We compare our convergence rates to these related works in Section 6. Furthermore, the algorithm suggested in both papers uses a subset (usually half) of the data set to learn the full tree model and then uses the remaining subset to prune the model based on the log-likelihood on the held-out set. We suggest a more direct and consistent method based on thresholding, which uses the *entire* data set to learn and prune the model without recourse to validation on a held-out data set. It is well known that validation is both computationally expensive (Bishop, 2008, pp. 33) and a potential waste of valuable data which may otherwise be employed to learn a better model. In Liu et al. (2011), the problem of estimating forests with restricted component sizes was considered and was proven to be NP-hard. We do not restrict the component size in this paper but instead attempt to learn the model with the minimum number of edges which best fits the data.

Our work is also related to and inspired by the vast body of literature in information theory and statistics on Markov order estimation. In these works, the authors use various regularization and model selection schemes to find the optimal order of a Markov chain (Merhav et al., 1989; Finesso et al., 1996; Csiszár and Shields, 2000), hidden Markov model (Gassiat and Boucheron, 2003) or exponential family (Merhav, 1989). We build on some of these ideas and proof techniques to identify the correct set of edges (and in particular the number of edges) in the forest model and also to provide strong theoretical guarantees of the rate of convergence of the estimated forest-structured distribution to the true one.

1.3 Organization of Paper

This paper is organized as follows: We define the mathematical notation and formally state the problem in Section 2. In Section 3, we describe the algorithm in full detail, highlighting its most salient aspect—the thresholding step. We state our main results on error rates for structure learning in Section 4 for a fixed forest-structured distribution. We extend these results to the high-dimensional case when (n,d,k) scale in Section 5. Extensions to rates of convergence of the estimated distribution, that is, the order of risk consistency, are discussed briefly in Section 6. Numerical simulations on synthetic and real data are presented in Section 7. Finally, we conclude the discussion in Section 8. The proofs of the majority of the results are provided in the appendices.

2. Preliminaries and Problem Formulation

Let $G=(V,E)$ be an undirected graph with vertex (or node) set $V:=\{1,\dots,d\}$ and edge set $E\subset\binom{V}{2}$ and let $\mathrm{nbd}(i):=\{j\in V:(i,j)\in E\}$ be the set of neighbors of vertex i. Let the set of labeled *trees* (connected, acyclic graphs) with d nodes be $\mathcal{T}^d$ and let the set of *forests* (acyclic graphs) with k edges and d nodes be $\mathcal{T}^d_k$ for $0\le k\le d-1$. The set of forests includes all the trees. We reserve the term *proper forests* for the set of disconnected acylic graphs $\cup_{k=0}^{d-2}\mathcal{T}^d_k$. We also use the notation $\mathcal{F}^d:=\cup_{k=0}^{d-1}\mathcal{T}^d_k$ to denote the set of labeled forests with d nodes.

A *graphical model* (Lauritzen, 1996) is a family of multivariate probability distributions (probability mass functions) in which each distribution factorizes according to a given undirected graph and where each variable is associated to a node in the graph. Let $\mathcal{X}=\{1,\dots,r\}$ (where $2\le r<\infty$) be a finite set and $\mathcal{X}^d$ the d-fold Cartesian product of the set $\mathcal{X}$. As usual, let $\mathcal{P}(\mathcal{X}^d)$ denote the probability simplex over the alphabet $\mathcal{X}^d$. We say that the random vector $\mathbf{X}=(X_1,\dots,X_d)$ with distribution $Q \in \mathcal{P}(\mathcal{X}^d)$ is *Markov on* the graph $G = (V, E)$ if

$$Q(x_i | x_{\mathrm{nbd}(i)}) = Q(x_i | x_{V \setminus i}), \qquad \forall\, i \in V, \tag{1}$$

where $x_{V \setminus i}$ is the collection of variables excluding variable i. Equation (1) is known as the *local Markov property* (Lauritzen, 1996). In this paper, we always assume that graphs are *minimal representations* for the corresponding graphical model, that is, if Q is Markov on G, then G has the smallest number of edges for the conditional independence relations in (1) to hold. We say the distribution Q is a *forest-structured distribution* if it is Markov on a forest. We also use the notation $\mathcal{D}(\mathcal{T}_k^d) \subset \mathcal{P}(\mathcal{X}^d)$ to denote the set of d-variate distributions Markov on a forest with k edges. Similarly, $\mathcal{D}(\mathcal{F}^d)$ is the set of forest-structured distributions.

Let $P \in \mathcal{D}(\mathcal{T}_k^d)$ be a discrete forest-structured distribution Markov on $T_P = (V, E_P) \in \mathcal{T}_k^d$ (for some $k = 0, \ldots, d-1$). It is known that the joint distribution P factorizes as follows (Lauritzen, 1996; Wainwright and Jordan, 2003):

$$P(\mathbf{x}) = \prod_{i \in V} P_i(x_i) \prod_{(i,j) \in E_P} \frac{P_{i,j}(x_i, x_j)}{P_i(x_i) P_j(x_j)},$$

where $\{P_i\}_{i \in V}$ and $\{P_{i,j}\}_{(i,j) \in E_P}$ are the node and pairwise marginals which are assumed to be positive everywhere.

The mutual information (MI) of two random variables X_i and X_j with joint distribution $P_{i,j}$ is the function $I(\cdot) : \mathcal{P}(\mathcal{X}^2) \to [0, \log r]$ defined as

$$I(P_{i,j}) := \sum_{(x_i, x_j) \in \mathcal{X}^2} P_{i,j}(x_i, x_j) \log \frac{P_{i,j}(x_i, x_j)}{P_i(x_i) P_j(x_j)}. \tag{2}$$

This notation for mutual information differs from the usual $I(X_i; X_j)$ used in Cover and Thomas (2006); we emphasize the dependence of I on the joint distribution $P_{i,j}$. The *minimum mutual information* in the forest, denoted as $I_{\min} := \min_{(i,j) \in E_P} I(P_{i,j})$ will turn out to be a fundamental quantity in the subsequent analysis. Note from our minimality assumption that $I_{\min} > 0$ since all edges in the forest have positive mutual information (none of the edges are degenerate). When we consider the scenario where d grows with n in Section 5, we assume that $I_{\min}$ is *uniformly* bounded away from zero.

2.1 Problem Statement

We now state the basic problem formally. We are given a set of i.i.d. samples, denoted as $\mathbf{x}^n := \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Each sample $\mathbf{x}_l = (x_{l,1}, \ldots, x_{l,d}) \in \mathcal{X}^d$ is drawn independently from $P \in \mathcal{D}(\mathcal{T}_k^d)$ a forest-structured distribution. From these samples, and the prior knowledge that the undirected graph is acyclic (but not necessarily connected), estimate the true set of edges E_P as well as the true distribution P consistently.

3. The Forest Learning Algorithm: CLThres

We now describe our algorithm for estimating the edge set E_P and the distribution P. This algorithm is a modification of the celebrated Chow-Liu algorithm for maximum-likelihood (ML) learning of tree-structured distributions (Chow and Liu, 1968). We call our algorithm CLThres which stands for *Chow-Liu with Thresholding*.

The inputs to the algorithm are the set of samples $\mathbf{x}^n$ and a *regularization* sequence $\{\varepsilon_n\}_{n\in\mathbb{N}}$ (to be specified precisely later) that typically decays to zero, that is, $\lim_{n\to\infty}\varepsilon_n = 0$. The outputs are the estimated edge set, denoted $\widehat{E}_{\widehat{k}_n}$, and the estimated distribution, denoted P^*.

1. Given $\mathbf{x}^n$, calculate the set of *pairwise empirical distributions*[4] (or *pairwise types*) $\{\widehat{P}_{i,j}\}_{i,j\in V}$. This is just a normalized version of the counts of each observed symbol in $\mathcal{X}^2$ and serves as a set of sufficient statistics for the estimation problem. The dependence of $\widehat{P}_{i,j}$ on the samples $\mathbf{x}^n$ is suppressed.

2. Form the set of *empirical mutual information* quantities:
$$I(\widehat{P}_{i,j}) := \sum_{(x_i,x_j)\in\mathcal{X}^2} \widehat{P}_{i,j}(x_i,x_j)\log\frac{\widehat{P}_{i,j}(x_i,x_j)}{\widehat{P}_i(x_i)\widehat{P}_j(x_j)},$$
for $1 \le i,j \le d$. This is a consistent estimator of the true mutual information in (2).

3. Run a max-weight spanning tree (MWST) algorithm (Prim, 1957; Kruskal, 1956) to obtain an estimate of the edge set:
$$\widehat{E}_{d-1} := \operatorname*{argmax}_{E:T=(V,E)\in\mathcal{T}^d} \sum_{(i,j)\in E} I(\widehat{P}_{i,j}).$$
Let the estimated edge set be $\widehat{E}_{d-1} := \{\widehat{e}_1,\dots,\widehat{e}_{d-1}\}$ where the edges $\widehat{e}_i$ are sorted according to decreasing empirical mutual information values. We index the edge set by $d-1$ to emphasize that it has $d-1$ edges and hence is connected. We denote the sorted empirical mutual information quantities as $I(\widehat{P}_{\widehat{e}_1}) \ge \dots \ge I(\widehat{P}_{\widehat{e}_{d-1}})$. These first three steps constitute the Chow-Liu algorithm (Chow and Liu, 1968).

4. Estimate the true number of edges using the *thresholding estimator*:
$$\widehat{k}_n := \operatorname*{argmin}_{1\le j\le d-1}\left\{I(\widehat{P}_{\widehat{e}_j}) : I(\widehat{P}_{\widehat{e}_j}) \ge \varepsilon_n, I(\widehat{P}_{\widehat{e}_{j+1}}) \le \varepsilon_n\right\}. \tag{3}$$
If there exists an empirical mutual information $I(\widehat{P}_{\widehat{e}_j})$ such that $I(\widehat{P}_{\widehat{e}_j}) = \varepsilon_n$, break the tie arbitrarily.[5]

5. Prune the tree by retaining only the top $\widehat{k}_n$ edges, that is, define the *estimated edge set* of the forest to be
$$\widehat{E}_{\widehat{k}_n} := \{\widehat{e}_1,\dots,\widehat{e}_{\widehat{k}_n}\},$$
where $\{\widehat{e}_i : 1 \le i \le d-1\}$ is the ordered edge set defined in Step 3. Define the estimated forest to be $\widehat{T}_{\widehat{k}_n} := (V, \widehat{E}_{\widehat{k}_n})$.

4. In this paper, the terms *empirical distribution* and *type* are used interchangeably.

5. Here were allow a bit of imprecision by noting that the non-strict inequalities in (3) simplify the subsequent analyses because the constraint sets that appear in optimization problems will be closed, hence compact, insuring the existence of optimizers.

6. Finally, define the estimated distribution P^* to be the *reverse I-projection* (Csiszár and Matúš, 2003) of the joint type $\widehat{P}$ onto $\widehat{T}_{\widehat{k}_n}$, that is,

$$P^*(\mathbf{x}) := \operatorname*{argmin}_{Q \in \mathcal{D}(\widehat{T}_{\widehat{k}_n})} D(\widehat{P} \,||\, Q).$$

   It can easily be shown that the projection can be expressed in terms of the marginal and pairwise joint types:

$$P^*(\mathbf{x}) = \prod_{i \in V} \widehat{P}_i(x_i) \prod_{(i,j) \in \widehat{E}_{\widehat{k}_n}} \frac{\widehat{P}_{i,j}(x_i, x_j)}{\widehat{P}_i(x_i)\widehat{P}_j(x_j)}.$$

Intuitively, CLThres first constructs a connected tree $(V, \widehat{E}_{d-1})$ via Chow-Liu (in Steps 1–3) before pruning the weak edges (with small mutual information) to obtain the final structure $\widehat{E}_{\widehat{k}_n}$. The estimated distribution P^* is simply the ML estimate of the parameters subject to the constraint that P^* is Markov on the learned tree $\widehat{T}_{\widehat{k}_n}$.

Note that if Step 4 is omitted and $\widehat{k}_n$ is defined to be $d-1$, then CLThres simply reduces to the Chow-Liu ML algorithm. Of course Chow-Liu, which outputs a tree, is guaranteed to fail (not be structurally consistent) if the number of edges in the true model $k < d-1$, which is the problem of interest in this paper. Thus, Step 4, a model selection step, is essential in estimating the true number of edges k. This step is a generalization of the test for independence of discrete memoryless sources discussed in Merhav (1989). In our work, we exploit the fact that the empirical mutual information $I(\widehat{P}_{\widehat{e}_j})$ corresponding to a pair of independent variables $\widehat{e}_j$ will be very small when n is large, thus a thresholding procedure using the (appropriately chosen) regularization sequence $\{\varepsilon_n\}$ will remove these edges. In fact, the subsequent analysis allows us to conclude that Step 4, in a formal sense, *dominates* the error probability in structure learning. CLThres is also efficient as shown by the following result.

Proposition 1 (Complexity of CLThres) CLThres *runs in time* $O((n + \log d)d^2)$.

Proof The computation of the sufficient statistics in Steps 1 and 2 requires $O(nd^2)$ operations. The MWST algorithm in Step 3 requires at most $O(d^2 \log d)$ operations (Prim, 1957). Steps 4 and 5 simply require the sorting of the empirical mutual information quantities on the learned tree which only requires $O(\log d)$ computations. ■

4. Structural Consistency For Fixed Model Size

In this section, we keep d and k fixed and consider a probability model P, which is assumed to be Markov on a forest in $\mathcal{T}_k^d$. This is to gain better insight into the problem before we analyze the high-dimensional scenario in Section 5 where d and k scale[6] with the sample size n. More precisely, we are interested in quantifying the rate at which the probability of the error event of structure learning

$$\mathcal{A}_n := \left\{ \mathbf{x}^n \in (\mathcal{X}^d)^n : \widehat{E}_{\widehat{k}_n}(\mathbf{x}^n) \neq E_P \right\} \tag{4}$$

6. In that case P must also scale, that is, we learn a *family* of models as d and k scale.

decays to zero as n tends to infinity. Recall that $\widehat{E}_{\widehat{k}_n}$, with cardinality $\widehat{k}_n$, is the learned edge set by using CLThres. As usual, P^n is the n-fold product probability measure corresponding to the forest-structured distribution P.

Before stating the main result of this section in Theorem 3, we first state an auxiliary result that essentially says that if one is provided with oracle knowledge of $I_{\min}$, the minimum mutual information in the forest, then the problem is greatly simplified.

Proposition 2 (Error Rate with knowledge of $I_{\min}$) *Assume that $I_{\min}$ is known in* CLThres. *Then by letting the regularization sequence be $\varepsilon_n = I_{\min}/2$ for all n, we have*

$$\lim_{n\to\infty} \frac{1}{n} \log P^n(\mathcal{A}_n) < 0, \tag{5}$$

that is, the error probability decays exponentially fast.

The proof of this theorem and all other results in the sequel can be found in the appendices.

Thus, the primary difficulty lies in estimating $I_{\min}$ or equivalently, the number of edges k. Note that if k is known, a simple modification to the Chow-Liu procedure by imposing the constraint that the final structure contains k edges will also yield exponential decay as in (5). However, in the realistic case where both $I_{\min}$ and k are unknown, we show in the rest of this section that we can design the regularization sequence ε_n in such a way that the rate of decay of $P^n(\mathcal{A}_n)$ decays almost exponentially fast.

4.1 Error Rate for Forest Structure Learning

We now state one of the main results in this paper. We emphasize that the following result is stated for a fixed forest-structured distribution $P \in \mathcal{D}(\mathcal{T}_k^d)$ so d and k are also fixed natural numbers.

Theorem 3 (Error Rate for Structure Learning) *Assume that the regularization sequence $\{\varepsilon_n\}_{n\in\mathbb{N}}$ satisfies the following two conditions:*

$$\lim_{n\to\infty} \varepsilon_n = 0, \qquad \lim_{n\to\infty} \frac{n\varepsilon_n}{\log n} = \infty. \tag{6}$$

Then, if the true model $T_P = (V, E_P)$ is a proper forest ($k < d-1$), there exists a constant $C_P \in (1,\infty)$ such that

$$-C_P \le \liminf_{n\to\infty} \frac{1}{n\varepsilon_n} \log P^n(\mathcal{A}_n) \tag{7}$$

$$\le \limsup_{n\to\infty} \frac{1}{n\varepsilon_n} \log P^n(\mathcal{A}_n) \le -1. \tag{8}$$

Finally, if the true model $T_P = (V, E_P)$ is a tree ($k = d-1$), then

$$\lim_{n\to\infty} \frac{1}{n} \log P^n(\mathcal{A}_n) < 0, \tag{9}$$

that is, the error probability decays exponentially fast.

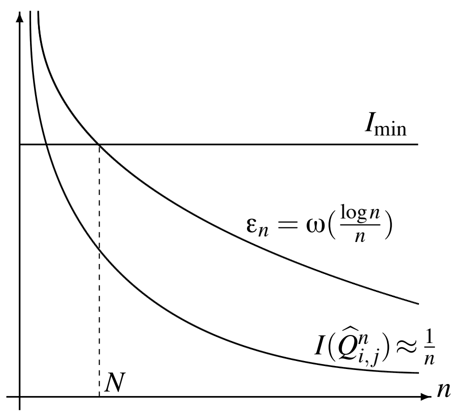


Figure 1: Graphical interpretation of the condition on ε_n. As $n \to \infty$, the regularization sequence ε_n will be smaller than $I_{\min}$ and larger than $I(\widehat{Q}^n_{i,j})$ with high probability.

4.2 Interpretation of Result

From (8), the rate of decay of the error probability for proper forests is subexponential but nonetheless can be made faster than any polynomial for an appropriate choice of ε_n. The reason for the subexponential rate is because of our lack of knowledge of $I_{\min}$, the minimum mutual information in the true forest T_P. For trees, the rate[7] is exponential ($\doteq \exp(-nF)$ for some positive constant F). Learning proper forests is thus, strictly "harder" than learning trees. The condition on ε_n in (6) is needed for the following intuitive reasons:

1. Firstly, (6) ensures that for all sufficiently large n, we have $\varepsilon_n < I_{\min}$. Thus, the true edges will be correctly identified by CLThres implying that with high probability, there will not be underestimation as $n \to \infty$.

2. Secondly, for two independent random variables X_i and X_j with distribution $Q_{i,j} = Q_i Q_j$, the sequence[8] $\sigma(I(\widehat{Q}^n_{i,j})) = \Theta(1/n)$, where $\widehat{Q}^n_{i,j}$ is the joint empirical distribution of n i.i.d. samples drawn from $Q_{i,j}$. Since the regularization sequence $\varepsilon_n = \omega(\log n/n)$ has a slower rate of decay than $\sigma(I(\widehat{Q}^n_{i,j}))$, $\varepsilon_n > I(\widehat{Q}^n_{i,j})$ with high probability as $n \to \infty$. Thus, with high probability there will not be overestimation as $n \to \infty$.

See Figure 1 for an illustration of this intuition. The formal proof follows from a method of types argument and we provide an outline in Section 4.3. A convenient choice of ε_n that satisfies (6) is

$$\varepsilon_n := n^{-\beta}, \qquad \forall \beta \in (0,1). \tag{10}$$

Note further that the upper bound in (8) is also independent of P since it is equal to -1 for all P. Thus, (8) is a *universal* result for all forest distributions $P \in \mathcal{D}(\mathcal{F}^d)$. The intuition for this

7. We use the asymptotic notation from information theory $\doteq$ to denote equality to first order in the exponent. More precisely, for two positive sequences $\{a_n\}_{n\in\mathbb{N}}$ and $\{b_n\}_{n\in\mathbb{N}}$ we say that $a_n \doteq b_n$ iff $\lim_{n\to\infty} n^{-1}\log(a_n/b_n) = 0$.
8. The notation $\sigma(Z)$ denotes the standard deviation of the random variable Z. The fact that the standard deviation of the empirical MI $\sigma(I(\widehat{Q}^n_{i,j}))$ decays as $1/n$ can be verified by Taylor expanding $I(\widehat{Q}^n_{i,j})$ around $Q_{i,j} = Q_i Q_j$ and using the fact that the ML estimate converges at a rate of $n^{-1/2}$ (Serfling, 1980).

universality is because in the large-n regime, the typical way an error occurs is due to overestimation. The overestimation error results from testing whether pairs of random variables are independent and our asymptotic bound for the error probability of this test does not depend on the true distribution P.

The lower bound C_P in (7), defined in the proof in Appendix B, means that we cannot hope to do much better using CLThres if the original structure (edge set) is a proper forest. Together, (7) and (8) imply that the rate of decay of the error probability for structure learning is tight to within a constant factor in the exponent. We believe that the error rates given in Theorem 3 cannot, in general, be improved without knowledge of $I_{\min}$. We state a converse (a necessary lower bound on sample complexity) in Theorem 7 by treating the unknown forest graph as a uniform random variable over all possible forests of fixed size.

4.3 Proof Idea

The method of proof for Theorem 3 involves using the Gallager-Fano bounding technique (Fano, 1961, pp. 24) and the union bound to decompose the overall error probability $P^n(\mathcal{A}_n)$ into three distinct terms: (i) the rate of decay of the error probability for learning the top k edges (in terms of the mutual information quantities) correctly—known as the *Chow-Liu error*, (ii) the rate of decay of the *overestimation error* $\{\widehat{k}_n > k\}$ and (iii) the rate of decay of the *underestimation error* $\{\widehat{k}_n < k\}$. Each of these terms is upper bounded using a method of types (Cover and Thomas, 2006, Ch. 11) argument. It turns out, as is the case with the literature on Markov order estimation (e.g., Finesso et al., 1996), that bounding the overestimation error poses the greatest challenge. Indeed, we show that the underestimation and Chow-Liu errors have exponential decay in n. However, the overestimation error has subexponential decay ($\approx \exp(-n\varepsilon_n)$).

The main technique used to analyze the overestimation error relies on *Euclidean information theory* (Borade and Zheng, 2008) which states that if two distributions ν_0 and ν_1 (both supported on a common finite alphabet $\mathcal{Y}$) are close entry-wise, then various information-theoretic measures can be approximated locally by quantities related to Euclidean norms. For example, the KL-divergence $D(\nu_0 \,||\, \nu_1)$ can be approximated by the square of a weighted Euclidean norm:

$$D(\nu_0 \,||\, \nu_1) = \frac{1}{2} \sum_{a \in \mathcal{Y}} \frac{(\nu_0(a) - \nu_1(a))^2}{\nu_0(a)} + o(\|\nu_0 - \nu_1\|_\infty^2). \tag{11}$$

Note that if $\nu_0 \approx \nu_1$, then $D(\nu_0 \,||\, \nu_1)$ is close to the sum in (11) and the $o(\|\nu_0 - \nu_1\|_\infty^2)$ term can be neglected. Using this approximation and Lagrangian duality (Bertsekas, 1999), we reduce a non-convex I-projection (Csiszár and Matúš, 2003) problem involving information-theoretic quantities (such as divergence) to a relatively simple *semidefinite program* (Vandenberghe and Boyd, 1996) which admits a closed-form solution. Furthermore, the approximation in (11) becomes *exact* as $n \to \infty$ (i.e., $\varepsilon_n \to 0$), which is the asymptotic regime of interest. The full details of the proof can be found Appendix B.

4.4 Error Rate for Learning the Forest Projection

In our discussion thus far, P has been assumed to be Markov on a forest. In this subsection, we consider the situation when the underlying unknown distribution P is not forest-structured but we wish to learn its best forest approximation. To this end, we define the projection of P onto the set of forests (or *forest projection*) to be

$$\widetilde{P} := \operatorname*{argmin}_{Q \in \mathcal{D}(\mathcal{F}^d)} D(P \,||\, Q). \tag{12}$$

If there are multiple optimizing distribution, choose a projection $\widetilde{P}$ that is minimal, that is, its graph $T_{\widetilde{P}} = (V, E_{\widetilde{P}})$ has the *fewest number of edges* such that (12) holds. If we redefine the event $\mathcal{A}_n$ in (4) to be $\widetilde{\mathcal{A}}_n := \{\widehat{E}_{\widehat{k}_n} \neq E_{\widetilde{P}}\}$, we have the following analogue of Theorem 3.

Corollary 4 (Error Rate for Learning Forest Projection) *Let P be an arbitrary distribution and the event $\widetilde{\mathcal{A}}_n$ be defined as above. Then the conclusions in* (7)–(9) *in Theorem 3 hold if the regularization sequence $\{\varepsilon_n\}_{n\in\mathbb{N}}$ satisfies* (6).

5. High-Dimensional Structural Consistency

In the previous section, we considered learning a fixed forest-structured distribution P (and hence fixed d and k) and derived bounds on the error rate for structure learning. However, for most problems of practical interest, the number of data samples is small compared to the data dimension d (see the asthma example in the introduction). In this section, we prove sufficient conditions on the scaling of (n,d,k) for structure learning to remain consistent. We will see that even if d and k are much larger than n, under some reasonable regularity conditions, structure learning remains consistent.

5.1 Structure Scaling Law

To pose the learning problem formally, we consider a *sequence* of structure learning problems indexed by the number of data points n. For the particular problem indexed by n, we have a data set $\mathbf{x}^n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of size n where each sample $\mathbf{x}_l \in \mathcal{X}^d$ is drawn independently from an unknown d-variate forest-structured distribution $P^{(d)} \in \mathcal{D}(\mathcal{T}_k^d)$, which has d nodes and k edges and where d and k depend on n. This *high-dimensional* setup allows us to model and subsequently analyze how d and k can scale with n while maintaining consistency. We will sometimes make the dependence of d and k on n explicit, that is, $d = d_n$ and $k = k_n$.

In order to be able to learn the structure of the models we assume that

$$\text{(A1)} \quad I_{\inf} := \inf_{d\in\mathbb{N}} \min_{(i,j)\in E_{P^{(d)}}} I(P_{i,j}^{(d)}) > 0, \tag{13}$$

$$\text{(A2)} \quad \kappa := \inf_{d\in\mathbb{N}} \min_{x_i,x_j\in\mathcal{X}} P_{i,j}^{(d)}(x_i,x_j) > 0. \tag{14}$$

That is, assumptions (A1) and (A2) insure that there exists *uniform* lower bounds on the minimum mutual information and the minimum entry in the pairwise probabilities in the forest models as the size of the graph grows. These are typical regularity assumptions for the high-dimensional setting. See Wainwright et al. (2006) and Meinshausen and Buehlmann (2006) for example. We again emphasize that the proposed learning algorithm CLThres has knowledge of neither $I_{\inf}$ nor κ. Equipped with (A1) and (A2) and assuming the asymptotic behavior of ε_n in (6), we claim the following theorem for CLThres.

Theorem 5 (Structure Scaling Law) *There exists two finite, positive constants C_1, C_2 such that if*

$$n > \max\left\{(2\log(d-k))^{1+\zeta}, C_1 \log d, C_2 \log k\right\}, \tag{15}$$

for any $\zeta > 0$, then the error probability of incorrectly learning the sequence of edge sets $\{E_{P^{(d)}}\}_{d\in\mathbb{N}}$ tends to zero as $(n,d,k) \to \infty$. When the sequence of forests are trees, $n > C\log d$ (where $C := \max\{C_1, C_2\}$) suffices for high-dimensional structure recovery.

Thus, if the model parameters (n,d,k) all grow with n but $d = o(\exp(n/C_1))$, $k = o(\exp(n/C_2))$ and $d-k = o(\exp(n^{1-\beta}/2))$ (for all $\beta > 0$), consistent structure recovery is possible in high dimensions. In other words, the number of nodes d can grow faster than any polynomial in the sample size n. In Liu et al. (2011), the bivariate densities are modeled by functions from a Hölder class with exponent α and it was mentioned (in Theorem 4.3) that the number of variables can grow like $o(\exp(n^{\alpha/(1+\alpha)}))$ for structural consistency. Our result is somewhat stronger but we model the pairwise joint distributions as (simpler) probability mass functions (the alphabet $\mathcal{X}$ is a finite set).

5.2 Extremal Forest Structures

In this subsection, we study the extremal structures for learning, that is, the structures that, roughly speaking, lead to the largest and smallest error probabilities for structure learning. Define the sequence

$$h_n(P) := \frac{1}{n\varepsilon_n}\log P^n(\mathcal{A}_n), \quad \forall n \in \mathbb{N}. \tag{16}$$

Note that h_n is a function of both the number of variables $d = d_n$ and the number of edges $k = k_n$ in the models $P^{(d)}$ since it is a sequence indexed by n. In the next result, we assume (n,d,k) satisfies the scaling law in (15) and answer the following question: How does h_n in (16) depend on the number of edges k_n for a given d_n? Let $P_1^{(d)}$ and $P_2^{(d)}$ be two sequences of forest-structured distributions with a common number of nodes d_n and number of edges $k_n(P_1^{(d)})$ and $k_n(P_2^{(d)})$ respectively.

Corollary 6 (Extremal Forests) *Assume that* CLThres *is employed as the forest learning algorithm. As $n \to \infty$, $h_n(P_1^{(d)}) \le h_n(P_2^{(d)})$ whenever $k_n(P_1^{(d)}) \ge k_n(P_2^{(d)})$ implying that h_n is maximized when $P^{(d)}$ are product distributions (i.e., $k_n = 0$) and minimized when $P^{(d)}$ are tree-structured distributions (i.e., $k_n = d_n - 1$). Furthermore, if $k_n(P_1^{(d)}) = k_n(P_2^{(d)})$, then $h_n(P_1^{(d)}) = h_n(P_2^{(d)})$.*

Note that the corollary is intimately tied to the proposed algorithm CLThres. We are not claiming that such a result holds for all other forest learning algorithms. The intuition for this result is the following: We recall from the discussion after Theorem 3 that the overestimation error dominates the probability of error for structure learning. Thus, the performance of CLThres degrades with the number of missing edges. If there are very few edges (i.e., k_n is very small relative to d_n), the CLThres estimator is more likely to overestimate the number of edges as compared to if there are many edges (i.e., k_n/d_n is close to 1). We conclude that a distribution which is Markov on an *empty graph* (all variables are independent) is the *hardest* to learn (in the sense of Corollary 6 above). Conversely, *trees* are the *easiest* structures to learn.

5.3 Lower Bounds on Sample Complexity

Thus far, our results are for a specific algorithm CLThres for learning the structure of Markov forest distributions. At this juncture, it is natural to ask whether the scaling laws in Theorem 5 are the best possible over all algorithms (estimators). To answer this question, we limit ourselves to the scenario where the true graph T_P is a uniformly distributed chance variable[9] with probability measure $\mathbb{P}$. Assume two different scenarios:

(a) T_P is drawn from the uniform distribution on $\mathcal{T}_k^d$, that is, $\mathbb{P}(T_P = t) = 1/|\mathcal{T}_k^d|$ for all forests $t \in \mathcal{T}_k^d$. Recall that $\mathcal{T}_k^d$ is the set of labeled forests with d nodes and k edges.

(b) T_P is drawn from the uniform distribution on $\mathcal{F}^d$, that is, $\mathbb{P}(T_P = t) = 1/|\mathcal{F}^d|$ for all forests $t \in \mathcal{F}^d$. Recall that $\mathcal{F}^d$ is the set of labeled forests with d nodes.

This following result is inspired by Theorem 1 in Bresler et al. (2008). Note that an *estimator* or *algorithm* $\widehat{T}^d$ is simply a map from the set of samples $(\mathcal{X}^d)^n$ to a set of graphs (either $\mathcal{T}_k^d$ or $\mathcal{F}^d$). We emphasize that the following result is stated with the assumption that we are *averaging* over the random choice of the true graph T_P.

Theorem 7 (Lower Bounds on Sample Complexity) *Let* $\rho < 1$ *and* $r := |\mathcal{X}|$. *In case (a) above, if*

$$n < \rho \, \frac{(k-1)\log d}{d \log r}, \tag{17}$$

then $\mathbb{P}(\widehat{T}^d \neq T_P) \to 1$ *for any estimator* $\widehat{T}^d : (\mathcal{X}^d)^n \to \mathcal{T}_k^d$. *Alternatively, in case (b), if*

$$n < \rho \, \frac{\log d}{\log r}, \tag{18}$$

then $\mathbb{P}(\widehat{T}^d \neq T_P) \to 1$ *for any estimator* $\widehat{T}^d : (\mathcal{X}^d)^n \to \mathcal{F}^d$.

This result, a *strong converse*, states that $n = \Omega(\frac{k}{d} \log d)$ is *necessary* for any estimator with oracle knowledge of k to succeed. Thus, we need at least logarithmically many samples in d if the fraction k/d is kept constant as the graph size grows even if *k is known precisely* and does not have to be estimated. Interestingly, (17) says that if k is large, then we need more samples. This is because there are fewer forests with a small number of edges as compared to forests with a large number of edges. In contrast, the performance of CLThres degrades when k is small because it is more sensitive to the overestimation error. Moreover, if the estimator does not know k, then (18) says that $n = \Omega(\log d)$ is *necessary* for successful recovery. We conclude that the set of scaling requirements prescribed in Theorem 5 is almost optimal. In fact, if the true structure T_P is a tree, then Theorem 7 for CLThres says that the (achievability) scaling laws in Theorem 5 are indeed optimal (up to constant factors in the O and Ω-notation) since $n > (2\log(d-k))^{1+\zeta}$ in (15) is trivially satisfied. Note that if T_P is a tree, then the Chow-Liu ML procedure or CLThres results in the sample complexity $n = O(\log d)$ (see Theorem 5).

6. Risk Consistency

In this section, we develop results for risk consistency to study how fast the parameters of the estimated distribution converge to their true values. For this purpose, we define the *risk* of the estimated distribution P^* (with respect to the true probability model P) as

$$\mathcal{R}_n(P^*) := D(P\,||\,P^*) - D(P\,||\,\widetilde{P}), \tag{19}$$

9. The term *chance variable*, attributed to Gallager (2001), describes random quantities $Y : \Omega \to W$ that take on values in arbitrary alphabets W. In contrast, a random variable X maps the sample space Ω to the reals $\mathbb{R}$.

where $\widetilde{P}$ is the forest projection of P defined in (12). Note that the original probability model P does not need to be a forest-structured distribution in the definition of the risk. Indeed, if P is Markov on a forest, (19) reduces to $\mathcal{R}_n(P^*) = D(P\,||\,P^*)$ since the second term is zero. We quantify the rate of decay of the risk when the number of samples n tends to infinity. For $\delta > 0$, we define the event

$$\mathcal{C}_{n,\delta} := \left\{ \mathbf{x}^n \in (\mathcal{X}^d)^n : \frac{\mathcal{R}_n(P^*)}{d} > \delta \right\}. \tag{20}$$

That is, $\mathcal{C}_{n,\delta}$ is the event that the *average risk* $\mathcal{R}_n(P^*)/d$ exceeds some constant δ. We say that the estimator P^* (or an algorithm) is δ-*risk consistent* if the probability of $\mathcal{C}_{n,\delta}$ tends to zero as $n \to \infty$. Intuitively, achieving δ-risk consistency is easier than achieving structural consistency since the learned model P^* can be close to the true forest-projection $\widetilde{P}$ in the KL-divergence sense even if their structures differ.

In order to quantify the rate of decay of the risk in (19), we need to define some stochastic order notation. We say that a sequence of random variables $Y_n = O_p(g_n)$ (for some deterministic positive sequence $\{g_n\}$) if for every $\varepsilon > 0$, there exists a $B = B_\varepsilon > 0$ such that $\limsup_{n\to\infty} \Pr(|Y_n| > Bg_n) < \varepsilon$. Thus, $\Pr(|Y_n| > Bg_n) \geq \varepsilon$ holds for only finitely many n.

We say that a reconstruction algorithm has *risk consistency of order* (or *rate*) g_n if $\mathcal{R}_n(P^*) = O_p(g_n)$. The definition of the order of risk consistency involves the true model P. Intuitively, we expect that as $n \to \infty$, the estimated distribution P^* converges to the projection $\widetilde{P}$ so $\mathcal{R}_n(P^*) \to 0$ in probability.

6.1 Error Exponent for Risk Consistency

In this subsection, we consider a fixed distribution P and state consistency results in terms of the event $\mathcal{C}_{n,\delta}$. Consequently, the model size d and the number of edges k are fixed. This lends insight into deriving results for the order of the risk consistency and provides intuition for the high-dimensional scenario in Section 6.2.

Theorem 8 (Error Exponent for δ-Risk Consistency) *For* CLThres*, there exists a constant* $\delta_0 > 0$ *such that for all* $0 < \delta < \delta_0$,

$$\limsup_{n\to\infty} \frac{1}{n} \log P^n(\mathcal{C}_{n,\delta}) \leq -\delta. \tag{21}$$

The corresponding lower bound is

$$\liminf_{n\to\infty} \frac{1}{n} \log P^n(\mathcal{C}_{n,\delta}) \geq -\delta d. \tag{22}$$

The theorem states that if δ is sufficiently small, the decay rate of the probability of $\mathcal{C}_{n,\delta}$ is exponential, hence clearly CLThres is δ-risk consistent. Furthermore, the bounds on the error exponent associated to the event $\mathcal{C}_{n,\delta}$ are *independent* of the parameters of P and only depend on δ and the dimensionality d. Intuitively, (21) is true because if we want the risk of P^* to be at most δd, then each of the empirical pairwise marginals $\widehat{P}_{i,j}$ should be δ-close to the true pairwise marginal $\widetilde{P}_{i,j}$. Note also that for $\mathcal{C}_{n,\delta}$ to occur with high probability, the edge set does not need to be estimated correctly so there is no dependence on k.

6.2 The High-Dimensional Setting

We again consider the high-dimensional setting where the tuple of parameters (n, d_n, k_n) tend to infinity and we have a sequence of learning problems indexed by the number of data points n. We again assume that (13) and (14) hold and derive sufficient conditions under which the probability of the event $\mathcal{C}_{n,\delta}$ tends to zero for a sequence of d-variate distributions $\{P^{(d)} \in \mathcal{P}(\mathcal{X}^d)\}_{d\in\mathbb{N}}$. The proof of Theorem 8 leads immediately to the following corollary.

Corollary 9 (δ-Risk Consistency Scaling Law) *Let $\delta > 0$ be a sufficiently small constant and $a \in (0,\delta)$. If the number of variables in the sequence of models $\{P^{(d)}\}_{d\in\mathbb{N}}$ satisfies $d_n = o(\exp(an))$, then* CLThres *is δ-risk consistent for $\{P^{(d)}\}_{d\in\mathbb{N}}$.*

Interestingly, this sufficient condition on how number of variables d should scale with n for consistency is very similar to Theorem 5. In particular, if d is polynomial in n, then CLThres is both structurally consistent as well as δ-risk consistent. We now study the order of the risk consistency of CLThres as the model size d grows.

Theorem 10 (Order of Risk Consistency) *The risk of the sequence of estimated distributions $\{(P^{(d)})^*\}_{d\in\mathbb{N}}$ with respect to $\{P^{(d)}\}_{d\in\mathbb{N}}$ satisfies*

$$\mathcal{R}_n((P^{(d)})^*) = O_p\left(\frac{d\log d}{n^{1-\gamma}}\right), \tag{23}$$

for every $\gamma > 0$, that is, the risk consistency for CLThres *is of order $(d\log d)/n^{1-\gamma}$.*

Note that since this result is stated for the high-dimensional case, $d = d_n$ is a sequence in n but the dependence on n is suppressed for notational simplicity in (23). This result implies that if $d = o(n^{1-2\gamma})$ then CLThres is risk consistent, that is, $\mathcal{R}_n((P^{(d)})^*) \to 0$ in probability. Note that this result is not the same as the conclusion of Corollary 9 which refers to the probability that the average risk is greater than a fixed constant δ. Also, the order of convergence given in (23) does not depend on the true number of edges k. This is a consequence of the result in (21) where the upper bound on the exponent associated to the event $\mathcal{C}_{n,\delta}$ is independent of the parameters of P.

The order of the risk, or equivalently the rate of convergence of the estimated distribution to the forest projection, is almost linear in the number of variables d and inversely proportional to n. We provide three intuitive reasons to explain why this is plausible: (i) the dimension of the sufficient statistics in a tree-structured graphical model is of order $O(d)$, (ii) the ML estimator of the natural parameters of an exponential family converge to their true values at the rate of $O_p(n^{-1/2})$ (Serfling, 1980, Sec. 4.2.2), and (iii) locally, the KL-divergence behaves like the square of a weighted Euclidean norm of the natural parameters (Cover and Thomas, 2006, Equation (11.320)).

We now compare Theorem 10 to the corresponding results in Liu et al. (2011). In these recent papers, it was shown that by modeling the bivariate densities $\widehat{P}_{i,j}$ as functions from a Hölder class with exponent $\alpha > 0$ and using a reconstruction algorithm based on validation on a held-out data set, the risk decays at a rate[10] of $\widetilde{O}_p(dn^{-\alpha/(1+2\alpha)})$, which is slower than the order of risk consistency in (23). This is due to the need to compute the bivariate densities via kernel density estimation. Furthermore, we model the pairwise joint distributions as discrete probability mass functions and not continuous probability density functions, hence there is no dependence on Hölder exponents.

10. The $\widetilde{O}_p(\cdot)$ notation suppresses the dependence on factors involving logarithms.

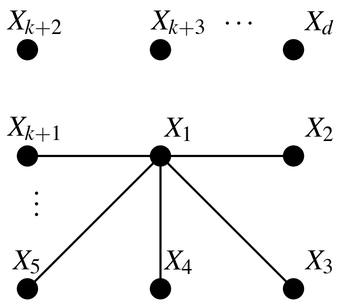


Figure 2: The forest-structured distribution Markov on d nodes and k edges. Variables $X_{k+1},\dots,X_d$ are not connected to the main star graph.

7. Numerical Results

In this section, we perform numerical simulations on synthetic and real data sets to study the effect of a finite number of samples on the probability of the event $\mathcal{A}_n$ defined in (4). Recall that this is the error event associated to an incorrect learned structure.

7.1 Synthetic Data Sets

In order to compare our estimate to the ground truth graph, we learn the structure of distributions that are Markov on the forest shown in Figure 2. Thus, a subgraph (nodes $1,\dots,k+1$) is a (connected) star while nodes $k+2,\dots,d-1$ are not connected to the star. Each random variable X_j takes on values from a binary alphabet $\mathcal{X}=\{0,1\}$. Furthermore, $P_j(x_j)=0.5$ for $x_j=0,1$ and all $j\in V$. The conditional distributions are governed by the "binary symmetric channel":

$$P_{j|1}(x_j|x_1)=\begin{cases} 0.7 & x_j=x_1 \\ 0.3 & x_j\neq x_1 \end{cases}$$

for $j=2,\dots,k+1$. We further assume that the regularization sequence is given by $\varepsilon_n := n^{-\beta}$ for some $\beta\in(0,1)$. Recall that this sequence satisfies the conditions in (6). We will vary β in our experiments to observe its effect on the overestimation and underestimation errors.

In Figure 3, we show the simulated error probability as a function of the sample size n for a $d=101$ node graph (as in Figure 2) with $k=50$ edges. The error probability is estimated based on 30,000 independent runs of CLThres (over different data sets $\mathbf{x}^n$). We observe that the error probability is minimized when $\beta\approx 0.625$. Figure 4 show the simulated overestimation and underestimation errors for this experiment. We see that as $\beta\to 0$, the overestimation (resp., underestimation) error is likely to be small (resp., large) because the regularization sequence ε_n is large. When the number of samples is relatively small as in this experiment, both types of errors contribute significantly to the overall error probability. When $\beta\approx 0.625$, we have the best tradeoff between overestimation and underestimation for this particular experimental setting.

Even though we mentioned that β in (10) should be chosen to be close to zero so that the error probability of structure learning decays as rapidly as possible, this example demonstrates that when given a finite number of samples, β should be chosen to balance the overestimation and underestimation errors. This does not violate Theorem 3 since Theorem 3 is an asymptotic result and refers to the typical way an error occurs in the limit as $n\to\infty$. Indeed, when the number of

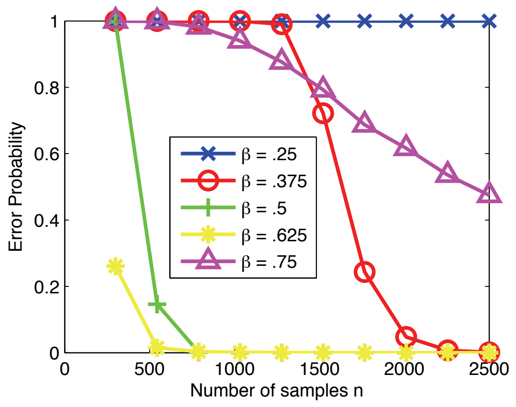


Figure 3: The error probability of structure learning for $\beta \in (0,1)$.

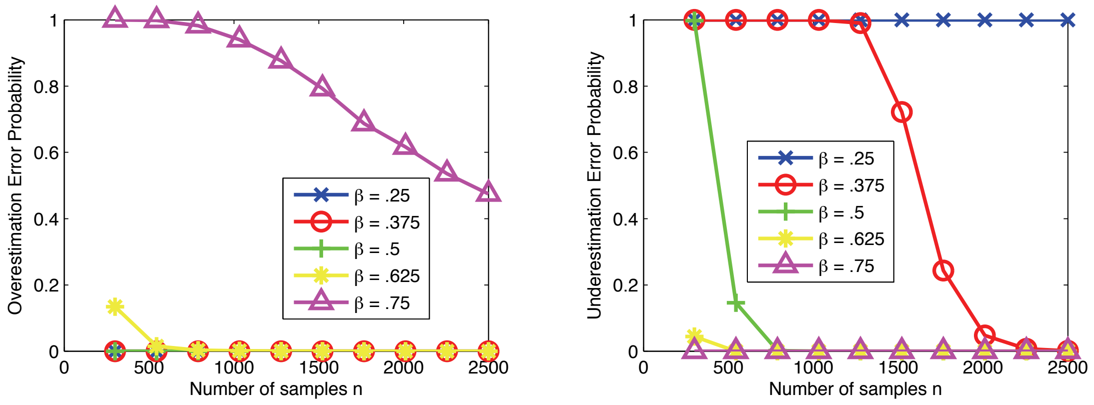


Figure 4: The overestimation and underestimation errors for $\beta \in (0,1)$.

samples is very large, it is shown that the overestimation error dominates the overall probability of error and so one should choose β to be close to zero. The question of how best to select optimal β when given only a finite number of samples appears to be a challenging one. We use cross-validation as a proxy to select this parameter for the real-world data sets in the next section.

In Figure 5, we fix the value of β at 0.625 and plot the KL-divergence $D(P\,||\,P^*)$ as a function of the number of samples. This is done for a forest-structured distribution P whose graph is shown in Figure 2 and with $d = 21$ nodes and $k = 10$ edges. The mean, minimum and maximum KL-divergences are computed based on 50 independent runs of CLThres. We see that $\log D(P\,||\,P^*)$ decays linearly. Furthermore, the slope of the mean curve is approximately -1, which is in agreement with (23). This experiment shows that if we want to reduce the KL-divergence between the estimated and true models by a constant factor $A > 0$, we need to increase the number of samples by roughly the same factor A.

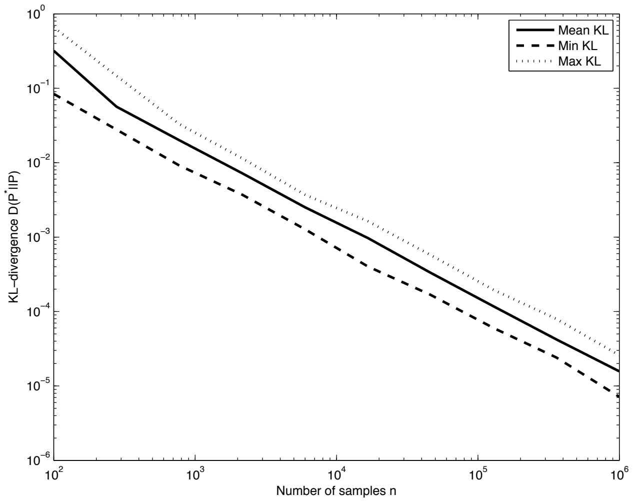


Figure 5: Mean, minimum and maximum (across 50 different runs) of the KL-divergence between the estimated model P^* and the true model P for a $d = 21$ node graph with $k = 10$ edges.

7.2 Real Data Sets

We now demonstrate how well forests-structured distributions can model two real data sets[11] which are obtained from the UCI Machine Learning Repository (Newman et al., 1998). The first data set we used is known as the SPECT Heart data set, which describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images on normal and abnormal patients. The data set contains $d = 22$ binary variables and $n = 80$ training samples. There are also 183 test samples. We learned a forest-structured distributions using the 80 training samples for different $\beta \in (0,1)$ and subsequently computed the log-likelihood of both the training and test samples. The results are displayed in Figure 6. We observe that, as expected, the log-likelihood of the training samples increases monotonically with β. This is because there are more edges in the model when β is large improving the modeling ability. However, we observe that there is overfitting when β is large as evidenced by the decrease in the log-likelihood of the 183 test samples. The optimal value of β in terms of the log-likelihood for this data set is ≈ 0.25, but surprisingly an approximation with an empty graph[12] also yields a high log-likelihood score on the test samples. This implies that according to the available data, the variables are nearly independent. The forest graph for $\beta = 0.25$ is shown in Figure 7(a) and is very sparse.

The second data set we used is the Statlog Heart data set containing physiological measurements of subjects with and without heart disease. There are 270 subjects and $d = 13$ discrete and continuous attributes, such as gender and resting blood pressure. We quantized the continuous attributes into two bins. Those measurements that are above the mean are encoded as 1 and those below the mean as 0. Since the raw data set is not partitioned into training and test sets, we learned forest-structured models based on a randomly chosen set of $n = 230$ training samples and then computed

11. These data sets are typically employed for binary classification but we use them for modeling purposes.
12. When $\beta = 0$ we have an empty graph because all empirical mutual information quantities in this experiment are smaller than 1.

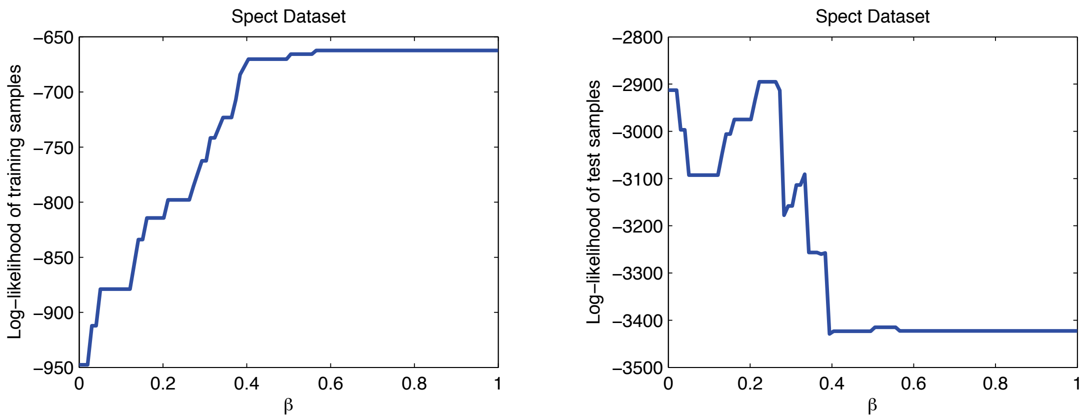


Figure 6: Log-likelihood scores on the SPECT data set

the log-likelihood of these training and 40 remaining test samples. We then chose an additional 49 randomly partitioned training and test sets and performed the same learning task and computation of log-likelihood scores. The mean of the log-likelihood scores over these 50 runs is shown in Figure 8. We observe that the log-likelihood on the test set is maximized at $\beta \approx 0.53$ and the tree approximation ($\beta \approx 1$) also yields a high likelihood score. The forest learned when $\beta = 0.53$ is shown in Figure 7(b). Observe that two nodes (ECG and Cholesterol) are disconnected from the main graph because their mutual information values with other variables are below the threshold. In contrast, HeartDisease, the label for this data set, has the highest degree, that is, it influences and is influenced by many other covariates. The strengths of the interactions between HeartDisease and its neighbors are also strong as evidenced by the bold edges.

From these experiments, we observe that some data sets can be modeled well as proper forests with very few edges while others are better modeled as distributions that are almost tree-structured (see Figure 7). Also, we need to choose β carefully to balance between data fidelity and overfitting. In contrast, our asymptotic result in Theorem 3 says that ε_n should be chosen according to (6) so that we have structural consistency. When the number of data points n is large, β in (10) should be chosen to be small to ensure that the learned edge set is equal to the true one (assuming the underlying model is a forest) with high probability as the overestimation error dominates.

8. Conclusion

In this paper, we proposed an efficient algorithm CLThres for learning the parameters and the structure of forest-structured graphical models. We showed that the asymptotic error rates associated to structure learning are nearly optimal. We also provided the rate at which the error probability of structure learning tends to zero and the order of the risk consistency. One natural question that arises from our analyses is whether β in (10) can be selected automatically in the finite-sample regime. There are many other open problems that could possibly leverage on the proof techniques employed here. For example, we are currently interested to analyze the learning of general graphical models using similar thresholding-like techniques on the empirical correlation coefficients. The analyses could potentially leverage on the use of the method of types. We are currently exploring this promising line of research.

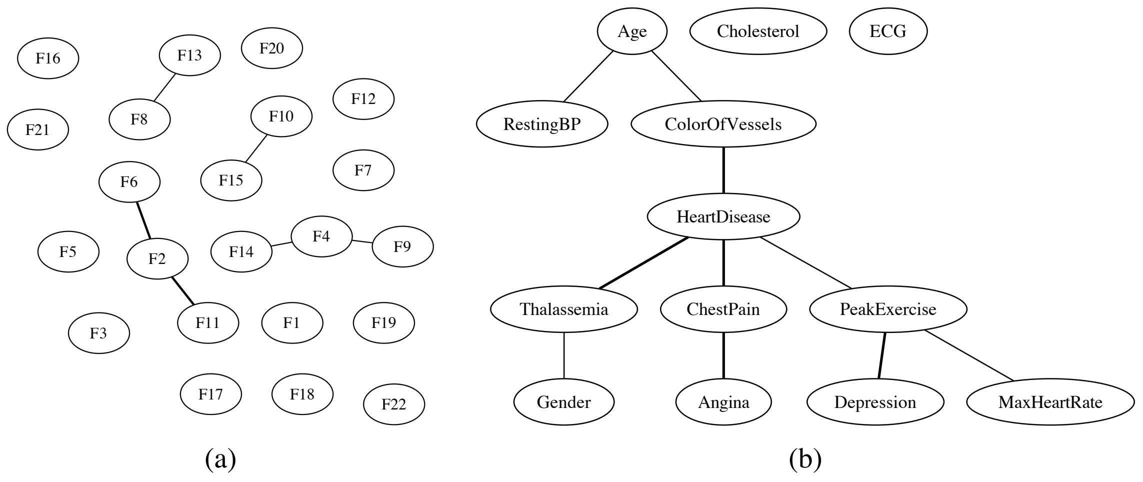


Figure 7: Learned forest graph of the (a) SPECT data set for $\beta = 0.25$ and (b) HEART data set for $\beta = 0.53$. Bold edges denote higher mutual information values. The features names are not provided for the SPECT data set.

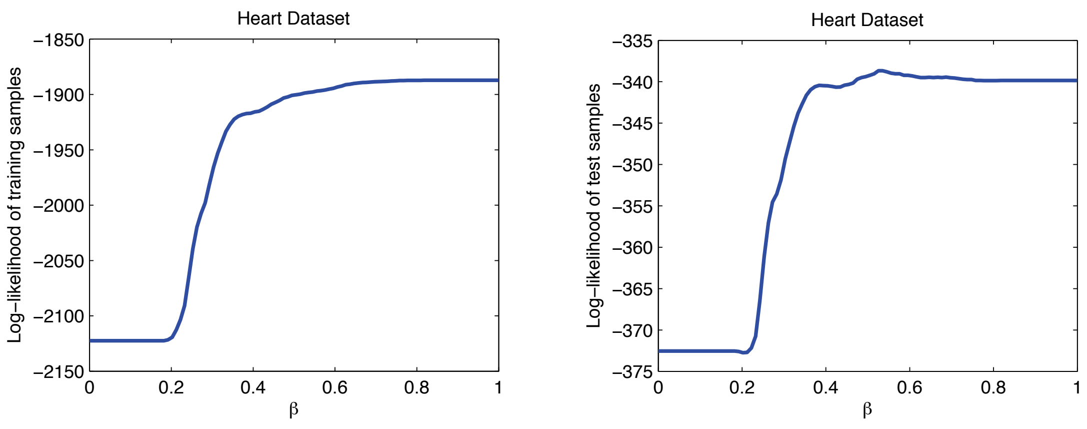


Figure 8: Log-likelihood scores on the HEART data set

Acknowledgments

This work was supported by a AFOSR funded through Grant FA9559-08-1-1080, a MURI funded through ARO Grant W911NF-06-1-0076 and a MURI funded through AFOSR Grant FA9550-06-1-0324. V. Tan is also funded by A*STAR, Singapore. The authors would like to thank Sanjoy Mitter, Lav Varshney, Matt Johnson and James Saunderson for discussions. The authors would also like to thank Rui Wu (UIUC) for pointing out an error in the proof of Theorem 3.

Appendix A. Proof of Proposition 2

Proof (*Sketch*) The proof of this result hinges on the fact that both the overestimation and underestimation errors decay to zero exponentially fast when the threshold is chosen to be $I_{\min}/2$. This threshold is able to differentiate between true edges (with MI larger than $I_{\min}$) from non-edges (with MI smaller than $I_{\min}$) with high probability for n sufficiently large. The error for learning the top k edges of the forest also decays exponentially fast (Tan et al., 2011). Thus, (5) holds. The full details of the proof follow in a straightforward manner from Appendix B which we present next. ■

Appendix B. Proof of Theorem 3

Define the event $\mathcal{B}_n := \{\widehat{E}_k \neq E_P\}$, where $\widehat{E}_k = \{\widehat{e}_1, \ldots, \widehat{e}_k\}$ is the set of top k edges (see Step 3 of CLThres for notation). This is the Chow-Liu error as mentioned in Section 4.3. Let $\mathcal{B}_n^c$ denote the complement of $\mathcal{B}_n$. Note that in $\mathcal{B}_n^c$, the estimated edge set depends on k, the true model order, which is *a-priori* unknown to the learner. Further define the constant

$$K_P := \lim_{n\to\infty} -\frac{1}{n} \log P^n(\mathcal{B}_n). \tag{24}$$

In other words, K_P is the error exponent for learning the forest structure incorrectly assuming the true model order k is known and Chow-Liu terminates after the addition of exactly k edges in the MWST procedure (Kruskal, 1956). The existence of the limit in (24) and the positivity of K_P follow from the main results in Tan et al. (2011).

We first state a result which relies on the Gallager-Fano bound (Fano, 1961, pp. 24). The proof will be provided at the end of this appendix.

Lemma 11 (Reduction to Model Order Estimation) *For every $\eta \in (0, K_P)$, there exists a $N \in \mathbb{N}$ sufficiently large such that for every $n > N$, the error probability $P^n(\mathcal{A}_n)$ satisfies*

$$(1-\eta)P^n(\widehat{k}_n \neq k|\mathcal{B}_n^c) \leq P^n(\mathcal{A}_n) \tag{25}$$

$$\leq P^n(\widehat{k}_n \neq k|\mathcal{B}_n^c) + 2\exp(-n(K_P - \eta)). \tag{26}$$

Proof (*of Theorem 3*) We will prove (i) the upper bound in (8) (ii) the lower bound in (7) and (iii) the exponential rate of decay in the case of trees (9).

B.1 Proof of Upper Bound in Theorem 3

We now bound the error probability $P^n(\widehat{k}_n \neq k|\mathcal{B}_n^c)$ in (26). Using the union bound,

$$P^n(\widehat{k}_n \neq k|\mathcal{B}_n^c) \leq P^n(\widehat{k}_n > k|\mathcal{B}_n^c) + P^n(\widehat{k}_n < k|\mathcal{B}_n^c). \tag{27}$$

The first and second terms are known as the *overestimation* and *underestimation* errors respectively. We will show that the underestimation error decays exponentially fast. The overestimation error decays only subexponentially fast and so its rate of decay dominates the overall rate of decay of the error probability for structure learning.

B.1.1 UNDERESTIMATION ERROR

We now bound these terms staring with the underestimation error. By the union bound,

$$\begin{aligned} P^n(\widehat{k}_n < k|\mathcal{B}_n^c) &\leq (k-1)\max_{1\leq j\leq k-1} P^n(\widehat{k}_n = j|\mathcal{B}_n^c) \\ &= (k-1)P^n(\widehat{k}_n = k-1|\mathcal{B}_n^c), \end{aligned} \tag{28}$$

where (28) follows because $P^n(\widehat{k}_n = j|\mathcal{B}_n^c)$ is maximized when $j = k-1$. This is because if, to the contrary, $P^n(\widehat{k}_n = j|\mathcal{B}_n^c)$ were to be maximized at some other $j \leq k-2$, then there exists at least two edges, call them $e_1, e_2 \in E_P$ such that events $\mathcal{E}_1 := \{I(\widehat{P}_{e_1}) \leq \varepsilon_n\}$ and $\mathcal{E}_2 := \{I(\widehat{P}_{e_2}) \leq \varepsilon_n\}$ occur. The probability of this joint event is smaller than the individual probabilities, that is, $P^n(\mathcal{E}_1 \cap \mathcal{E}_2) \leq \min\{P^n(\mathcal{E}_1), P^n(\mathcal{E}_2)\}$. This is a contradiction.

By the rule for choosing $\widehat{k}_n$ in (3), we have the upper bound

$$P^n(\widehat{k}_n = k-1|\mathcal{B}_n^c) = P^n(\exists e \in E_P \text{ s.t. } I(\widehat{P}_e) \leq \varepsilon_n) \leq k \max_{e\in E_P} P^n(I(\widehat{P}_e) \leq \varepsilon_n), \tag{29}$$

where the inequality follows from the union bound. Now, note that if $e \in E_P$, then $I(P_e) > \varepsilon_n$ for n sufficiently large (since $\varepsilon_n \to 0$). Thus, by Sanov's theorem (Cover and Thomas, 2006, Ch. 11), $P^n(I(\widehat{P}_e) \leq \varepsilon_n)$ can be upper bounded as

$$P^n(I(\widehat{P}_e) \leq \varepsilon_n) \leq (n+1)^{r^2} \exp\left(-n \min_{Q\in\mathcal{P}(\mathcal{X}^2)} \{D(Q\,||\,P_e) : I(Q) \leq \varepsilon_n\}\right). \tag{30}$$

Define the good rate function (Dembo and Zeitouni, 1998) in (30) to be $L : \mathcal{P}(\mathcal{X}^2) \times [0,\infty) \to [0,\infty)$, which is given by

$$L(P_e;a) := \min_{Q\in\mathcal{P}(\mathcal{X}^2)} \{D(Q\,||\,P_e) : I(Q) \leq a\}. \tag{31}$$

Clearly, $L(P_e;a)$ is continuous in a. Furthermore it is monotonically decreasing in a for fixed P_e. Thus by using the continuity of $L(P_e;\cdot)$ we can assert: To every $\eta > 0$, there exists a $N \in \mathbb{N}$ such that for all $n > N$ we have $L(P_e;\varepsilon_n) > L(P_e;0) - \eta$. As such, we can further upper bound the error probability in (30) as

$$P^n(I(\widehat{P}_e) \leq \varepsilon_n) \leq (n+1)^{r^2} \exp\left(-n(L(P_e;0) - \eta)\right). \tag{32}$$

By using the fact that $I_{\min} > 0$, the exponent $L(P_e;0) > 0$ and thus, we can put the pieces in (28), (29) and (32) together to show that the underestimation error is upper bounded as

$$P^n(\widehat{k}_n < k|\mathcal{B}_n^c) \leq k(k-1)(n+1)^{r^2} \exp\left(-n \min_{e\in E_P}(L(P_e;0) - \eta)\right). \tag{33}$$

Hence, if k is constant, the underestimation error $P^n(\widehat{k}_n < k|\mathcal{B}_n^c)$ decays to zero exponentially fast as $n \to \infty$, that is, the normalized logarithm of the underestimation error can be bounded as

$$\limsup_{n\to\infty} \frac{1}{n} \log P^n(\widehat{k}_n < k|\mathcal{B}_n^c) \leq -\min_{e\in E_P}(L(P_e;0) - \eta).$$

The above statement is now independent of n. Hence, we can take the limit as $\eta \to 0$ to conclude that:

$$\limsup_{n\to\infty} \frac{1}{n} \log P^n(\widehat{k}_n < k | \mathcal{B}_n^c) \leq -L_P. \tag{34}$$

The exponent $L_P := \min_{e\in E_P} L(P_e;0)$ is positive because we assumed that the model is minimal and so $I_{\min} > 0$, which ensures the positivity of the rate function $L(P_e;0)$ for each true edge $e \in E_P$.

B.1.2 OVERESTIMATION ERROR

Bounding the overestimation error is harder. It follows by first applying the union bound:

$$\begin{aligned} P^n(\widehat{k}_n > k|\mathcal{B}_n^c) &\leq (d-k-1) \max_{k+1\leq j\leq d-1} P^n(\widehat{k}_n = j|\mathcal{B}_n^c) \\ &= (d-k-1) P^n(\widehat{k}_n = k+1|\mathcal{B}_n^c), \end{aligned} \tag{35}$$

where (35) follows because $P^n(\widehat{k}_n = j|\mathcal{B}_n^c)$ is maximized when $j = k+1$ (by the same argument as for the underestimation error). Apply the union bound again, we have

$$P^n(\widehat{k}_n = k+1|\mathcal{B}_n^c) \leq (d-k-1) \max_{e\in V\times V: I(P_e)=0} P^n(I(\widehat{P}_e) \geq \varepsilon_n). \tag{36}$$

From (36), it suffices to bound $P^n(I(\widehat{P}_e) \geq \varepsilon_n)$ for any pair of independent random variables (X_i, X_j) and $e = (i,j)$. We proceed by applying the upper bound in Sanov's theorem (Cover and Thomas, 2006, Ch. 11) to $P^n(I(\widehat{P}_e) \geq \varepsilon_n)$ which yields

$$P^n(I(\widehat{P}_e) \geq \varepsilon_n) \leq (n+1)^{r^2} \exp\left(-n \min_{Q\in\mathcal{P}(\mathcal{X}^2)} \{D(Q\,||\,P_e) : I(Q) \geq \varepsilon_n\}\right), \tag{37}$$

for all $n \in \mathbb{N}$. Our task now is to lower bound the good rate function in (37), which we denote as $M : \mathcal{P}(\mathcal{X}^2) \times [0,\infty) \to [0,\infty)$:

$$M(P_e;b) := \min_{Q\in\mathcal{P}(\mathcal{X}^2)} \{D(Q\,||\,P_e) : I(Q) \geq b\}. \tag{38}$$

Note that $M(P_e;b)$ is monotonically increasing and continuous in b for fixed P_e. Because the sequence $\{\varepsilon_n\}_{n\in\mathbb{N}}$ tends to zero, when n is sufficiently large, ε_n is arbitrarily small and we are in the so-called *very-noisy learning regime* (Borade and Zheng, 2008; Tan et al., 2011), where the optimizer to (38), denoted as Q_n^*, is very close to P_e. See Figure 9.

Thus, when n is large, the KL-divergence and mutual information can be approximated as

$$D(Q_n^*\,||\,P_e) = \frac{1}{2}\mathbf{v}^T \mathbf{\Pi}_e \mathbf{v} + o(\|\mathbf{v}\|^2), \tag{39}$$

$$I(Q_n^*) = \frac{1}{2}\mathbf{v}^T \mathbf{H}_e \mathbf{v} + o(\|\mathbf{v}\|^2), \tag{40}$$

where[13] $\mathbf{v} := \mathrm{vec}(Q_n^*) - \mathrm{vec}(P_e) \in \mathbb{R}^{r^2}$. The $r^2 \times r^2$ matrices $\mathbf{\Pi}_e$ and $\mathbf{H}_e$ are defined as

$$\mathbf{\Pi}_e := \mathrm{diag}(1/\mathrm{vec}(P_e)), \tag{41}$$

$$\mathbf{H}_e := \nabla^2_{\mathrm{vec}(Q)} I(\mathrm{vec}(Q))\big|_{Q=P_e}. \tag{42}$$

13. The operator $\mathrm{vec}(\mathbf{C})$ vectorizes a matrix in a column oriented way. Thus, if $\mathbf{C} \in \mathbb{R}^{l\times l}$, $\mathrm{vec}(\mathbf{C})$ is a length-l^2 vector with the columns of $\mathbf{C}$ stacked one on top of another ($\mathbf{C}(:)$ in Matlab).

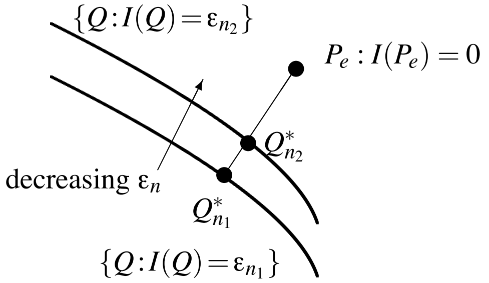

$\{Q : I(Q) = \varepsilon_{n_2}\}$
$P_e : I(P_e) = 0$
$Q^*_{n_2}$
decreasing ε_n
$Q^*_{n_1}$
$\{Q : I(Q) = \varepsilon_{n_1}\}$

Figure 9: As $\varepsilon_n \to 0$, the projection of P_e onto the constraint set $\{Q : I(Q) \geq \varepsilon_n\}$, denoted Q^*_n (the optimizer in (38)), approaches P_e. The approximations in (39) and (40) become increasingly accurate as ε_n tends to zero. In the figure, $n_2 > n_1$ and $\varepsilon_{n_1} > \varepsilon_{n_2}$ and the curves are the (sub-)manifold of distributions such that the mutual information is constant, that is, the mutual information level sets.

In other words, $\mathbf{\Pi}_e$ is the diagonal matrix that contains the reciprocal of the elements of $\mathrm{vec}(P_e)$ on its diagonal. $\mathbf{H}_e$ is the Hessian[14] of $I(\mathrm{vec}(Q^*_n))$, viewed as a function of $\mathrm{vec}(Q^*_n)$ and evaluated at P_e. As such, the exponent for overestimation in (38) can be approximated by a *quadratically constrained quadratic program* (QCQP), where $\mathbf{z} := \mathrm{vec}(Q) - \mathrm{vec}(P_e)$:

$$\widetilde{M}(P_e;\varepsilon_n) = \min_{\mathbf{z}\in\mathbb{R}^{r^2}} \frac{1}{2}\mathbf{z}^T\mathbf{\Pi}_e\mathbf{z},$$
$$\text{subject to} \quad \frac{1}{2}\mathbf{z}^T\mathbf{H}_e\mathbf{z} \geq \varepsilon_n, \quad \mathbf{z}^T\mathbf{1} = 0. \tag{43}$$

Note that the constraint $\mathbf{z}^T\mathbf{1} = 0$ does not necessarily ensure that Q is a probability distribution so $\widetilde{M}(P_e;\varepsilon_n)$ is an approximate lower bound to the true rate function $M(P_e;\varepsilon_n)$, defined in (38). We now argue that the approximate rate function $\widetilde{M}$ in (43), can be lower bounded by a quantity that is proportional to ε_n. To show this, we resort to Lagrangian duality (Bertsekas, 1999, Ch. 5). It can easily be shown that the *Lagrangian dual* corresponding to the primal in (43) is

$$g(P_e;\varepsilon_n) := \varepsilon_n \max_{\mu\geq 0}\{\mu : \mathbf{\Pi}_e \succeq \mu\mathbf{H}_e\}. \tag{44}$$

We see from (44) that $g(P_e;\varepsilon_n)$ is proportional to ε_n. By weak duality (Bertsekas, 1999, Proposition 5.1.3), any dual feasible solution provides a lower bound to the primal, that is,

$$g(P_e;\varepsilon_n) \leq \widetilde{M}(P_e;\varepsilon_n). \tag{45}$$

Note that strong duality (equality in (45)) does not hold in general due in part to the non-convex constraint set in (43). Interestingly, our manipulations lead lower bounding $\widetilde{M}$ by (44), which is a (convex) semidefinite program (Vandenberghe and Boyd, 1996).

Now observe that the approximations in (39) and (40) are accurate in the limit of large n because the optimizing distribution Q^*_n becomes increasingly close to P_e. By continuity of the optimization

14. The first two terms in the Taylor expansion of the mutual information $I(\mathrm{vec}(Q^*_n))$ in (40) vanish because (i) $I(P_e) = 0$ and (ii) $(\mathrm{vec}(Q^*_n) - \mathrm{vec}(P_e))^T\nabla_{\mathrm{vec}(Q)}I(\mathrm{vec}(P_e)) = 0$. Indeed, if we expand $I(\mathrm{vec}(Q))$ around a product distribution, the constant and linear terms vanish (Borade and Zheng, 2008). Note that $\mathbf{H}_e$ in (42) is an indefinite matrix because $I(\mathrm{vec}(Q))$ is not convex.

problems in (perturbations of) the objective and the constraints, $\widetilde{M}(P_e;\varepsilon_n)$ and $M(P_e;\varepsilon_n)$ are close when n is large, that is,

$$\lim_{n\to\infty}\frac{\widetilde{M}(P_e;\varepsilon_n)}{M(P_e;\varepsilon_n)}=1. \tag{46}$$

This can be seen from (39) in which the ratio of the KL-divergence to its approximation $\mathbf{v}^T\mathbf{\Pi}_e\mathbf{v}/2$ is unity in the limit as $\|\mathbf{v}\|\to 0$. The same holds true for the ratio of the mutual information to its approximation $\mathbf{v}^T\mathbf{H}_e\mathbf{v}/2$ in (40). By applying the continuity statement in (46) to the upper bound in (37), we can conclude that for every $\eta>0$, there exists a $N\in\mathbb{N}$ such that

$$P^n(I(\widehat{P}_e)\geq\varepsilon_n)\leq(n+1)^{r^2}\exp\left(-n\widetilde{M}(P_e;\varepsilon_n)(1-\eta)\right),$$

for all $n>N$. Define the constant

$$c_P:=\min_{e\in V\times V:I(P_e)=0}\ \max_{\mu\geq 0}\ \{\mu:\mathbf{\Pi}_e\succeq\mu\mathbf{H}_e\}. \tag{47}$$

By (44), (45) and the definition of c_P in (47),

$$P^n(I(\widehat{P}_e)\geq\varepsilon_n)\leq(n+1)^{r^2}\exp\left(-n\varepsilon_n c_P(1-\eta)\right). \tag{48}$$

Putting (35), (36) and (48) together, we see that the overestimation error

$$P^n(\widehat{k}_n>k|\mathcal{B}_n^c)\leq(d-k-1)^2(n+1)^{r^2}\exp\left(-n\varepsilon_n c_P(1-\eta)\right). \tag{49}$$

Note that the above probability tends to zero by the assumption that $n\varepsilon_n/\log n\to\infty$ in (6). Thus, we have consistency overall (since the underestimation, Chow-Liu and now the overestimation errors all tend to zero). Thus, by taking the normalized logarithm (normalized by $n\varepsilon_n$), the $\limsup$ in n (keeping in mind that d and k are constant), we conclude that

$$\limsup_{n\to\infty}\frac{1}{n\varepsilon_n}\log P^n(\widehat{k}_n>k|\mathcal{B}_n^c)\leq -c_P(1-\eta). \tag{50}$$

Now by take $\eta\to 0$, it remains to prove that $c_P=1$ for all P. For this purpose, it suffices to show that the optimal solution to the optimization problem in (44), denoted μ^*, is equal to one for all $\mathbf{\Pi}_e$ and $\mathbf{H}_e$. Note that μ^* can be expressed in terms of eigenvalues:

$$\mu^*=\left(\max\left\{\mathrm{eig}(\mathbf{\Pi}_e^{-1/2}\mathbf{H}_e\mathbf{\Pi}_e^{-1/2})\right\}\right)^{-1}, \tag{51}$$

where $\mathrm{eig}(\mathbf{A})$ denotes the set of real eigenvalues of the symmetric matrix $\mathbf{A}$. By using the definitions of $\mathbf{\Pi}_e$ and $\mathbf{H}_e$ in (41) and (42) respectively, we can verify that the matrix $\mathbf{I}-\mathbf{\Pi}_e^{-1/2}\mathbf{H}_e\mathbf{\Pi}_e^{-1/2}$ is positive semidefinite with an eigenvalue at zero. This proves that the largest eigenvalue of $\mathbf{\Pi}_e^{-1/2}\mathbf{H}_e\mathbf{\Pi}_e^{-1/2}$ is one and hence from (51), $\mu^*=1$. The proof of the upper bound in (8) is completed by combining the estimates in (26), (34) and (50).

B.2 Proof of Lower Bound in Theorem 3

The key idea is to bound the overestimation error using a modification of the lower bound in Sanov's theorem. Denote the set of types supported on a finite set $\mathcal{Y}$ with denominator n as $\mathcal{P}_n(\mathcal{Y})$ and the *type class* of a distribution $Q \in \mathcal{P}_n(\mathcal{Y})$ as

$$\mathsf{T}_n(Q) := \{y^n \in \mathcal{Y}^n : \widehat{P}(\cdot\,; y^n) = Q(\cdot)\},$$

where $\widehat{P}(\cdot\,; y^n)$ is the empirical distribution of the sequence $y^n = (y_1, \dots, y_n)$. The following bounds on the type class are well known (Cover and Thomas, 2006, Ch. 11).

Lemma 12 (Probability of Type Class) *For any $Q \in \mathcal{P}_n(\mathcal{Y})$ and any distribution P, the probability of the type class $\mathsf{T}_n(Q)$ under P^n satisfies:*

$$(n+1)^{-|\mathcal{Y}|} \exp(-nD(Q\,||\,P)) \le P^n(\mathsf{T}_n(Q)) \le \exp(-nD(Q\,||\,P)). \tag{52}$$

To prove the lower bound in (7), assume that $k < d-1$ and note that the error probability $P^n(\widehat{k}_n \ne k|\mathcal{B}_n^c)$ can be lower bounded by $P^n(I(\widehat{P}_e) \ge \varepsilon_n)$ for any node pair e such that $I(P_e) = 0$. We seek to lower bound the latter probability by appealing to (52). Now choose a sequence of distributions $Q^{(n)} \in \{Q \in \mathcal{P}_n(\mathcal{X}^2) : I(Q) \ge \varepsilon_n\}$ such that

$$\lim_{n\to\infty} \left| M(P_e; \varepsilon_n) - D(Q^{(n)}\,||\,P_e) \right| = 0.$$

This is possible because the set of types is dense in the probability simplex (Dembo and Zeitouni, 1998, Lemma 2.1.2(b)). Thus,

$$\begin{aligned} P^n(I(\widehat{P}_e) \ge \varepsilon_n) &= \sum_{Q \in \mathcal{P}_n(\mathcal{X}^2): I(Q) \ge \varepsilon_n} P^n(\mathsf{T}_n(Q)) \\ &\ge P^n(\mathsf{T}_n(Q^{(n)})) \\ &\ge (n+1)^{-r^2} \exp(-nD(Q^{(n)}\,||\,P_e)), \end{aligned} \tag{53}$$

where (53) follows from the lower bound in (52). Note from (46) that the following convergence holds: $|\widetilde{M}(P_e; \varepsilon_n) - M(P_e; \varepsilon_n)| \to 0$. Using this and the fact that if $|a_n - b_n| \to 0$ and $|b_n - c_n| \to 0$ then, $|a_n - c_n| \to 0$ (triangle inequality), we also have

$$\lim_{n\to\infty} \left| \widetilde{M}(P_e; \varepsilon_n) - D(Q^{(n)}\,||\,P_e) \right| = 0.$$

Hence, continuing the chain in (53), for any $\eta > 0$, there exists a $N \in \mathbb{N}$ such that for all $n > N$,

$$P^n(I(\widehat{P}_e) \ge \varepsilon_n) \ge (n+1)^{-r^2} \exp(-n(\widetilde{M}(P_e; \varepsilon_n) + \eta)). \tag{54}$$

Note that an upper bound for $\widetilde{M}(P_e; \varepsilon_n)$ in (43) is simply given by the objective evaluated at any feasible point. In fact, by manipulating (43), we see that the upper bound is also proportional to ε_n, that is,

$$\widetilde{M}(P_e; \varepsilon_n) \le C_{P_e} \varepsilon_n,$$

where $C_{P_e} \in (0,\infty)$ is some constant[15] that depends on the matrices $\mathbf{\Pi}_e$ and $\mathbf{H}_e$. Define $C_P := \max_{e \in V\times V : I(P_e)=0} C_{P_e}$. Continuing the lower bound in (54), we obtain

$$P^n(I(\widehat{P}_e) \geq \varepsilon_n) \geq (n+1)^{-r^2} \exp(-n\varepsilon_n(C_P+\eta)),$$

for n sufficiently large. Now take the normalized logarithm and the lim inf to conclude that

$$\liminf_{n\to\infty} \frac{1}{n\varepsilon_n} \log P^n(\widehat{k}_n \neq k | \mathcal{B}_n^c) \geq -(C_P+\eta). \tag{55}$$

Substitute (55) into the lower bound in (25). Now the resulting inequality is independent of n and we can take $\eta \to 0$ to complete the proof of the lower bound in Theorem 3.

B.3 Proof of the Exponential Rate of Decay for Trees in Theorem 3

For the claim in (9), note that for n sufficiently large,

$$P^n(\mathcal{A}_n) \geq \max\{(1-\eta)P^n(\widehat{k}_n \neq k_n|\mathcal{B}_n^c), P^n(\mathcal{B}_n)\}, \tag{56}$$

from Lemma 11 and the fact that $\mathcal{B}_n \subseteq \mathcal{A}_n$. Equation (56) gives us a lower bound on the error probability in terms of the Chow-Liu error $P^n(\mathcal{B}_n)$ and the underestimation and overestimation errors $P^n(\widehat{k}_n \neq k_n|\mathcal{B}_n^c)$. If $k = d-1$, the overestimation error probability is identically zero, so we only have to be concerned with the underestimation error. Furthermore, from (34) and a corresponding lower bound which we omit, the underestimation error event satisfies $P^n(\widehat{k}_n < k|\mathcal{B}_n^c) \doteq \exp(-nL_P)$. Combining this fact with the definition of the error exponent K_P in (24) and the result in (56) establishes (9). Note that the relation in (56) and our preceding upper bounds ensure that the limit in (9) exists. ■

Proof (*of Lemma 11*) We note that $P^n(\mathcal{A}_n|\widehat{k}_n \neq k) = 1$ and thus,

$$P^n(\mathcal{A}_n) \leq P^n(\widehat{k}_n \neq k) + P^n(\mathcal{A}_n|\widehat{k}_n = k). \tag{57}$$

By using the definition of K_P in (24), the second term in (57) is precisely $P^n(\mathcal{B}_n)$ therefore,

$$P^n(\mathcal{A}_n) \leq P^n(\widehat{k}_n \neq k) + \exp(-n(K_P-\eta)), \tag{58}$$

for all $n > N_1$. We further bound $P^n(\widehat{k}_n \neq k)$ by conditioning on the event $\mathcal{B}_n^c$. Thus, for $\eta > 0$,

$$\begin{aligned} P^n(\widehat{k}_n \neq k) &\leq P^n(\widehat{k}_n \neq k|\mathcal{B}_n^c) + P^n(\mathcal{B}_n) \\ &\leq P^n(\widehat{k}_n \neq k|\mathcal{B}_n^c) + \exp(-n(K_P-\eta)), \end{aligned} \tag{59}$$

for all $n > N_2$. The upper bound result follows by combining (58) and (59). The lower bound follows by the chain

$$\begin{aligned} P^n(\mathcal{A}_n) &\geq P^n(\widehat{k}_n \neq k) \geq P^n(\{\widehat{k}_n \neq k\} \cap \mathcal{B}_n^c) \\ &= P^n(\widehat{k}_n \neq k|\mathcal{B}_n^c)P^n(\mathcal{B}_n^c) \geq (1-\eta)P^n(\widehat{k}_n \neq k|\mathcal{B}_n^c), \end{aligned}$$

which holds for all $n > N_3$ since $P^n(\mathcal{B}_n^c) \to 1$. Now the claims in (25) and (26) follow by taking $N := \max\{N_1, N_2, N_3\}$. ■

15. We can easily remove the constraint $\mathbf{z}^T\mathbf{1}$ in (43) by a simple change of variables to only consider those vectors in the subspace orthogonal to the all ones vector so we ignore it here for simplicity. To obtain C_{P_e}, suppose the matrix $\mathbf{W}_e$ diagonalizes $\mathbf{H}_e$, that is, $\mathbf{H}_e = \mathbf{W}_e^T\mathbf{D}_e\mathbf{W}_e$, then one can, for example, choose $C_{P_e} = \min_{i:[\mathbf{D}_e]_{i,i}>0} [\mathbf{W}_e^T\mathbf{\Pi}_e\mathbf{W}_e]_{i,i}$.

Appendix C. Proof of Corollary 4

Proof This claim follows from the fact that three errors (i) Chow-Liu error (ii) underestimation error and (iii) overestimation error behave in exactly the same way as in Theorem 3. In particular, the Chow-Liu error, that is, the error for the learning the top k edges in the forest projection model $\widetilde{P}$ decays with error exponent K_P. The underestimation error behaves as in (34) and the overestimation error as in (50). ■

Appendix D. Proof of Theorem 5

Proof Given assumptions (A1) and (A2), we claim that the underestimation exponent $L_{P^{(d)}}$, defined in (34), is uniformly bounded away from zero, that is,

$$L := \inf_{d\in\mathbb{N}} L_{P^{(d)}} = \inf_{d\in\mathbb{N}} \min_{e\in E_{P^{(d)}}} L(P_e^{(d)};0) \tag{60}$$

is positive. Before providing a formal proof, we provide a plausible argument to show that this claim is true. Recall the definition of $L(P_e;0)$ in (31). Assuming that the joint $P_e = P_{i,j}$ is close to a product distribution or equivalently if its mutual information $I(P_e)$ is small (which is the worst-case scenario),

$$L(P_e;0) \approx \min_{Q\in\mathcal{P}(\mathcal{X}^2)} \{D(P_e\,||\,Q) : I(Q) = 0\} \tag{61}$$

$$= D(P_e\,||\,P_iP_j) = I(P_e) \geq I_{\text{inf}} > 0, \tag{62}$$

where in (61), the arguments in the KL-divergence have been swapped. This is because when $Q \approx P_e$ entry-wise, $D(Q\,||\,P_e) \approx D(P_e\,||\,Q)$ in the sense that their difference is small compared to their absolute values (Borade and Zheng, 2008). In (62), we used the fact that the reverse I-projection of P_e onto the set of product distributions is P_iP_j. Since I_{inf} is constant, this proves the claim, that is, $L > 0$.

More formally, let

$$B_{\kappa'} := \{Q_{i,j} \in \mathcal{P}(\mathcal{X}^2) : Q_{i,j}(x_i,x_j) \geq \kappa', \forall x_i,x_j \in \mathcal{X}\}$$

be the set of joint distributions whose entries are bounded away from zero by $\kappa' > 0$. Now, consider a pair of joint distributions $P_e^{(d)}, \widetilde{P}_e^{(d)} \in B_{\kappa'}$ whose minimum values are uniformly bounded away from zero as assumed in (A2). Then there exists a Lipschitz constant (independent of d) $U \in (0,\infty)$ such that for all d,

$$|I(P_e^{(d)}) - I(\widetilde{P}_e^{(d)})| \leq U\|\text{vec}(P_e^{(d)}) - \text{vec}(\widetilde{P}_e^{(d)})\|_1, \tag{63}$$

where $\|\cdot\|_1$ is the vector ℓ_1 norm. In fact, $U := \max_{Q\in B_{\kappa'}} \|\nabla I(\text{vec}(Q))\|_\infty$ is the Lipschitz constant of $I(\cdot)$ which is uniformly bounded because the joint distributions $P_e^{(d)}$ and $\widetilde{P}_e^{(d)}$ are assumed to be uniformly bounded away from zero. Suppose, to the contrary, $L = 0$. Then by the definition of the infimum in (60), for every $\varepsilon > 0$, there exists a $d \in \mathbb{N}$ and a corresponding $e \in E_{P^{(d)}}$ such that if Q^* is the optimizer in (31),

$$\varepsilon > D(Q^*\,||\,P_e^{(d)}) \overset{(a)}{\geq} \frac{\|\text{vec}(P_e^{(d)}) - \text{vec}(Q^*)\|_1^2}{2\log 2} \overset{(b)}{\geq} \frac{|I(P_e^{(d)}) - I(Q^*)|^2}{(2\log 2)U^2} \overset{(c)}{\geq} \frac{I_{\text{inf}}^2}{(2\log 2)U^2},$$

where (a) follows from Pinsker's inequality (Cover and Thomas, 2006, Lemma 11.6.1), (b) is an application of (63) and the fact that if $P_e^{(d)} \in B_\kappa$ is uniformly bounded from zero (as assumed in (14)) so is the associated optimizer Q^* (i.e., in $B_{\kappa''}$ for some possibly different uniform $\kappa'' > 0$). Statement (c) follows from the definition of $I_{\inf}$ and the fact that Q^* is a product distribution, that is, $I(Q^*) = 0$. Since ε can be chosen to be arbitrarily small, we arrive at a contradiction. Thus L in (60) is positive. Finally, we observe from (33) that if $n > (3/L)\log k$ the underestimation error tends to zero because (33) can be further upper bounded as

$$P^n(\widehat{k}_n < k | \mathcal{B}_n^c) \leq (n+1)^{r^2} \exp(2\log k - nL) < (n+1)^{r^2} \exp\left(\frac{2}{3}nL - nL\right) \to 0$$

as $n \to \infty$. Take $C_2 = 3/L$ in (15).

Similarly, given the same assumptions, the error exponent for structure learning $K_{P^{(d)}}$, defined in (24), is also uniformly bounded away from zero, that is,

$$K := \inf_{d \in \mathbb{N}} K_{P^{(d)}} > 0.$$

Thus, if $n > (4/K)\log d$, the error probability associated to estimating the top k edges (event $\mathcal{B}_n$) decays to zero along similar lines as in the case of the underestimation error. Take $C_1 = 4/K$ in (15).

Finally, from (49), if $n\varepsilon_n > 2\log(d-k)$, then the overestimation error tends to zero. Since from (6), ε_n can take the form $n^{-\beta}$ for $\beta > 0$, this is equivalent to $n^{1-\beta} > 2\log(d-k)$, which is the same as the first condition in (15), namely $n > (2\log(d-k))^{1+\zeta}$. By (26) and (27), these three probabilities constitute the overall error probability when learning the sequence of forest structures $\{E_{P^{(d)}}\}_{d \in \mathbb{N}}$. Thus the conditions in (15) suffice for high-dimensional consistency. ■

Appendix E. Proof of Corollary 6

Proof First note that $k_n \in \{0, \dots, d_n - 1\}$. From (49), we see that for n sufficiently large, the sequence $h_n(P) := (n\varepsilon_n)^{-1} \log P^n(\mathcal{A}_n)$ is upper bounded by

$$-1 + \frac{2}{n\varepsilon_n} \log(d_n - k_n - 1) + \frac{r^2 \log(n+1)}{n\varepsilon_n}. \tag{64}$$

The last term in (64) tends to zero by (6). Thus $h_n(P) = O((n\varepsilon_n)^{-1} \log(d_n - k_n - 1))$, where the implied constant is 2 by (64). Clearly, this sequence is maximized (resp., minimized) when $k_n = 0$ (resp., $k_n = d_n - 1$). Equation (64) also shows that the sequence h_n is monotonically decreasing in k_n. ■

Appendix F. Proof of Theorem 7

Proof We first focus on part (a). Part (b) follows in a relatively straightforward manner. Define

$$\widehat{T}_{\mathrm{MAP}}(\mathbf{x}^n) := \operatorname*{argmax}_{t \in \mathcal{T}_k^d} \mathbb{P}(T_P = t | \mathbf{x}^n)$$

to be the maximum a-posteriori (MAP) decoding rule.[16] By the optimality of the MAP rule, this lower bounds the error probability of any other estimator. Let $\mathcal{W} := \widehat{T}_{\mathrm{MAP}}((\mathcal{X}^d)^n)$ be the range of the function $\widehat{T}_{\mathrm{MAP}}$, that is, a forest $t \in \mathcal{W}$ if and only if there exists a sequence $\mathbf{x}^n$ such that $\widehat{T}_{\mathrm{MAP}} = t$. Note that $\mathcal{W} \cup \mathcal{W}^c = \mathcal{T}_k^d$. Then, consider the lower bounds:

$$\begin{aligned}
\mathbb{P}(\widehat{T} \neq T_P) &= \sum_{t \in \mathcal{T}_k^d} \mathbb{P}(\widehat{T} \neq T_P | T_P = t)\mathbb{P}(T_P = t) \\
&\geq \sum_{t \in \mathcal{W}^c} \mathbb{P}(\widehat{T} \neq T_P | T_P = t)\mathbb{P}(T_P = t) \\
&= \sum_{t \in \mathcal{W}^c} \mathbb{P}(T_P = t) = 1 - \sum_{t \in \mathcal{W}} \mathbb{P}(T_P = t) && (65) \\
&= 1 - \sum_{t \in \mathcal{W}} |\mathcal{T}_k^d|^{-1} && (66) \\
&\geq 1 - r^{nd} |\mathcal{T}_k^d|^{-1}, && (67)
\end{aligned}$$

where in (65), we used the fact that $\mathbb{P}(\widehat{T} \neq T_P | T_P = t) = 1$ if $t \in \mathcal{W}^c$, in (66), the fact that $\mathbb{P}(T_P = t) = 1/|\mathcal{T}_k^d|$. In (67), we used the observation $|\mathcal{W}| \leq (|\mathcal{X}^d|)^n = r^{nd}$ since the function $\widehat{T}_{\mathrm{MAP}} : (\mathcal{X}^d)^n \to \mathcal{W}$ is surjective. Now, the number of labeled forests with k edges and d nodes is (Aigner and Ziegler, 2009, pp. 204) $|\mathcal{T}_k^d| \geq (d-k)d^{k-1} \geq d^{k-1}$. Applying this lower bound to (67), we obtain

$$\mathbb{P}(\widehat{T} \neq T_P) \geq 1 - \exp\left(nd \log r - (k-1)\log d\right) > 1 - \exp\left((\rho - 1)(k-1)\log d\right), \qquad (68)$$

where the second inequality follows by choice of n in (17). The estimate in (68) converges to 1 as $(k,d) \to \infty$ since $\rho < 1$. The same reasoning applies to part (b) but we instead use the following estimates of the cardinality of the set of forests (Aigner and Ziegler, 2009, Ch. 30):

$$(d-2)\log d \leq \log |\mathcal{F}^d| \leq (d-1)\log(d+1). \qquad (69)$$

Note that we have lower bounded $|\mathcal{F}^d|$ by the number trees with d nodes which is d^{d-2} by Cayley's formula (Aigner and Ziegler, 2009, Ch. 30). The upper bound[17] follows by a simple combinatorial argument which is omitted. Using the lower bound in (69), we have

$$\mathbb{P}(\widehat{T} \neq T_P) \geq 1 - \exp(nd \log r)\exp(-(d-2)\log d) > 1 - d^2 \exp((\rho - 1)d \log d), \qquad (70)$$

with the choice of n in (18). The estimate in (70) converges to 1, completing the proof. ■

Appendix G. Proof of Theorem 8

Proof We assume that P is Markov on a forest since the extension to non-forest-structured P is a straightforward generalization. We start with some useful definitions. Recall from Appendix B that $\mathcal{B}_n := \{\widehat{E}_k \neq E_P\}$ is the event that the top k edges (in terms of mutual information) in the edge set $\widehat{E}_{d-1}$ are not equal to the edges in E_P. Also define $\widetilde{\mathcal{C}}_{n,\delta} := \{D(P^* \,||\, P) > \delta d\}$ to be the event that the divergence between the learned model and the true (forest) one is greater than δd. We will

16. In fact, this proof works for *any* decoding rule, and not just the MAP rule. We focus on the MAP rule for concreteness.
17. The purpose of the upper bound is to show that our estimates of $|\mathcal{F}^d|$ in (69) are reasonably tight.

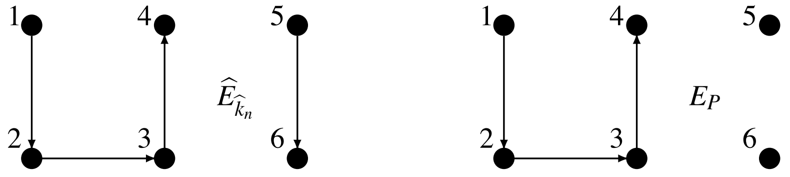


Figure 10: In $\widehat{E}_{\widehat{k}_n}$ (left), nodes 1 and 5 are the roots. The parents are defined as $\pi(i;\widehat{E}_{\widehat{k}_n}) = i-1$ for $i = 2,3,4,6$ and $\pi(i;\widehat{E}_{\widehat{k}_n}) = \emptyset$ for $i = 1,5$. In E_P (right), the parents are defined as $\pi(i;E_P) = i-1$ for $i = 2,3,4$ but $\pi(i;E_P) = \emptyset$ for $i = 1,5,6$ since $(5,6),(\emptyset,1),(\emptyset,5) \notin E_P$.

see that $\widetilde{\mathcal{C}}_{n,\delta}$ is closely related to the event of interest $\mathcal{C}_{n,\delta}$ defined in (20). Let $\mathcal{U}_n := \{\widehat{k}_n < k\}$ be the underestimation event. Our proof relies on the following result, which is similar to Lemma 11, hence its proof is omitted.

Lemma 13 *For every $\eta > 0$, there exists a $N \in \mathbb{N}$ such that for all $n > N$, the following bounds on $P^n(\widetilde{\mathcal{C}}_{n,\delta})$ hold:*

$$(1-\eta)P^n(\widetilde{\mathcal{C}}_{n,\delta}|\mathcal{B}_n^c,\mathcal{U}_n^c) \leq P^n(\widetilde{\mathcal{C}}_{n,\delta}) \tag{71}$$

$$\leq P^n(\widetilde{\mathcal{C}}_{n,\delta}|\mathcal{B}_n^c,\mathcal{U}_n^c) + \exp(-n(\min\{K_P,L_P\}-\eta)). \tag{72}$$

Note that the exponential term in (72) comes from an application of the union bound and the "largest-exponent-wins" principle in large-deviations theory (Den Hollander, 2000). From (71) and (72) we see that it is possible to bound the probability of $\widetilde{\mathcal{C}}_{n,\delta}$ by providing upper and lower bounds for $P^n(\widetilde{\mathcal{C}}_{n,\delta}|\mathcal{B}_n^c,\mathcal{U}_n^c)$. In particular, we show that the upper bound equals $\exp(-n\delta)$ to first order in the exponent. This will lead directly to (21). To proceed, we rely on the following lemma, which is a generalization of a well-known result (Cover and Thomas, 2006, Ch. 11). We defer the proof to the end of the section.

Lemma 14 (Empirical Divergence Bounds) *Let X,Y be two random variables whose joint distribution is $P_{X,Y} \in \mathcal{P}(\mathcal{X}^2)$ and $|\mathcal{X}| = r$. Let $(x^n,y^n) = \{(x_1,y_1),\dots,(x_n,y_n)\}$ be n independent and identically distributed observations drawn from $P_{X,Y}$. Then, for every n,*

$$P^n_{X,Y}(D(\widehat{P}_{X|Y}\,||\,P_{X|Y}) > \delta) \leq (n+1)^{r^2}\exp(-n\delta), \tag{73}$$

where $\widehat{P}_{X|Y} = \widehat{P}_{X,Y}/\widehat{P}_Y$ is the conditional type of (x^n,y^n). Furthermore,

$$\liminf_{n\to\infty}\frac{1}{n}\log P^n_{X,Y}(D(\widehat{P}_{X|Y}\,||\,P_{X|Y}) > \delta) \geq -\delta. \tag{74}$$

It is worth noting that the bounds in (73) and (74) are independent of the distribution $P_{X,Y}$ (cf. discussion after Theorem 8). We now proceed with the proof of Theorem 8. To do so, we consider the directed representation of a tree distribution Q (Lauritzen, 1996):

$$Q(\mathbf{x}) = \prod_{i\in V} Q_{i|\pi(i)}(x_i|x_{\pi(i)}), \tag{75}$$

where $\pi(i)$ is the parent of i in the edge set of Q (assuming a fixed root). Using (75) and conditioned on the fact that the top k edges of the graph of P^* are the same as those in E_P (event $\mathcal{B}_n^c$) and underestimation does not occur (event $\mathcal{U}_n^c$), the KL-divergence between P^* (which is a function of the samples $\mathbf{x}^n$ and hence of n) and P can be expressed as a sum over d terms:

$$D(P^*\,||\,P) = \sum_{i\in V} D(\widehat{P}_{i|\pi(i;\widehat{E}_{\widehat{k}_n})}\,||\,P_{i|\pi(i;E_P)}), \tag{76}$$

where the parent of node i in $\widehat{E}_{\widehat{k}_n}$, denoted $\pi(i;\widehat{E}_{\widehat{k}_n})$, is defined by arbitrarily choosing a root in each component tree of the forest $\widehat{T}_{\widehat{k}_n} = (V, \widehat{E}_{\widehat{k}_n})$. The parents of the chosen roots are empty sets. The parent of node i in E_P are "matched" to those in $\widehat{E}_{\widehat{k}_n}$, that is, defined as $\pi(i;E_P) := \pi(i;\widehat{E}_{\widehat{k}_n})$ if $(i,\pi(i;\widehat{E}_{\widehat{k}_n})) \in E_P$ and $\pi(i;E_P) := \emptyset$ otherwise. See Figure 10 for an example. Note that this can be done because $\widehat{E}_{\widehat{k}_n} \supseteq E_P$ by conditioning on the events $\mathcal{B}_n^c$ and $\mathcal{U}_n^c = \{\widehat{k}_n \geq k\}$. Then, the error probability $P^n(\widetilde{\mathcal{C}}_{n,\delta}|\mathcal{B}_n^c, \mathcal{U}_n^c)$ in (72) can be upper bounded as

$$\begin{aligned}
P^n(\widetilde{\mathcal{C}}_{n,\delta}|\mathcal{B}_n^c,\mathcal{U}_n^c) &= P^n\left(\sum_{i\in V} D(\widehat{P}_{i|\pi(i;\widehat{E}_{\widehat{k}_n})}||P_{i|\pi(i;E_P)}) > \delta d \,\middle|\, \mathcal{B}_n^c,\mathcal{U}_n^c\right) && (77)\\
&= P^n\left(\frac{1}{d}\sum_{i\in V} D(\widehat{P}_{i|\pi(i;\widehat{E}_{\widehat{k}_n})}||P_{i|\pi(i;E_P)}) > \delta \,\middle|\, \mathcal{B}_n^c,\mathcal{U}_n^c\right)\\
&\leq P^n\left(\max_{i\in V}\left\{D(\widehat{P}_{i|\pi(i;\widehat{E}_{\widehat{k}_n})}||P_{i|\pi(i;E_P)})\right\} > \delta \,\middle|\, \mathcal{B}_n^c,\mathcal{U}_n^c\right) && (78)\\
&\leq \sum_{i\in V} P^n\left(D(\widehat{P}_{i|\pi(i;\widehat{E}_{\widehat{k}_n})}||P_{i|\pi(i;E_P)}) > \delta \,\middle|\, \mathcal{B}_n^c,\mathcal{U}_n^c\right) && (79)\\
&\leq \sum_{i\in V} (n+1)^{r^2}\exp(-n\delta) = d(n+1)^{r^2}\exp(-n\delta), && (80)
\end{aligned}$$

where Equation (77) follows from the decomposition in (76). Equation (78) follows from the fact that if the arithmetic mean of d positive numbers exceeds δ, then the maximum exceeds δ. Equation (79) follows from the union bound. Equation (80), which holds for all $n \in \mathbb{N}$, follows from the upper bound in (73). Combining (72) and (80) shows that if $\delta < \min\{K_P, L_P\}$,

$$\limsup_{n\to\infty} \frac{1}{n} \log P^n(\widetilde{\mathcal{C}}_{n,\delta}) \leq -\delta.$$

Now recall that $\widetilde{\mathcal{C}}_{n,\delta} = \{D(P^*\,||\,P) > \delta d\}$. In order to complete the proof of (21), we need to swap the arguments in the KL-divergence to bound the probability of the event $\mathcal{C}_{n,\delta} = \{D(P\,||\,P^*) > \delta d\}$. To this end, note that for $\varepsilon > 0$ and n sufficiently large, $|D(P^*\,||\,P) - D(P\,||\,P^*)| < \varepsilon$ with high probability since the two KL-divergences become close ($P^* \approx P$ w.h.p. as $n \to \infty$). More precisely, the probability of $\{|D(P^*\,||\,P) - D(P\,||\,P^*)| \geq \varepsilon\} = \{o(\|P - P^*\|_\infty^2) \geq \varepsilon\}$ decays exponentially with some rate $M_P > 0$. Hence,

$$\limsup_{n\to\infty} \frac{1}{n} \log P^n(D(P\,||\,P^*) > \delta d) \leq -\delta, \tag{81}$$

if $\delta < \min\{K_P, L_P, M_P\}$. If P is not Markov on a forest, (81) holds with the forest projection $\widetilde{P}$ in place of P, that is,

$$\limsup_{n\to\infty} \frac{1}{n} \log P^n(D(\widetilde{P}\,||\,P^*) > \delta d) \leq -\delta. \tag{82}$$

The Pythagorean relationship (Simon, 1973; Bach and Jordan, 2003) states that

$$D(P\,||\,P^*) = D(P\,||\,\widetilde{P}) + D(\widetilde{P}\,||\,P^*) \tag{83}$$

which means that the risk is $\mathcal{R}_n(P^*) = D(\widetilde{P}\,||\,P^*)$. Combining this fact with (82) implies the assertion of (21) by choosing $\delta_0 := \min\{K_P, L_P, M_P\}$.

Now we exploit the lower bound in Lemma 14 to prove the lower bound in Theorem 8. The error probability $P^n(\widetilde{\mathcal{C}}_{n,\delta}|\mathcal{B}_n^c,\mathcal{U}_n^c)$ in (72) can now be lower bounded by

$$\begin{aligned} P^n(\widetilde{\mathcal{C}}_{n,\delta}|\mathcal{B}_n^c,\mathcal{U}_n^c) &\ge \max_{i\in V} P^n\left(D(\widehat{P}_{i|\pi(i;\widehat{E}_{\widehat{k}_n})}\,||\,P_{i|\pi(i;E_P)}) > \delta d \,\middle|\, \mathcal{B}_n^c,\mathcal{U}_n^c\right) & (84)\\ &\ge \exp(-n(\delta d+\eta)), & (85)\end{aligned}$$

where (84) follows from the decomposition in (77) and (85) holds for every η for sufficiently large n by (74). Using the same argument that allows us to swap the arguments of the KL-divergence as in the proof of the upper bound completes the proof of (22). ■

Proof (*of Lemma 14*) Define the δ-*conditional-typical set with respect to* $P_{X,Y}\in\mathcal{P}(\mathcal{X}^2)$ as

$$\mathcal{S}^{\delta}_{P_{X,Y}} := \{(x^n,y^n)\in(\mathcal{X}^2)^n : D(\widehat{P}_{X|Y}\,||\,P_{X|Y}) \le \delta\},$$

where $\widehat{P}_{X|Y}$ is the conditional type of (x^n,y^n). We now estimate the $P^n_{X,Y}$-probability of the δ-conditional-atypical set, that is, $P^n_{X,Y}((\mathcal{S}^{\delta}_{P_{X,Y}})^c)$

$$\begin{aligned} P^n_{X,Y}((\mathcal{S}^{\delta}_{P_{X,Y}})^c) &= \sum_{(x^n,y^n)\in\mathcal{X}^2 : D(\widehat{P}_{X|Y}||P_{X|Y})>\delta} P^n_{X,Y}((x^n,y^n)) & (86)\\ &= \sum_{Q_{X,Y}\in\mathcal{P}_n(\mathcal{X}^2):D(Q_{X|Y}||P_{X|Y})>\delta} P^n_{X,Y}(\mathsf{T}_n(Q_{X,Y})) & (87)\\ &\le \sum_{Q_{X,Y}\in\mathcal{P}_n(\mathcal{X}^2):D(Q_{X|Y}||P_{X|Y})>\delta} \exp(-nD(Q_{X,Y}\,||\,P_{X,Y})) & (88)\\ &\le \sum_{Q_{X,Y}\in\mathcal{P}_n(\mathcal{X}^2):D(Q_{X|Y}||P_{X|Y})>\delta} \exp(-nD(Q_{X|Y}\,||\,P_{X|Y})) & (89)\\ &\le \sum_{Q_{X,Y}\in\mathcal{P}_n(\mathcal{X}^2):D(Q_{X|Y}||P_{X|Y})>\delta} \exp(-n\delta) & (90)\\ &\le (n+1)^{r^2}\exp(-n\delta), & (91)\end{aligned}$$

where (86) and (87) are the same because summing over sequences is equivalent to summing over the corresponding type classes since every sequence in each type class has the same probability (Cover and Thomas, 2006, Ch. 11). Equation (88) follows from the method of types result in Lemma 12. Equation (89) follows from the KL-divergence version of the chain rule, namely,

$$D(Q_{X,Y}\,||\,P_{X,Y}) = D(Q_{X|Y}\,||\,P_{X|Y}) + D(Q_Y\,||\,P_Y)$$

and non-negativity of the KL-divergence $D(Q_Y\,||\,P_Y)$. Equation (90) follows from the fact that $D(Q_{X|Y}\,||\,P_{X|Y}) > \delta$ for $Q_{X,Y}\in(\mathcal{S}^{\delta}_{P_{X,Y}})^c$. Finally, (91) follows the fact that the number of types with denominator n and alphabet $\mathcal{X}^2$ is upper bounded by $(n+1)^{r^2}$. This concludes the proof of (73).

We now prove the lower bound in (74). To this end, construct a sequence of distributions $\{Q_{X,Y}^{(n)} \in \mathcal{P}_n(\mathcal{X}^2)\}_{n\in\mathbb{N}}$ such that $Q_Y^{(n)} = P_Y$ and $D(Q_{X|Y}^{(n)} || P_{X|Y}) \to \delta$. Such a sequence exists by the denseness of types in the probability simplex (Dembo and Zeitouni, 1998, Lemma 2.1.2(b)). Now we lower bound (87):

$$P_{X,Y}^n((\mathcal{S}_{P_{X,Y}}^{\delta})^c) \geq P_{X,Y}^n(\mathsf{T}_n(Q_{X,Y}^{(n)})) \geq (n+1)^{-r^2} \exp(-nD(Q_{X,Y}^{(n)} \,||\, P_{X,Y})). \tag{92}$$

Taking the normalized logarithm and lim inf in n on both sides of (92) yields

$$\liminf_{n\to\infty} \frac{1}{n} \log P_{X,Y}^n((\mathcal{S}_{P_{X,Y}}^{\delta})^c) \geq \liminf_{n\to\infty} \left\{ -D(Q_{X|Y}^{(n)} || P_{X|Y}) - D(Q_Y^{(n)} || P_Y) \right\} = -\delta.$$

This concludes the proof of Lemma 14. ■

Appendix H. Proof of Corollary 9

Proof If the dimension $d = o(\exp(n\delta))$, then the upper bound in (80) is asymptotically majorized by $\mathrm{poly}(n) o(\exp(na)) \exp(-n\delta) = o(\exp(n\delta)) \exp(-n\delta)$, which can be made arbitrarily small for n sufficiently large. Thus the probability tends to zero as $n \to \infty$. ■

Appendix I. Proof of Theorem 10

Proof In this proof, we drop the superscript (d) for all distributions P for notational simplicity but note that $d = d_n$. We first claim that $D(P^* \,||\, \widetilde{P}) = O_p(d \log d / n^{1-\gamma})$. Note from (72) and (80) that by taking $\delta = (\tau \log d)/n^{1-\gamma}$ (for any $\tau > 0$),

$$P^n \left(\frac{n^{1-\gamma}}{d \log d} D(P^* \,||\, \widetilde{P}) > \tau \right) \leq d(n+1)^{r^2} \exp(-\tau n^{\gamma} \log d) + \exp(-\Theta(n)) = o_n(1). \tag{93}$$

Therefore, the scaled sequence of random variables $\frac{n^{1-\gamma}}{d \log d} D(P^* \,||\, \widetilde{P})$ is stochastically bounded (Serfling, 1980) which proves the claim.[18]

Now, we claim that $D(\widetilde{P} || P^*) = O_p(d \log d / n^{1-\gamma})$. A simple calculation using Pinsker's Inequality and Lemma 6.3 in Csiszár and Talata (2006) yields

$$D(\widehat{P}_{X,Y} \,||\, P_{X,Y}) \leq \frac{c}{\kappa} D(P_{X,Y} \,||\, \widehat{P}_{X,Y}),$$

where $\kappa := \min_{x,y} P_{X,Y}(x,y)$ and $c = 2 \log 2$. Using this fact, we can use (73) to show that for all n sufficiently large,

$$P_{X,Y}^n(D(P_{X|Y} \,||\, \widehat{P}_{X|Y}) > \delta) \leq (n+1)^{r^2} \exp(-n\delta\kappa/c),$$

that is, if the arguments in the KL-divergence in (73) are swapped, then the exponent is reduced by a factor proportional to κ. Using this fact and the assumption in (14) (uniformity of the minimum

18. In fact, we have in fact proven the stronger assertion that $D(P^* \,||\, \widetilde{P}) = o_p(d \log d / n^{1-\gamma})$ since the right-hand-side of (93) converges to zero.

entry in the pairwise joint $\kappa > 0$), we can replicate the proof of the result in (80) with $\delta\kappa/c$ in place of δ giving

$$P^n(D(P\,||\,P^*) > \delta) \leq d(n+1)^{r^2} \exp(-n\delta\kappa/c).$$

We then arrive at a similar result to (93) by taking $\delta = (\tau \log d)/n^{1-\gamma}$. We conclude that $D(\widetilde{P}\,||\,P^*) = O_p(d \log d/n^{1-\gamma})$. This completes the proof of the claim.

Equation (23) then follows from the definition of the risk in (19) and from the Pythagorean theorem in (83). This implies the assertion of Theorem 10. ■

References

P. Abbeel, D. Koller, and A. Y. Ng. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, Dec 2006.

M. Aigner and G. M. Ziegler. *Proofs From THE BOOK*. Springer, 2009.

F. Bach and M. I. Jordan. Beyond independent components: trees and clusters. *Journal of Machine Learning Research*, 4:1205–1233, 2003.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2008.

S. Borade and L. Zheng. Euclidean information theory. In *IEEE International Zurich Seminar on Communications*, pages 14–17, 2008.

G. Bresler, E. Mossel, and A. Sly. Reconstruction of Markov random fields from samples: Some observations and algorithms. In *11th International workshop APPROX 2008 and 12th International workshop RANDOM*, pages 343–356., 2008.

A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Advances of Neural Information Processing Systems (NIPS)*, 2007.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Infomation Theory*, 14(3):462–467, May 1968.

C. K. Chow and T. Wagner. Consistency of an estimate of tree-dependent probability distributions . *IEEE Transactions in Information Theory*, 19(3):369 – 371, May 1973.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

I. Csiszár and F. Matúš. Information projections revisited. *IEEE Transactions on Infomation Theory*, 49(6):1474–1490, 2003.

I. Csiszár and P. Shields. The consistency of the BIC Markov order estimator. *Ann. Statist.*, 28(6): 1601–1619, 2000.

I. Csiszár and Z. Talata. Context tree estimation for not necessarily finite memory processes, via bic and mdl. *IEEE Transactions on Information Theory*, 52(3):1007–16, 2006.

A. Custovic, B. M. Simpson, C. S. Murray, L. Lowe, and A. Woodcock. The national asthma campaign Manchester asthma and allergy study. *Pediatr Allergy Immunol*, 13:32–37, 2002.

A. Dembo and O. Zeitouni. *Large Deviations Techniques and Applications*. Springer, 2nd edition, 1998.

F. Den Hollander. *Large Deviations (Fields Institute Monographs, 14)*. American Mathematical Society, Feb 2000.

M. Dudik, S. J. Phillips, and R. E. Schapire. Performance guarantees for regularized maximum entropy density estimation. In *Conference on Learning Theory (COLT)*, 2004.

R. M. Fano. *Transmission of Information*. New York: Wiley, 1961.

L. Finesso, C. C. Liu, and P. Narayan. The optimal error exponent for Markov order estimation. *IEEE Trans. on Info Th.*, 42(5):1488–1497, 1996.

R. G. Gallager. Claude E. Shannon: A retrospective on his life, work and impact. *IEEE Trans. on Info. Th.*, 47:2687–95, Nov 2001.

E. Gassiat and S. Boucheron. Optimal error exponents in hidden Markov models order estimation. *IEEE Transactions on Infomation Theory*, 49(4):964–980, Apr 2003.

J. Johnson, V. Chandrasekaran, and A. S. Willsky. Learning Markov structure by maximum entropy relaxation. In *Artificial Intelligence and Statistics (AISTATS)*, 2007.

J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1), Feb 1956.

S. Lauritzen. *Graphical Models*. Oxford University Press, USA, 1996.

S. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

H. Liu, M. Xu, H. Gu, A. Gupta, J. Lafferty, and L. Wasserman. Forest density estimation. *Journal of Machine Learning Research*, 12:907–951, March 2011.

M. Meilă and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, Oct 2000.

N. Meinshausen and P. Buehlmann. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

N. Merhav. The estimation of the model order in exponential families. *IEEE Transactions on Infomation Theory*, 35(5):1109–1115, 1989.

N. Merhav, M. Gutman., and J. Ziv. On the estimation of the order of a Markov chain and universal data compression. *IEEE Transactions on Infomation Theory*, 35:1014–1019, 1989.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI Repository of Machine Learning Databases, University of California, Irvine, 1998.

R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36, 1957.

N. Santhanam and M. J. Wainwright. Information-theoretic limits of selecting binary graphical models in high dimensions. In *Proc. of IEEE Intl. Symp. on Info. Theory*, Toronto, Canada, July 2008.

R. J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley-Interscience, Nov 1980.

G. Simon. Additivity of information in exponential family probability laws. *Amer. Statist. Assoc.*, 68(478–482), 1973.

A. Simpson, V. Y. F. Tan, J. M. Winn, M. Svensén, C. M. Bishop, D. E. Heckerman, I. Buchan, and A. Custovic. Beyond atopy: Multiple patterns of sensitization in relation to asthma in a birth cohort study. *Am J Respir Crit Care Med*, 2010.

V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning Gaussian tree models: Analysis of error exponents and extremal structures. *IEEE Transactions on Signal Processing*, 58(5):2701 – 2714, May 2010a.

V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Error exponents for composite hypothesis testing of Markov forest distributions. In *Proc. of Intl. Symp. on Info. Th.*, June 2010b.

V. Y. F. Tan, A. Anandkumar, L. Tong, and A. S. Willsky. A large-deviation analysis for the maximum-likelihood learning of Markov tree structures. *IEEE Transactions on Infomation Theory*, Mar 2011.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, Mar 1996.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical report, University of California, Berkeley, 2003.

M. J. Wainwright, P. Ravikumar, and J. D. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In *Advances of Neural Information Processing Systems (NIPS)*, pages 1465–1472, 2006.

O. Zuk, S. Margel, and E. Domany. On the number of samples needed to learn the correct structure of a Bayesian network. In *Proc of Uncertainty in Artificial Intelligence (UAI)*, 2006.

Journal of Machine Learning Research 12 (2011) 1655-1695 Submitted 1/10; Revised 11/10; Published 5/11

$\mathcal{X}$-Armed Bandits

Sébastien Bubeck SBUBECK@CRM.CAT
Centre de Recerca Matemàtica
Campus de Bellaterra, Edifici C
08193 Bellaterra (Barcelona), Spain

Rémi Munos REMI.MUNOS@INRIA.FR
INRIA Lille, SequeL Project
40 avenue Halley
59650 Villeneuve d'Ascq, France

Gilles Stoltz[*] GILLES.STOLTZ@ENS.FR
Ecole Normale Supérieure, CNRS, INRIA
45 rue d'Ulm
75005 Paris, France

Csaba Szepesvári SZEPESVA@CS.UALBERTA.CA
University of Alberta
Department of Computing Science
Edmonton T6G 2E8, Canada

Editor: Nicolò Cesa-Bianchi

Abstract

We consider a generalization of stochastic bandits where the set of arms, $\mathcal{X}$, is allowed to be a generic measurable space and the mean-payoff function is "locally Lipschitz" with respect to a dissimilarity function that is known to the decision maker. Under this condition we construct an arm selection policy, called HOO (hierarchical optimistic optimization), with improved regret bounds compared to previous results for a large class of problems. In particular, our results imply that if $\mathcal{X}$ is the unit hypercube in a Euclidean space and the mean-payoff function has a finite number of global maxima around which the behavior of the function is locally continuous with a known smoothness degree, then the expected regret of HOO is bounded up to a logarithmic factor by $\sqrt{n}$, that is, the rate of growth of the regret is independent of the dimension of the space. We also prove the minimax optimality of our algorithm when the dissimilarity is a metric. Our basic strategy has quadratic computational complexity as a function of the number of time steps and does not rely on the doubling trick. We also introduce a modified strategy, which relies on the doubling trick but runs in linearithmic time. Both results are improvements with respect to previous approaches.

Keywords: bandits with infinitely many arms, optimistic online optimization, regret bounds, minimax rates

1. Introduction

In the classical stochastic bandit problem a gambler tries to maximize his revenue by sequentially playing one of a finite number of slot machines that are associated with initially unknown (and potentially different) payoff distributions (Robbins, 1952). Assuming old-fashioned slot machines,

$*$. Also at HEC Paris, CNRS, 1 rue de la Libération, 78351 Jouy-en-Josas, France.

the gambler pulls the arms of the machines one by one in a sequential manner, simultaneously learning about the machines' payoff-distributions and gaining actual monetary reward. Thus, in order to maximize his gain, the gambler must choose the next arm by taking into consideration both the urgency of gaining reward ("exploitation") and acquiring new information ("exploration").

Maximizing the total cumulative payoff is equivalent to minimizing the (total) *regret*, that is, minimizing the difference between the total cumulative payoff of the gambler and the one of another clairvoyant gambler who chooses the arm with the best mean-payoff in every round. The quality of the gambler's strategy can be characterized as the rate of growth of his expected regret with time. In particular, if this rate of growth is sublinear, the gambler in the long run plays as well as the clairvoyant gambler. In this case the gambler's strategy is called Hannan consistent.

Bandit problems have been studied in the Bayesian framework (Gittins, 1989), as well as in the frequentist parametric (Lai and Robbins, 1985; Agrawal, 1995a) and non-parametric settings (Auer et al., 2002a), and even in non-stochastic scenarios (Auer et al., 2002b; Cesa-Bianchi and Lugosi, 2006). While in the Bayesian case the question is whether the optimal actions can be computed efficiently, in the frequentist case the question is how to achieve low rate of growth of the regret in the lack of prior information, that is, it is a statistical question. In this paper we consider the stochastic, frequentist, non-parametric setting.

Although the first papers studied bandits with a finite number of arms, researchers have soon realized that bandits with infinitely many arms are also interesting, as well as practically significant. One particularly important case is when the arms are identified by a finite number of continuous-valued parameters, resulting in *online optimization* problems over continuous finite-dimensional spaces. Such problems are ubiquitous to operations research and control. Examples are "pricing a new product with uncertain demand in order to maximize revenue, controlling the transmission power of a wireless communication system in a noisy channel to maximize the number of bits transmitted per unit of power, and calibrating the temperature or levels of other inputs to a reaction so as to maximize the yield of a chemical process" (Cope, 2009). Other examples are optimizing parameters of schedules, rotational systems, traffic networks or online parameter tuning of numerical methods. During the last decades numerous authors have investigated such "continuum-armed" bandit problems (Agrawal, 1995b; Kleinberg, 2004; Auer et al., 2007; Kleinberg et al., 2008a; Cope, 2009). A special case of interest, which forms a bridge between the case of a finite number of arms and the continuum-armed setting, is formed by bandit linear optimization, see Abernethy et al. (2008) and the references therein.

In many of the above-mentioned problems, however, the natural domain of some of the optimization parameters is a discrete set, while other parameters are still continuous-valued. For example, in the pricing problem different product lines could also be tested while tuning the price, or in the case of transmission power control different protocols could be tested while optimizing the power. In other problems, such as in online sequential search, the parameter-vector to be optimized is an infinite sequence over a finite alphabet (Coquelin and Munos, 2007; Bubeck and Munos, 2010).

The motivation for this paper is to handle all these various cases in a unified framework. More precisely, we consider a general setting that allows us to study bandits with almost no restriction on the set of arms. In particular, we allow the set of arms to be an arbitrary measurable space. Since we allow non-denumerable sets, we shall assume that the gambler has some knowledge about the behavior of the mean-payoff function (in terms of its local regularity around its maxima, roughly speaking). This is because when the set of arms is uncountably infinite and absolutely no assumptions are made on the payoff function, it is impossible to construct a strategy that simultaneously achieves sublinear regret for all bandits problems (see, e.g., Bubeck et al., 2011, Corollary 4). When the set of arms is a metric space (possibly with the power of the continuum) previous works have assumed either the global smoothness of the payoff function (Agrawal, 1995b; Kleinberg, 2004; Kleinberg et al., 2008a; Cope, 2009) or local smoothness in the vicinity of the maxima (Auer et al., 2007). Here, smoothness means that the payoff function is either Lipschitz or Hölder continuous (locally or globally). These smoothness assumptions are indeed reasonable in many practical problems of interest.

In this paper, we assume that there exists a dissimilarity function that constrains the behavior of the mean-payoff function, where a dissimilarity function is a measure of the discrepancy between two arms that is neither symmetric, nor reflexive, nor satisfies the triangle inequality. (The same notion was introduced simultaneously and independently of us by Kleinberg et al., 2008b, Section 4.4, under the name "quasi-distance.") In particular, the dissimilarity function is assumed to locally set a bound on the decrease of the mean-payoff function at each of its global maxima. We also assume that the decision maker can construct a recursive covering of the space of arms in such a way that the diameters of the sets in the covering shrink at a known geometric rate when measured with this dissimilarity.

1.1 Relation to the Literature

Our work generalizes and improves previous works on continuum-armed bandits.

In particular, Kleinberg (2004) and Auer et al. (2007) focused on one-dimensional problems, while we allow general spaces. In this sense, the closest work to the present contribution is that of Kleinberg et al. (2008a), who considered generic metric spaces assuming that the mean-payoff function is Lipschitz with respect to the (known) metric of the space; its full version (Kleinberg et al., 2008b) relaxed this condition and only requires that the mean-payoff function is Lipschitz at some maximum with respect to some (known) dissimilarity.[1] Kleinberg et al. (2008b) proposed a novel algorithm that achieves essentially the best possible regret bound in a minimax sense with respect to the environments studied, as well as a much better regret bound if the mean-payoff function has a small "zooming dimension".

Our contribution furthers these works in two ways:

(i) our algorithms, motivated by the recent successful tree-based optimization algorithms (Kocsis and Szepesvari, 2006; Gelly et al., 2006; Coquelin and Munos, 2007), are easy to implement;

(ii) we show that a version of our main algorithm is able to exploit the local properties of the mean-payoff function at its maxima only, which, as far as we know, was not investigated in the approach of Kleinberg et al. (2008a,b).

The precise discussion of the improvements (and drawbacks) with respect to the papers by Kleinberg et al. (2008a,b) requires the introduction of somewhat extensive notations and is therefore deferred to Section 5. However, in a nutshell, the following can be said.

1. The present paper paper is a concurrent and independent work with respect to the paper of Kleinberg et al. (2008b). An extended abstract (Kleinberg et al., 2008a) of the latter was published in May 2008 at STOC'08, while the NIPS'08 version (Bubeck et al., 2009) of the present paper was submitted at the beginning of June 2008. At that time, we were not aware of the existence of the full version (Kleinberg et al., 2008b), which was released in September 2008.

First, by resorting to a hierarchical approach, we are able to avoid the use of the doubling trick, as well as the need for the (covering) oracle, both of which the so-called zooming algorithm of Kleinberg et al. (2008a) relies on. This comes at the cost of slightly more restrictive assumptions on the mean-payoff function, as well as a more involved analysis. Moreover, the oracle is replaced by an *a priori* choice of a covering tree. In standard metric spaces, such as the Euclidean spaces, such trees are trivial to construct, though, in full generality they may be difficult to obtain when their construction must start from (say) a distance function only. We also propose a variant of our algorithm that has smaller computational complexity of order $n \ln n$ compared to the quadratic complexity n^2 of our basic algorithm. However, the cheaper algorithm requires the doubling trick to achieve an anytime guarantee (just like the zooming algorithm).

Second, we are also able to weaken our assumptions and to consider only properties of the mean-payoff function in the neighborhoods of its maxima; this leads to regret bounds scaling as $\widetilde{O}(\sqrt{n})$[2] when, for example, the space is the unit hypercube and the mean-payoff function has a finite number of global maxima x^* around which it is locally equivalent to a function $\|x - x^*\|^\alpha$ with some known degree $\alpha > 0$. Thus, in this case, we get the desirable property that the rate of growth of the regret is independent of the dimensionality of the input space. (Comparable dimensionality-free rates are obtained under different assumptions in Kleinberg et al., 2008b.)

Finally, in addition to the strong theoretical guarantees, we expect our algorithm to work well in practice since the algorithm is very close to the recent, empirically very successful tree-search methods from the games and planning literature (Gelly and Silver, 2007, 2008; Schadd et al., 2008; Chaslot et al., 2008; Finnsson and Bjornsson, 2008).

1.2 Outline

The outline of the paper is as follows:

1. In Section 2 we formalize the $\mathcal{X}$-armed bandit problem.

2. In Section 3 we describe the basic strategy proposed, called HOO (*hierarchical optimistic optimization*).

3. We present the main results in Section 4. We start by specifying and explaining our assumptions (Section 4.1) under which various regret bounds are proved. Then we prove a distribution-dependent bound for the basic version of HOO (Section 4.2). A problem with the basic algorithm is that its computational cost increases quadratically with the number of time steps. Assuming the knowledge of the horizon, we thus propose a computationally more efficient variant of the basic algorithm, called *truncated HOO* and prove that it enjoys a regret bound identical to the one of the basic version (Section 4.3) while its computational complexity is only log-linear in the number of time steps. The first set of assumptions constrains the mean-payoff function everywhere. A second set of assumptions is therefore presented that puts constraints on the mean-payoff function only in a small vicinity of its global maxima; we then propose another algorithm, called *local-HOO*, which is proven to enjoy a regret again essentially similar to the one of the basic version (Section 4.4). Finally, we prove the minimax optimality of HOO in metric spaces (Section 4.5).

4. In Section 5 we compare the results of this paper with previous works.

2. We write $u_n = \widetilde{O}(v_n)$ when $u_n = O(v_n)$ up to a logarithmic factor.

2. Problem Setup

A *stochastic bandit problem* $\mathcal{B}$ is a pair $\mathcal{B} = (\mathcal{X}, M)$, where $\mathcal{X}$ is a measurable space of arms and M determines the distribution of rewards associated with each arm. We say that M is a *bandit environment* on $\mathcal{X}$. Formally, M is an mapping $\mathcal{X} \to \mathcal{M}_1(\mathbb{R})$, where $\mathcal{M}_1(\mathbb{R})$ is the space of probability distributions over the reals. The distribution assigned to arm $x \in \mathcal{X}$ is denoted by M_x. We require that for each arm $x \in \mathcal{X}$, the distribution M_x admits a first-order moment; we then denote by $f(x)$ its expectation ("mean payoff"),

$$f(x) = \int y \, dM_x(y) .$$

The mean-payoff function f thus defined is assumed to be measurable. For simplicity, we shall also assume that all M_x have bounded supports, included in some fixed bounded interval,[3] say, the unit interval $[0,1]$. Then, f also takes bounded values, in $[0,1]$.

A decision maker (the gambler of the introduction) that interacts with a stochastic bandit problem $\mathcal{B}$ plays a game at discrete time steps according to the following rules. In the first round the decision maker can select an arm $X_1 \in \mathcal{X}$ and receives a reward Y_1 drawn at random from M_{X_1}. In round $n > 1$ the decision maker can select an arm $X_n \in \mathcal{X}$ based on the information available up to time n, that is, $(X_1, Y_1, \ldots, X_{n-1}, Y_{n-1})$, and receives a reward Y_n drawn from M_{X_n}, independently of $(X_1, Y_1, \ldots, X_{n-1}, Y_{n-1})$ given X_n. Note that a decision maker may randomize his choice, but can only use information available up to the point in time when the choice is made.

Formally, a *strategy of the decision maker* in this game ("bandit strategy") can be described by an infinite sequence of measurable mappings, $\varphi = (\varphi_1, \varphi_2, \ldots)$, where φ_n maps the space of past observations,

$$\mathcal{H}_n = \big(\mathcal{X} \times [0,1]\big)^{n-1},$$

to the space of probability measures over $\mathcal{X}$. By convention, φ_1 does not take any argument. A strategy is called *deterministic* if for every n, φ_n is a Dirac distribution.

The goal of the decision maker is to maximize his expected cumulative reward. Equivalently, the goal can be expressed as minimizing the expected cumulative regret, which is defined as follows. Let

$$f^* = \sup_{x \in \mathcal{X}} f(x)$$

be the best expected payoff in a single round. At round n, the *cumulative regret* of a decision maker playing $\mathcal{B}$ is

$$\widehat{R}_n = n\, f^* - \sum_{t=1}^{n} Y_t ,$$

that is, the difference between the maximum expected payoff in n rounds and the actual total payoff. In the sequel, we shall restrict our attention to the expected cumulative regret, which is defined as the expectation $\mathbb{E}[\widehat{R}_n]$ of the cumulative regret $\widehat{R}_n$.

Finally, we define the cumulative *pseudo-regret* as

$$R_n = n\, f^* - \sum_{t=1}^{n} f(X_t) ,$$

3. More generally, our results would also hold when the tails of the reward distributions are uniformly sub-Gaussian.

that is, the actual rewards used in the definition of the regret are replaced by the mean-payoffs of the arms pulled. Since (by the tower rule)

$$\mathbb{E}\big[Y_t\big] = \mathbb{E}\big[\mathbb{E}\left[Y_t|X_t\right]\big] = \mathbb{E}\big[f(X_t)\big],$$

the expected values $\mathbb{E}[\widehat{R}_n]$ of the cumulative regret and $\mathbb{E}[R_n]$ of the cumulative pseudo-regret are the same. Thus, we focus below on the study of the behavior of $\mathbb{E}\big[R_n\big]$.

Remark 1 *As it is argued in Bubeck et al. (2011), in many real-world problems, the decision maker is not interested in his cumulative regret but rather in its simple regret. The latter can be defined as follows. After n rounds of play in a stochastic bandit problem $\mathcal{B}$, the decision maker is asked to make a recommendation $Z_n \in \mathcal{X}$ based on the n obtained rewards $Y_1, \ldots, Y_n$. The simple regret of this recommendation equals*

$$r_n = f^* - f(Z_n).$$

In this paper we focus on the cumulative regret R_n, but all the results can be readily extended to the simple regret by considering the recommendation $Z_n = X_{T_n}$, where T_n is drawn uniformly at random in $\{1, \ldots, n\}$. Indeed, in this case,

$$\mathbb{E}\big[r_n\big] \leqslant \frac{\mathbb{E}\big[R_n\big]}{n},$$

as is shown in Bubeck et al. (2011, Section 3).

3. The Hierarchical Optimistic Optimization (HOO) Strategy

The HOO strategy (cf., Algorithm 1) incrementally builds an estimate of the mean-payoff function f over $\mathcal{X}$. The core idea (as in previous works) is to estimate f precisely around its maxima, while estimating it loosely in other parts of the space $\mathcal{X}$. To implement this idea, HOO maintains a binary tree whose nodes are associated with measurable regions of the arm-space $\mathcal{X}$ such that the regions associated with nodes deeper in the tree (further away from the root) represent increasingly smaller subsets of $\mathcal{X}$. The tree is built in an incremental manner. At each node of the tree, HOO stores some statistics based on the information received in previous rounds. In particular, HOO keeps track of the number of times a node was traversed up to round n and the corresponding empirical average of the rewards received so far. Based on these, HOO assigns an optimistic estimate (denoted by B) to the maximum mean-payoff associated with each node. These estimates are then used to select the next node to "play". This is done by traversing the tree, beginning from the root, and always following the node with the highest B-value (cf., lines 4–14 of Algorithm 1). Once a node is selected, a point in the region associated with it is chosen (line 16) and is sent to the environment. Based on the point selected and the received reward, the tree is updated (lines 18–33).

The tree of coverings which HOO needs to receive as an input is an infinite binary tree whose nodes are associated with subsets of $\mathcal{X}$. The nodes in this tree are indexed by pairs of integers (h,i); node (h,i) is located at depth $h \geqslant 0$ from the root. The range of the second index, i, associated with nodes at depth h is restricted by $1 \leqslant i \leqslant 2^h$. Thus, the root node is denoted by $(0,1)$. By convention, $(h+1, 2i-1)$ and $(h+1, 2i)$ are used to refer to the two children of the node (h,i). Let $\mathcal{P}_{h,i} \subset \mathcal{X}$ be the region associated with node (h,i). By assumption, these regions are measurable and must satisfy the constraints

$$\mathcal{P}_{0,1} = \mathcal{X}\,, \tag{1}$$

$$\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h,2i}\,, \qquad \text{for all } h \geqslant 0 \text{ and } 1 \leqslant i \leqslant 2^h. \tag{2}$$

As a corollary, the regions $\mathcal{P}_{h,i}$ at any level $h \geqslant 0$ cover the space $\mathcal{X}$,

$$\mathcal{X} = \bigcup_{i=1}^{2^h} \mathcal{P}_{h,i}\,,$$

explaining the term "tree of coverings".

In the algorithm listing the recursive computation of the B-values (lines 28–33) makes a local copy of the tree; of course, this part of the algorithm could be implemented in various other ways. Other arbitrary choices in the algorithm as shown here are how tie breaking in the node selection part is done (lines 9–12), or how a point in the region associated with the selected node is chosen (line 16). We note in passing that implementing these differently would not change our theoretical results.

To facilitate the formal study of the algorithm, we shall need some more notation. In particular, we shall introduce time-indexed versions ($\mathcal{T}_n$, $(H_n, I_n), X_n, Y_n, \widehat{\mu}_{h,i}(n)$, etc.) of the quantities used by the algorithm. The convention used is that the indexation by n is used to indicate the value taken at the end of the n^{th} round.

In particular, $\mathcal{T}_n$ is used to denote the finite subtree stored by the algorithm at the end of round n. Thus, the initial tree is $\mathcal{T}_0 = \{(0,1)\}$ and it is expanded round after round as

$$\mathcal{T}_n = \mathcal{T}_{n-1} \cup \{(H_n, I_n)\}\,,$$

where (H_n, I_n) is the node selected in line 15. We call (H_n, I_n) *the node played in round n*. We use X_n to denote the point selected by HOO in the region associated with the node played in round n, while Y_n denotes the received reward.

Node selection works by comparing B-values and always choosing the node with the highest B-value. The B-value, $B_{h,i}(n)$, at node (h,i) by the end of round n is an estimated upper bound on the mean-payoff function at node (h,i). To define it we first need to introduce the average of the rewards received in rounds when some descendant of node (h,i) was chosen (by convention, each node is a descendant of itself):

$$\widehat{\mu}_{h,i}(n) = \frac{1}{T_{h,i}(n)} \sum_{t=1}^{n} Y_t \, \mathbb{I}_{\{(H_t,I_t) \in \mathcal{C}(h,i)\}}\,.$$

Here, $\mathcal{C}(h,i)$ denotes the set of all descendants of a node (h,i) in the infinite tree,

$$\mathcal{C}(h,i) = \big\{(h,i)\big\} \cup \mathcal{C}(h+1, 2i-1) \cup \mathcal{C}(h+1, 2i)\,,$$

and $T_{h,i}(n)$ is the number of times a descendant of (h,i) is played up to and including round n, that is,

$$T_{h,i}(n) = \sum_{t=1}^{n} \mathbb{I}_{\{(H_t,I_t) \in \mathcal{C}(h,i)\}}\,.$$

Algorithm 1 The HOO strategy

Parameters: Two real numbers $\nu_1 > 0$ and $\rho \in (0,1)$, a sequence $(\mathcal{P}_{h,i})_{h \geqslant 0, 1 \leqslant i \leqslant 2^h}$ of subsets of $\mathcal{X}$ satisfying the conditions (1) and (2).

Auxiliary function LEAF($\mathcal{T}$): outputs a leaf of $\mathcal{T}$.

Initialization: $\mathcal{T} = \{(0,1)\}$ and $B_{1,2} = B_{2,2} = +\infty$.

```
1: for $n = 1, 2, \ldots$ do                                  ▷ Strategy HOO in round $n \geqslant 1$
2:     $(h,i) \leftarrow (0,1)$                                ▷ Start at the root
3:     $P \leftarrow \{(h,i)\}$                                ▷ $P$ stores the path traversed in the tree
4:     while $(h,i) \in \mathcal{T}$ do                        ▷ Search the tree $\mathcal{T}$
5:         if $B_{h+1,2i-1} > B_{h+1,2i}$ then                 ▷ Select the "more promising" child
6:             $(h,i) \leftarrow (h+1, 2i-1)$
7:         else if $B_{h+1,2i-1} < B_{h+1,2i}$ then
8:             $(h,i) \leftarrow (h+1, 2i)$
9:         else                                                ▷ Tie-breaking rule
10:            $Z \sim \mathrm{Ber}(0.5)$                      ▷ e.g., choose a child at random
11:            $(h,i) \leftarrow (h+1, 2i-Z)$
12:        end if
13:        $P \leftarrow P \cup \{(h,i)\}$
14:    end while
15:    $(H,I) \leftarrow (h,i)$                                ▷ The selected node
16:    Choose arm $X$ in $\mathcal{P}_{H,I}$ and play it      ▷ Arbitrary selection of an arm
17:    Receive corresponding reward $Y$
18:    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(H,I)\}$    ▷ Extend the tree
19:    for all $(h,i) \in P$ do                                ▷ Update the statistics $T$ and $\widehat{\mu}$ stored in the path
20:        $T_{h,i} \leftarrow T_{h,i} + 1$                    ▷ Increment the counter of node $(h,i)$
21:        $\widehat{\mu}_{h,i} \leftarrow (1 - 1/T_{h,i})\widehat{\mu}_{h,i} + Y/T_{h,i}$    ▷ Update the mean $\widehat{\mu}_{h,i}$ of node $(h,i)$
22:    end for
23:    for all $(h,i) \in \mathcal{T}$ do                      ▷ Update the statistics $U$ stored in the tree
24:        $U_{h,i} \leftarrow \widehat{\mu}_{h,i} + \sqrt{(2\ln n)/T_{h,i}} + \nu_1 \rho^h$    ▷ Update the $U$-value of node $(h,i)$
25:    end for
26:    $B_{H+1,2I-1} \leftarrow +\infty$                       ▷ $B$-values of the children of the new leaf
27:    $B_{H+1,2I} \leftarrow +\infty$
28:    $\mathcal{T}' \leftarrow \mathcal{T}$                   ▷ Local copy of the current tree $\mathcal{T}$
29:    while $\mathcal{T}' \neq \{(0,1)\}$ do                  ▷ Backward computation of the $B$-values
30:        $(h,i) \leftarrow$ LEAF($\mathcal{T}'$)             ▷ Take any remaining leaf
31:        $B_{h,i} \leftarrow \min\{U_{h,i}, \max\{B_{h+1,2i-1}, B_{h+1,2i}\}\}$    ▷ Backward computation
32:        $\mathcal{T}' \leftarrow \mathcal{T}' \setminus \{(h,i)\}$    ▷ Drop updated leaf $(h,i)$
33:    end while
34: end for
```

A key quantity determining $B_{h,i}(n)$ is $U_{h,i}(n)$, an initial estimate of the maximum of the mean-payoff function in the region $\mathcal{P}_{h,i}$ associated with node (h,i):

$$U_{h,i}(n) = \begin{cases} \widehat{\mu}_{h,i}(n) + \sqrt{\dfrac{2\ln n}{T_{h,i}(n)}} + \nu_1 \rho^h, & \text{if } T_{h,i}(n) > 0; \\ +\infty, & \text{otherwise.} \end{cases} \tag{3}$$

In the expression corresponding to the case $T_{h,i}(n) > 0$, the first term added to the average of rewards accounts for the uncertainty arising from the randomness of the rewards that the average is based on, while the second term, $\nu_1 \rho^h$, accounts for the maximum possible variation of the mean-payoff function over the region $\mathcal{P}_{h,i}$. The actual bound on the maxima used in HOO is defined recursively by

$$B_{h,i}(n) = \begin{cases} \min\Big\{U_{h,i}(n), \max\big\{B_{h+1,2i-1}(n), B_{h+1,2i}(n)\big\}\Big\}, & \text{if } (h,i) \in \mathcal{T}_n; \\ +\infty, & \text{otherwise.} \end{cases}$$

The role of $B_{h,i}(n)$ is to put a tight, optimistic, high-probability upper bound on the best mean-payoff that can be achieved in the region $\mathcal{P}_{h,i}$. By assumption, $\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h+1,2i}$. Thus, assuming that $B_{h+1,2i-1}(n)$ (resp., $B_{h+1,2i}(n)$) is a valid upper bound for region $\mathcal{P}_{h+1,2i-1}$ (resp., $\mathcal{P}_{h+1,2i}$), we see that $\max\big\{B_{h+1,2i-1}(n), B_{h+1,2i}(n)\big\}$ must be a valid upper bound for region $\mathcal{P}_{h,i}$. Since $U_{h,i}(n)$ is another valid upper bound for region $\mathcal{P}_{h,i}$, we get a tighter (less overoptimistic) upper bound by taking the minimum of these bounds.

Obviously, for leafs (h,i) of the tree $\mathcal{T}_n$, one has $B_{h,i}(n) = U_{h,i}(n)$, while close to the root one may expect that $B_{h,i}(n) < U_{h,i}(n)$; that is, the upper bounds close to the root are expected to be less biased than the ones associated with nodes farther away from the root.

Note that at the beginning of round n, the algorithm uses $B_{h,i}(n-1)$ to select the node (H_n, I_n) to be played (since $B_{h,i}(n)$ will only be available at the end of round n). It does so by following a path from the root node to an inner node with only one child or a leaf and finally considering a child (H_n, I_n) of the latter; at each node of the path, the child with highest B-value is chosen, till the node (H_n, I_n) with infinite B-value is reached.

3.1 Illustrations

Figure 1 illustrates the computation done by HOO in round n, as well as the correspondence between the nodes of the tree constructed by the algorithm and their associated regions. Figure 2 shows trees built by running HOO for a specific environment.

3.2 Computational Complexity

At the end of round n, the size of the active tree $\mathcal{T}_n$ is at most n, making the storage requirements of HOO linear in n. In addition, the statistics and B-values of all nodes in the active tree need to be updated, which thus takes time $O(n)$. HOO runs in time $O(n)$ at each round n, making the algorithm's total running time up to round n quadratic in n. In Section 4.3 we modify HOO so that if the time horizon n_0 is known in advance, the total running time is $O(n_0 \ln n_0)$, while the modified algorithm will be shown to enjoy essentially the same regret bound as the original version.

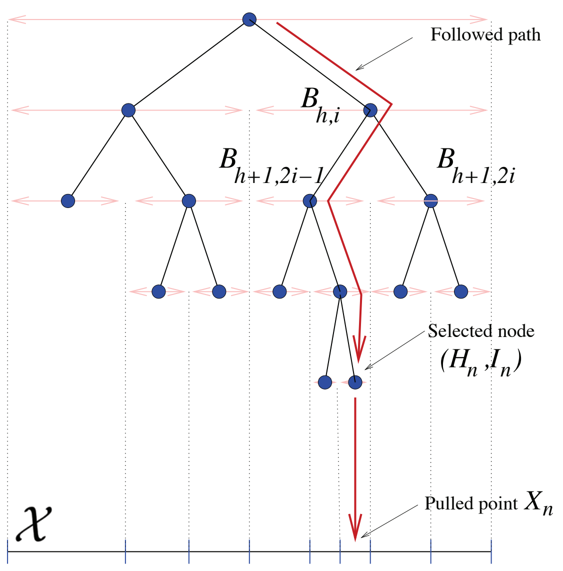


Figure 1: Illustration of the node selection procedure in round n. The tree represents $\mathcal{T}_n$. In the illustration, $B_{h+1,2i-1}(n-1) > B_{h+1,2i}(n-1)$, therefore, the selected path included the node $(h+1,2i-1)$ rather than the node $(h+1,2i)$.

4. Main Results

We start by describing and commenting on the assumptions that we need to analyze the regret of HOO. This is followed by stating the first upper bound, followed by some improvements on the basic algorithm. The section is finished by the statement of our results on the minimax optimality of HOO.

4.1 Assumptions

The main assumption will concern the "smoothness" of the mean-payoff function. However, somewhat unconventionally, we shall use a notion of smoothness that is built around dissimilarity functions rather than distances, allowing us to deal with function classes of highly different smoothness degrees in a unified manner. Before stating our smoothness assumptions, we define the notion of a dissimilarity function and some associated concepts.

Definition 2 (Dissimilarity) *A* dissimilarity ℓ *over* $\mathcal{X}$ *is a non-negative mapping* $\ell : \mathcal{X}^2 \to \mathbb{R}$ *satisfying* $\ell(x,x) = 0$ *for all* $x \in \mathcal{X}$.

Given a dissimilarity ℓ, the *diameter* of a subset A of $\mathcal{X}$ as measured by ℓ is defined by

$$\operatorname{diam}(A) = \sup_{x,y \in A} \ell(x,y),$$

while the ℓ*—open ball* of $\mathcal{X}$ with radius $\varepsilon > 0$ and center $x \in \mathcal{X}$ is defined by

$$\mathcal{B}(x,\varepsilon) = \{ y \in \mathcal{X} : \ell(x,y) < \varepsilon \} .$$

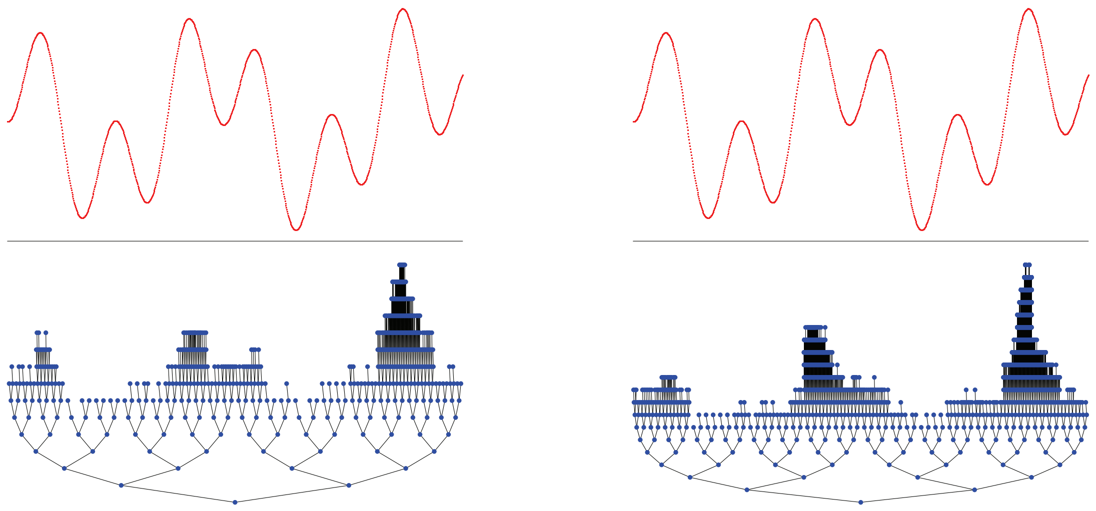


Figure 2: The trees (bottom figures) built by HOO after 1,000 (left) and 10,000 (right) rounds. The mean-payoff function (shown in the top part of the figure) is $x \in [0,1] \longmapsto 1/2\big(\sin(13x)\sin(27x)+1\big)$; the corresponding payoffs are Bernoulli-distributed. The inputs of HOO are as follows: the tree of coverings is formed by all dyadic intervals, $\nu_1 = 1$ and $\rho = 1/2$. The tie-breaking rule is to choose a child at random (as shown in the Algorithm 1), while the points in $\mathcal{X}$ to be played are chosen as the centers of the dyadic intervals. Note that the tree is extensively refined where the mean-payoff function is near-optimal, while it is much less developed in other regions.

Note that the dissimilarity ℓ is only used in the theoretical analysis of HOO; the algorithm does not require ℓ as an explicit input. However, when choosing its parameters (the tree of coverings and the real numbers $\nu_1 > 0$ and $\rho < 1$) for the (set of) two assumptions below to be satisfied, the user of the algorithm probably has in mind a given dissimilarity.

However, it is also natural to wonder what is the class of functions for which the algorithm (given a fixed tree) can achieve non-trivial regret bounds; a similar question for regression was investigated, for example, by Yang (2007). We shall indicate below how to construct a subset of such a class, right after stating our assumptions connecting the tree, the dissimilarity, and the environment (the mean-payoff function). Of these, Assumption A2 will be interpreted, discussed, and equivalently reformulated below into (5), a form that might be more intuitive. The form (4) stated below will turn out to be the most useful one in the proofs.

Assumptions Given the parameters of HOO, that is, the real numbers $\nu_1 > 0$ and $\rho \in (0,1)$ and the tree of coverings $(\mathcal{P}_{h,i})$, there exists a dissimilarity function ℓ such that the following two assumptions are satisfied.

A1. There exists $\nu_2 > 0$ such that for all integers $h \geqslant 0$,

(a) $\operatorname{diam}(\mathcal{P}_{h,i}) \leqslant \nu_1 \rho^h$ for all $i = 1, \ldots, 2^h$;

(b) for all $i = 1, \dots, 2^h$, there exists $x^{\circ}_{h,i} \in \mathcal{P}_{h,i}$ such that

$$\mathcal{B}_{h,i} \overset{\text{def}}{=} \mathcal{B}\bigl(x^{\circ}_{h,i}, \nu_2 \rho^h\bigr) \subset \mathcal{P}_{h,i} \, ;$$

(c) $\mathcal{B}_{h,i} \cap \mathcal{B}_{h,j} = \emptyset$ for all $1 \leqslant i < j \leqslant 2^h$.

A2. The mean-payoff function f satisfies that for all $x, y \in \mathcal{X}$,

$$f^* - f(y) \leqslant f^* - f(x) + \max\bigl\{ f^* - f(x), \, \ell(x,y) \bigr\} \, . \tag{4}$$

We show next how a tree induces in a natural way first a dissimilarity and then a class of environments. For this, we need to assume that the tree of coverings $(\mathcal{P}_{h,i})$—in addition to (1) and (2)—is such that the subsets $\mathcal{P}_{h,i}$ and $\mathcal{P}_{h,j}$ are disjoint whenever $1 \leqslant i < j \leqslant 2^h$ and that none of them is empty. Then, each $x \in \mathcal{X}$ corresponds to a unique path in the tree, which can be represented as an infinite binary sequence $x_0 x_1 x_2 \dots$, where

$$\begin{aligned}
x_0 &= \mathbb{I}_{\left\{x \in \mathcal{P}_{1,1+1}\right\}} \, , \\
x_1 &= \mathbb{I}_{\left\{x \in \mathcal{P}_{2,1+(2x_0+1)}\right\}} \, , \\
x_2 &= \mathbb{I}_{\left\{x \in \mathcal{P}_{3,1+(4x_0+2x_1+1)}\right\}} \, , \\
&\dots
\end{aligned}$$

For points $x, y \in \mathcal{X}$ with respective representations $x_0 x_1 \dots$ and $y_0 y_1 \dots$, we let

$$\ell(x,y) = (1-\rho)\nu_1 \sum_{h=0}^{\infty} \mathbb{I}_{\{x_h \neq y_h\}} \rho^h \, .$$

It is not hard to see that this dissimilarity satisfies A1. Thus, the associated class of environments $\mathcal{C}$ is formed by those with mean-payoff functions satisfying A2 with the so-defined dissimilarity. This is a "natural class" underlying the tree for which our tree-based algorithm can achieve non-trivial regret. (However, we do not know if this is the largest such class.)

In general, Assumption A1 ensures that the regions in the tree of coverings $(\mathcal{P}_{h,i})$ shrink exactly at a geometric rate. The following example shows how to satisfy A1 when the domain $\mathcal{X}$ is a D-dimensional hyper-rectangle and the dissimilarity is some positive power of the Euclidean (or supremum) norm.

Example 1 *Assume that $\mathcal{X}$ is a D-dimension hyper-rectangle and consider the dissimilarity $\ell(x,y) = b\|x-y\|_2^a$, where $a > 0$ and $b > 0$ are real numbers and $\|\cdot\|_2$ is the Euclidean norm. Define the tree of coverings $(\mathcal{P}_{h,i})$ in the following inductive way: let $\mathcal{P}_{0,1} = \mathcal{X}$. Given a node $\mathcal{P}_{h,i}$, let $\mathcal{P}_{h+1,2i-1}$ and $\mathcal{P}_{h+1,2i}$ be obtained from the hyper-rectangle $\mathcal{P}_{h,i}$ by splitting it in the middle along its longest side (ties can be broken arbitrarily).*

We now argue that Assumption A1 is satisfied. With no loss of generality we take $\mathcal{X} = [0,1]^D$. Then, for all integers $u \geqslant 0$ and $0 \leqslant k \leqslant D-1$,

$$\operatorname{diam}(\mathcal{P}_{uD+k,1}) = b\left(\frac{1}{2^u}\sqrt{D - \frac{3}{4}k}\right)^a \leqslant b\left(\frac{\sqrt{D}}{2^u}\right)^a \, .$$

It is now easy to see that Assumption A1 is satisfied for the indicated dissimilarity, for example, with the choice of the parameters $\rho = 2^{-a/D}$ *and* $\nu_1 = b\left(2\sqrt{D}\right)^a$ *for HOO, and the value* $\nu_2 = b/2^a$.

Example 2 *In the same setting, with the same tree of coverings* $(\mathcal{P}_{h,i})$ *over* $\mathcal{X} = [0,1]^D$*, but now with the dissimilarity* $\ell(x,y) = b\|x-y\|_\infty^a$*, we get that for all integers* $u \geqslant 0$ *and* $0 \leqslant k \leqslant D-1$,

$$\operatorname{diam}(\mathcal{P}_{uD+k,1}) = b\left(\frac{1}{2^u}\right)^a .$$

This time, Assumption A1 is satisfied, for example, with the choice of the parameters $\rho = 2^{-a/D}$ *and* $\nu_1 = b\,2^a$ *for HOO, and the value* $\nu_2 = b/2^a$.

The second assumption, A2, concerns the environment; when Assumption A2 is satisfied, we say that f is *weakly Lipschitz* with respect to (w.r.t.) ℓ. The choice of this terminology follows from the fact that if f is 1-Lipschitz w.r.t. ℓ, that is, for all $x,y \in \mathcal{X}$, one has $|f(x)-f(y)| \leqslant \ell(x,y)$, then it is also weakly Lipschitz w.r.t. ℓ.

On the other hand, weak Lipschitzness is a milder requirement. It implies local (one-sided) 1-Lipschitzness at any global maximum, since at any arm x^* such that $f(x^*) = f^*$, the criterion (4) rewrites to $f(x^*) - f(y) \leqslant \ell(x^*,y)$. In the vicinity of other arms x, the constraint is milder as the arm x gets worse (as $f^* - f(x)$ increases) since the condition (4) rewrites to

$$\forall y \in \mathcal{X}, \qquad f(x) - f(y) \leqslant \max\bigl\{f^* - f(x),\, \ell(x,y)\bigr\}. \tag{5}$$

Here is another interpretation of these two facts; it will be useful when considering local assumptions in Section 4.4 (a weaker set of assumptions). First, concerning the behavior around global maxima, Assumption A2 implies that for any set $\mathcal{A} \subset \mathcal{X}$ with $\sup_{x\in\mathcal{A}} f(x) = f^*$,

$$f^* - \inf_{x\in\mathcal{A}} f(x) \leqslant \operatorname{diam}(\mathcal{A}). \tag{6}$$

Second, it can be seen that Assumption A2 is equivalent[4] to the following property: for all $x \in \mathcal{X}$ and $\varepsilon \geqslant 0$,

$$\mathcal{B}\bigl(x, f^* - f(x) + \varepsilon\bigr) \subset \mathcal{X}_{2\left(f^*-f(x)\right)+\varepsilon} \tag{7}$$

where

$$\mathcal{X}_\varepsilon = \bigl\{x \in \mathcal{X} : f(x) \geqslant f^* - \varepsilon\bigr\}$$

denotes the set of ε*-optimal arms*. This second property essentially states that there is no sudden and large drop in the mean-payoff function around the global maxima (note that this property can be satisfied even for discontinuous functions).

Figure 3 presents an illustration of the two properties discussed above.

Before stating our main results, we provide a straightforward, though useful consequence of Assumptions A1 and A2, which should be seen as an intuitive justification for the third term in (3).

4. That Assumption A2 implies (7) is immediate; for the converse, it suffices to consider, for each $y \in \mathcal{X}$, the sequence

$$\varepsilon_n = \Bigl(\ell(x,y) - \bigl(f^* - f(x)\bigr)\Bigr)_+ + 1/n,$$

where $(\,\cdot\,)_+$ denotes the nonnegative part.

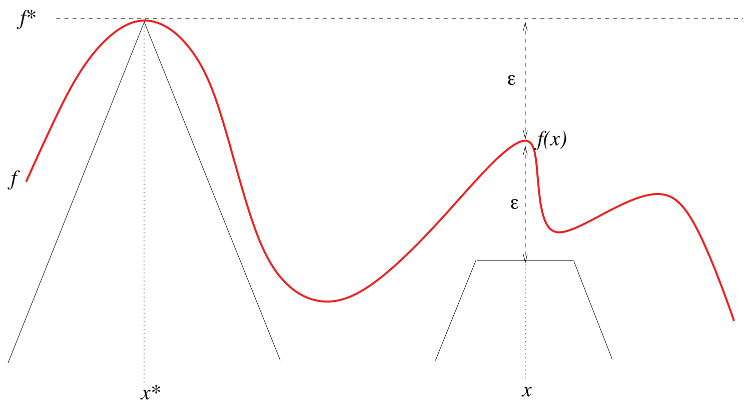


Figure 3: Illustration of the property of weak Lipschitzness (on the real line and for the distance $\ell(x,y) = |x-y|$). Around the optimum x^* the values $f(y)$ should be above $f^* - \ell(x^*,y)$. Around any ε-optimal point x the values $f(y)$ should be larger than $f^* - 2\varepsilon$ for $\ell(x,y) \leqslant \varepsilon$ and larger than $f(x) - \ell(x,y)$ elsewhere.

For all nodes (h,i), let

$$f^*_{h,i} = \sup_{x \in \mathcal{P}_{h,i}} f(x) \qquad \text{and} \qquad \Delta_{h,i} = f^* - f^*_{h,i}\,.$$

$\Delta_{h,i}$ is called the *suboptimality factor* of node (h,i). Depending whether it is positive or not, a node (h,i) is called *suboptimal* ($\Delta_{h,i} > 0$) or *optimal* ($\Delta_{h,i} = 0$).

Lemma 3 *Under Assumptions A1 and A2, if the suboptimality factor $\Delta_{h,i}$ of a region $\mathcal{P}_{h,i}$ is bounded by $c\nu_1\rho^h$ for some $c \geqslant 0$, then all arms in $\mathcal{P}_{h,i}$ are $\max\{2c, c+1\}\,\nu_1\rho^h$-optimal, that is,*

$$\mathcal{P}_{h,i} \subset \mathcal{X}_{\max\{2c,c+1\}\nu_1\rho^h}\,.$$

Proof For all $\delta > 0$, we denote by $x^*_{h,i}(\delta)$ an element of $\mathcal{P}_{h,i}$ such that

$$f\big(x^*_{h,i}(\delta)\big) \geqslant f^*_{h,i} - \delta = f^* - \Delta_{h,i} - \delta\,.$$

By the weak Lipschitz property (Assumption A2), it then follows that for all $y \in \mathcal{P}_{h,i}$,

$$\begin{aligned} f^* - f(y) &\leqslant f^* - f\big(x^*_{h,i}(\delta)\big) + \max\Big\{f^* - f\big(x^*_{h,i}(\delta)\big),\, \ell\big(x^*_{h,i}(\delta),\, y\big)\Big\} \\ &\leqslant \Delta_{h,i} + \delta + \max\big\{\Delta_{h,i} + \delta,\ \operatorname{diam} \mathcal{P}_{h,i}\big\}\,. \end{aligned}$$

Letting $\delta \to 0$ and substituting the bounds on the suboptimality and on the diameter of $\mathcal{P}_{h,i}$ (Assumption A1) concludes the proof. ■

4.2 Upper Bound for the Regret of HOO

Auer et al. (2007, Assumption 2) observed that the regret of a continuum-armed bandit algorithm should depend on how fast the volumes of the sets of ε-optimal arms shrink as $\varepsilon \to 0$. Here, we capture this by defining a new notion, the near-optimality dimension of the mean-payoff function. The connection between these concepts, as well as with the zooming dimension defined by Kleinberg et al. (2008a), will be further discussed in Section 5. We start by recalling the definition of packing numbers.

Definition 4 (Packing number) *The ε-packing number $\mathcal{N}(\mathcal{X}, \ell, \varepsilon)$ of $\mathcal{X}$ w.r.t. the dissimilarity ℓ is the size of the largest packing of $\mathcal{X}$ with disjoint ℓ-open balls of radius ε. That is, $\mathcal{N}(\mathcal{X}, \ell, \varepsilon)$ is the largest integer k such that there exists k disjoint ℓ-open balls with radius ε contained in $\mathcal{X}$.*

We now define the c–near-optimality dimension, which characterizes the size of the sets $\mathcal{X}_{c\varepsilon}$ as a function of ε. It can be seen as some growth rate in ε of the metric entropy (measured in terms of ℓ and with packing numbers rather than covering numbers) of the set of $c\varepsilon$-optimal arms.

Definition 5 (Near-optimality dimension) *For $c > 0$ the* c–near-optimality dimension *of f w.r.t. ℓ equals*

$$\max\left\{0,\ \limsup_{\varepsilon\to 0} \frac{\ln \mathcal{N}\big(\mathcal{X}_{c\varepsilon}, \ell, \varepsilon\big)}{\ln\big(\varepsilon^{-1}\big)}\right\}.$$

The following example shows that using a dissimilarity (rather than a metric, for instance) may sometimes allow for a significant reduction of the near-optimality dimension.

Example 3 *Let $\mathcal{X} = [0,1]^D$ and let $f : [0,1]^D \to [0,1]$ be defined by $f(x) = 1 - \|x\|^a$ for some $a \geqslant 1$ and some norm $\|\cdot\|$ on $\mathbb{R}^D$. Consider the dissimilarity ℓ defined by $\ell(x,y) = \|x-y\|^a$. We shall see in Example 4 that f is weakly Lipschitz w.r.t. ℓ (in a sense however slightly weaker than the one given by* (6) *and* (7) *but sufficiently strong to ensure a result similar to the one of the main result, Theorem 6 below). Here we claim that the c–near-optimality dimension (for any $c > 0$) of f w.r.t. ℓ is* 0*. On the other hand, the c–near-optimality dimension (for any $c > 0$) of f w.r.t. the dissimilarity ℓ' defined, for $0 < b < a$, by $\ell'(x,y) = \|x-y\|^b$ is $(1/b - 1/a)D > 0$. In particular, when $a > 1$ and $b = 1$, the c–near-optimality dimension is $(1-1/a)D$.*

Proof (sketch) Fix $c > 0$. The set $\mathcal{X}_{c\varepsilon}$ is the $\|\cdot\|$-ball with center 0 and radius $(c\varepsilon)^{1/a}$, that is, the ℓ-ball with center 0 and radius $c\varepsilon$. Its ε-packing number w.r.t. ℓ is bounded by a constant depending only on D, c and a; hence, the value 0 for the near-optimality dimension w.r.t. the dissimilarity ℓ.

In case of ℓ', we are interested in the packing number of the $\|\cdot\|$-ball with center 0 and radius $(c\varepsilon)^{1/a}$ w.r.t. ℓ'-balls. The latter is of the order of

$$\left(\frac{(c\varepsilon)^{1/a}}{\varepsilon^{1/b}}\right)^D = c^{D/a}\big(\varepsilon^{-1}\big)^{(1/b-1/a)D}\ ;$$

hence, the value $(1/b - 1/a)D$ for the near-optimality dimension in the case of the dissimilarity ℓ'.

Note that in all these cases the c-near-optimality dimension of f is independent of the value of c. ■

We can now state our first main result. The proof is presented in Section A.1.

Theorem 6 (Regret bound for HOO) *Consider HOO tuned with parameters such that Assumptions A1 and A2 hold for some dissimilarity ℓ. Let d be the $4\nu_1/\nu_2$–near-optimality dimension of the mean-payoff function f w.r.t. ℓ. Then, for all $d' > d$, there exists a constant γ such that for all $n \geqslant 1$,*

$$\mathbb{E}\big[R_n\big] \leqslant \gamma n^{(d'+1)/(d'+2)} \big(\ln n\big)^{1/(d'+2)}.$$

Note that if d is infinite, then the bound is vacuous. The constant γ in the theorem depends on d' and on all other parameters of HOO and of the assumptions, as well as on the bandit environment M. (The value of γ is determined in the analysis; it is in particular proportional to $\nu_2^{-d'}$.) The next section will exhibit a refined upper bound with a more explicit value of γ in terms of all these parameters.

Remark 7 *The tuning of the parameters of HOO is critical for the assumptions to be satisfied, thus to achieve a good regret; given some environment, one should select the parameters of HOO such that the near-optimality dimension of the mean-payoff function is minimized. Since the mean-payoff function is unknown to the user, this might be difficult to achieve. Thus, ideally, these parameters should be selected adaptively based on the observation of some preliminary sample. For now, the investigation of this possibility is left for future work.*

4.3 Improving the Running Time when the Time Horizon is Known

A deficiency of the basic HOO algorithm is that its computational complexity scales quadratically with the number of time steps. In this section we propose a simple modification to HOO that achieves essentially the same regret as HOO and whose computational complexity scales only log-linearly with the number of time steps. The needed amount of memory is still linear. We work out the case when the time horizon, n_0, is known in advance. The case of unknown horizon can be dealt with by resorting to the so-called doubling trick, see, for example, Cesa-Bianchi and Lugosi (2006, Section 2.3), which consists of periodically restarting the algorithm for regimes of lengths that double at each such fresh start, so that the r^{th} instance of the algorithm runs for 2^r rounds.

We consider two modifications to the algorithm described in Section 3. First, the quantities $U_{h,i}(n)$ of (3) are redefined by replacing the factor $\ln n$ by $\ln n_0$, that is, now

$$U_{h,i}(n) = \widehat{\mu}_{h,i}(n) + \sqrt{\frac{2\ln n_0}{T_{h,i}(n)}} + \nu_1\rho^h .$$

(This results in a policy which explores the arms with a slightly increased frequency.) The definition of the B-values in terms of the $U_{h,i}(n)$ is unchanged. A pleasant consequence of the above modification is that the B-value of a given node changes only when this node is part of a path selected by the algorithm. Thus at each round n, only the nodes along the chosen path need to be updated according to the obtained reward.

However, and this is the reason for the second modification, in the basic algorithm, a path at round n may be of length linear in n (because the tree could have a depth linear in n). This is why we also truncate the trees $\mathcal{T}_n$ at a depth D_{n_0} of the order of $\ln n_0$. More precisely, the algorithm now selects the node (H_n, I_n) to pull at round n by following a path in the tree $\mathcal{T}_{n-1}$, starting from the root and choosing at each node the child with the highest B-value (with the new definition above using $\ln n_0$), and stopping either when it encounters a node which has not been expanded before or a node at depth equal to

$$D_{n_0} = \left\lceil \frac{(\ln n_0)/2 - \ln(1/\nu_1)}{\ln(1/\rho)} \right\rceil .$$

(It is assumed that $n_0 > 1/\nu_1^2$ so that $D_{n_0} \geqslant 1$.) Note that since no child of a node (D_{n_0}, i) located at depth D_{n_0} will ever be explored, its B-value at round $n \leqslant n_0$ simply equals $U_{D_{n_0},i}(n)$.

We call this modified version of HOO the *truncated HOO* algorithm. The computational complexity of updating all B-values at each round n is of the order of D_{n_0} and thus of the order of $\ln n_0$. The total computational complexity up to round n_0 is therefore of the order of $n_0 \ln n_0$, as claimed in the introduction of this section.

As the next theorem indicates this new procedure enjoys almost the same cumulative regret bound as the basic HOO algorithm.

Theorem 8 (Upper bound on the regret of truncated HOO) *Fix a horizon n_0 such that $D_{n_0} \geqslant 1$. Then, the regret bound of Theorem 6 still holds true at round n_0 for truncated HOO up to an additional additive $4\sqrt{n_0}$ factor.*

4.4 Local Assumptions

In this section we further relax the weak Lipschitz assumption and require it only to hold locally around the maxima. Doing so, we will be able to deal with an even larger class of functions and in fact we will show that the algorithm studied in this section achieves a $O(\sqrt{n})$ bound on the regret regret when it is used for functions that are smooth around their maxima (e.g., equivalent to $\|x - x^*\|^\alpha$ for some known smoothness degree $\alpha > 0$).

For the sake of simplicity and to derive exact constants we also state in a more explicit way the assumption on the near-optimality dimension. We then propose a simple and efficient adaptation of the HOO algorithm suited for this context.

4.4.1 MODIFIED SET OF ASSUMPTIONS

Assumptions Given the parameters of (the adaption of) HOO, that is, the real numbers $\nu_1 > 0$ and $\rho \in (0,1)$ and the tree of coverings $(\mathcal{P}_{h,i})$, there exists a dissimilarity function ℓ such that Assumption A1 (for some $\nu_2 > 0$) as well as the following two assumptions hold.

A2'. There exists $\varepsilon_0 > 0$ such that for all optimal subsets $\mathcal{A} \subset \mathcal{X}$ (i.e., $\sup_{x \in \mathcal{A}} f(x) = f^*$) with diameter $\operatorname{diam}(\mathcal{A}) \leqslant \varepsilon_0$,

$$f^* - \inf_{x \in \mathcal{A}} f(x) \leqslant \operatorname{diam}(\mathcal{A}) .$$

Further, there exists $L > 0$ such that for all $x \in \mathcal{X}_{\varepsilon_0}$ and $\varepsilon \in [0, \varepsilon_0]$,

$$\mathcal{B}\big(x,\ f^* - f(x) + \varepsilon\big) \subset \mathcal{X}_{L\left(2(f^* - f(x)) + \varepsilon\right)} .$$

A3. There exist $C > 0$ and $d > 0$ such that for all $\varepsilon \leqslant \varepsilon_0$,

$$\mathcal{N}\big(\mathcal{X}_{c\varepsilon}, \ell, \varepsilon\big) \leqslant C\varepsilon^{-d},$$

where $c = 4L\nu_1/\nu_2$.

When f satisfies Assumption A2', we say that f is ε_0—locally L—*weakly Lipschitz* w.r.t. ℓ. Note that this assumption was obtained by weakening the characterizations (6) and (7) of weak Lipschitzness.

Assumption A3 is not a real assumption but merely a reformulation of the definition of near optimality (with the small added ingredient that the limit can be achieved, see the second step of the proof of Theorem 6 in Section A.1).

Example 4 *We consider again the domain $\mathcal{X}$ and function f studied in Example 3 and prove (as announced beforehand) that f is ε_0—locally 2^{a-1}—weakly Lipschitz w.r.t. the dissimilarity ℓ defined by $\ell(x,y) = \|x-y\|^a$; which, in fact, holds for all ε_0.*

Proof Note that $x^* = (0,\ldots,0)$ is such that $f^* = 1 = f(x^*)$. Therefore, for all $x \in \mathcal{X}$,

$$f^* - f(x) = \|x\|^a = \ell(x^*,x),$$

which yields the first part of Assumption A2'. To prove that the second part is true for $L = 2^{a-1}$ and with no constraint on the considered ε, we first note that since $a \geqslant 1$, it holds by convexity that $(u+v)^a \leqslant 2^{a-1}(u^a+v^a)$ for all $u,v \geqslant 0$. Now, for all $\varepsilon \geqslant 0$ and $y \in \mathcal{B}\big(x, \|x\|^a+\varepsilon\big)$, that is, y such that $\ell(x,y) = \|x-y\|^a \leqslant \|x\|^a + \varepsilon$,

$$f^* - f(y) = \|y\|^a \leqslant \big(\|x\| + \|x-y\|\big)^a \leqslant 2^{a-1}\big(\|x\|^a + \|x-y\|^a\big) \leqslant 2^{a-1}\big(2\|x\|^a + \varepsilon\big),$$

which concludes the proof of the second part of A2'. ■

4.4.2 MODIFIED HOO ALGORITHM

We now describe the proposed modifications to the basic HOO algorithm.

We first consider, as a building block, the algorithm called *z-HOO*, which takes an integer z as an additional parameter to those of HOO. Algorithm z-HOO works as follows: it never plays any node with depth smaller or equal to $z-1$ and starts directly the selection of a new node at depth z. To do so, it first picks the node at depth z with the best B-value, chooses a path and then proceeds as the basic HOO algorithm. Note in particular that the initialization of this algorithm consists (in the first 2^z rounds) in playing once each of the 2^z nodes located at depth z in the tree (since by definition a node that has not been played yet has a B-value equal to $+\infty$). We note in passing that when $z = 0$, algorithm z-HOO coincides with the basic HOO algorithm.

Algorithm *local-HOO* employs the doubling trick in conjunction with consecutive instances of z-HOO. It works as follows. The integers $r \geqslant 1$ will index different regimes. The r^{th} regime starts at round $2^r - 1$ and ends when the next regime starts; it thus lasts for 2^r rounds. At the beginning of regime r, a fresh copy of z_r-HOO, where $z_r = \lceil \log_2 r \rceil$, is initialized and is then used throughout the regime.

Note that each fresh start needs to pull each of the 2^{z_r} nodes located at depth z_r at least once (the number of these nodes is $\approx r$). However, since round r lasts for 2^r time steps (which is exponentially larger than the number of nodes to explore), the time spent on the initialization of z_r-HOO in any regime r is greatly outnumbered by the time spent in the rest of the regime.

In the rest of this section, we propose first an upper bound on the regret of z-HOO (with exact and explicit constants). This result will play a key role in proving a bound on the performance of local-HOO.

4.4.3 Adaptation of the Regret Bound

In the following we write h_0 for the smallest integer such that

$$2\nu_1\rho^{h_0} < \varepsilon_0$$

and consider the algorithm z-HOO, where $z \geqslant h_0$. In particular, when $z = 0$ is chosen, the obtained bound is the same as the one of Theorem 6, except that the constants are given in analytic forms.

Theorem 9 (Regret bound for z-HOO) *Consider z-HOO tuned with parameters ν_1 and ρ such that Assumptions A1, A2' and A3 hold for some dissimilarity ℓ and the values $\nu_2, L, \varepsilon_0, C, d$. If, in addition, $z \geqslant h_0$ and $n \geqslant 2$ is large enough so that*

$$z \leqslant \frac{1}{d+2} \frac{\ln(4L\nu_1 n) - \ln(\gamma \ln n)}{\ln(1/\rho)},$$

where

$$\gamma = \frac{4CL\nu_1\nu_2^{-d}}{(1/\rho)^{d+1} - 1}\left(\frac{16}{\nu_1^2\rho^2} + 9\right),$$

then the following bound holds for the expected regret of z-HOO:

$$\mathbb{E}\big[R_n\big] \leqslant \left(1 + \frac{1}{\rho^{d+2}}\right)\big(4L\nu_1 n\big)^{(d+1)/(d+2)}(\gamma \ln n)^{1/(d+2)} + \big(2^z - 1\big)\left(\frac{8\ln n}{\nu_1^2\rho^{2z}} + 4\right).$$

The proof, which is a modification of the proof to Theorem 6, can be found in Section A.3 of the Appendix. The main complication arises because the weakened assumptions do not allow one to reason about the smoothness at an arbitrary scale; this is essentially due to the threshold ε_0 used in the formulation of the assumptions. This is why in the proposed variant of HOO we discard nodes located too close to the root (at depth smaller than $h_0 - 1$). Note that in the bound the second term arises from playing in regions corresponding to the descendants of "poor" nodes located at level z. In particular, this term disappears when $z = 0$, in which case we get a bound on the regret of HOO provided that $2\nu_1 < \varepsilon_0$ holds.

Example 5 *We consider again the setting of Examples 2, 3, and 4. The domain is $\mathcal{X} = [0,1]^D$ and the mean-payoff function f is defined by $f(x) = 1 - \|x\|_\infty^2$. We assume that HOO is run with parameters $\rho = (1/4)^{1/D}$ and $\nu_1 = 4$. We already proved that Assumptions A1, A2' and A3 are satisfied with the dissimilarity $\ell(x,y) = \|x-y\|_\infty^2$, the constants $\nu_2 = 1/4$, $L = 2$, $d = 0$, and*[5] $C =$

5. To compute C, one can first note that $4L\nu_1/\nu_2 = 128$; the question at hand for Assumption A3 to be satisfied is therefore to upper bound the number of balls of radius $\sqrt{\varepsilon}$ (w.r.t. the supremum norm $\|\cdot\|_\infty$) that can be packed in a ball of radius $\sqrt{128\varepsilon}$, giving rise to the bound $C \leqslant \sqrt{128}^D$.

$128^{D/2}$, *as well as any* $\varepsilon_0 > 0$ *(that is, with* $h_0 = 0$*). Thus, resorting to Theorem 9 (applied with* $z = 0$*), we obtain*

$$\gamma = \frac{32 \times 128^{D/2}}{4^{1/D} - 1}\left(4^{2/D} + 9\right)$$

and get

$$\mathbb{E}\left[R_n\right] \leqslant \left(1 + 4^{2/D}\right)\sqrt{32\gamma n \ln n} = \sqrt{\exp\big(O(D)\big)\, n \ln n}\,.$$

Under the prescribed assumptions, the rate of convergence is of order $\sqrt{n}$ *no matter the ambient dimension* D*. Although the rate is independent of* D*, the latter impacts the performance through the multiplicative factor in front of the rate, which is exponential in* D*.*

The following theorem is an almost straightforward consequence of Theorem 9 (the detailed proof can be found in Section A.4 of the Appendix). Note that local-HOO does not require the knowledge of the parameter ε_0 in A2'.

Theorem 10 (Regret bound for local-HOO) *Consider local-HOO and assume that its parameters are tuned such that Assumptions A1, A2' and A3 hold for some dissimilarity* ℓ*. Then the expected regret of local-HOO is bounded (in a distribution-dependent sense) as follows,*

$$\mathbb{E}\left[R_n\right] = \widetilde{O}\left(n^{(d+1)/(d+2)}\right).$$

4.5 Minimax Optimality in Metric Spaces

In this section we provide two theorems showing the minimax optimality of HOO in metric spaces. The notion of packing dimension is key.

Definition 11 (Packing dimension) *The* ℓ*-packing dimension of a set* $\mathcal{X}$ *(w.r.t. a dissimilarity* ℓ*) is defined as*

$$\limsup_{\varepsilon \to 0} \frac{\ln \mathcal{N}(\mathcal{X}, \ell, \varepsilon)}{\ln(\varepsilon^{-1})}\,.$$

For instance, it is easy to see that whenever ℓ is a norm, compact subsets of $\mathbb{R}^D$ with non-empty interiors have a packing dimension of D. We note in passing that the packing dimension provides a bound on the near-optimality dimension that only depends on $\mathcal{X}$ and ℓ but not on the underlying mean-payoff function.

Let $\mathcal{F}_{\mathcal{X},\ell}$ be the class of all bandit environments on $\mathcal{X}$ with a weak Lipschitz mean-payoff function (i.e., satisfying Assumption A2). For the sake of clarity, we now denote, for a bandit strategy φ and a bandit environment M on $\mathcal{X}$, the expectation of the cumulative regret of φ over M at time n by $\mathbb{E}_M\left[R_n(\varphi)\right]$.

The following theorem provides a uniform upper bound on the regret of HOO over this class of environments. It is a corollary of Theorem 9; most of the efforts in the proof consist of showing that the distribution-dependent constant γ in the statement of Theorem 9 can be upper bounded by a quantity (the γ in the statement below) that only depends on $\mathcal{X}, \nu_1, \rho, \ell, \nu_2, D'$, but not on the underlying mean-payoff functions. The proof is provided in Section A.5 of the Appendix.

Theorem 12 (Uniform upper bound on the regret of HOO) *Assume that $\mathcal{X}$ has a finite ℓ-packing dimension D and that the parameters of HOO are such that A1 is satisfied. Then, for all $D' > D$ there exists a constant γ such that for all $n \geqslant 1$,*

$$\sup_{M \in \mathcal{F}_{\mathcal{X},\ell}} \mathbb{E}_M\big[R_n(\mathrm{HOO})\big] \leqslant \gamma\, n^{(D'+1)/(D'+2)} \big(\ln n\big)^{1/(D'+2)} .$$

The next result shows that in the case of metric spaces this upper bound is optimal up to a multiplicative logarithmic factor. Similar lower bounds appeared in Kleinberg (2004) (for $D = 1$) and in Kleinberg et al. (2008a). We propose here a weaker statement that suits our needs. Note that if $\mathcal{X}$ is a large enough compact subset of $\mathbb{R}^D$ with non-empty interior and the dissimilarity ℓ is some norm of $\mathbb{R}^D$, then the assumption of the following theorem is satisfied.

Theorem 13 (Uniform lower bound) *Consider a set $\mathcal{X}$ equipped with a dissimilarity ℓ that is a metric. Assume that there exists some constant $c \in (0,1]$ such that for all $\varepsilon \leqslant 1$, the packing numbers satisfy $\mathcal{N}(\mathcal{X},\ell,\varepsilon) \geqslant c\,\varepsilon^{-D} \geqslant 2$. Then, there exist two constants $N(c,D)$ and $\gamma(c,D)$ depending only on c and D such that for all bandit strategies φ and all $n \geqslant N(c,D)$,*

$$\sup_{M \in \mathcal{F}_{\mathcal{X},\ell}} \mathbb{E}_M\big[R_n(\varphi)\big] \geqslant \gamma(c,D)\, n^{(D+1)/(D+2)} .$$

The reader interested in the explicit expressions of $N(c,D)$ and $\gamma(c,D)$ is referred to the last lines of the proof of the theorem in the Appendix.

5. Discussion

In this section we would like to shed some light on the results of the previous sections. In particular we generalize the situation of Example 5, discuss the regret that we can obtain, and compare it with what could be obtained by previous works.

5.1 Examples of Regret Bounds for Functions Locally Smooth at their Maxima

We equip $\mathcal{X} = [0,1]^D$ with a norm $\|\cdot\|$. We assume that the mean-payoff function f has a finite number of global maxima and that it is locally equivalent to the function $\|x - x^*\|^\alpha$—with degree $\alpha \in [0,\infty)$—around each such global maximum x^* of f; that is,

$$f(x^*) - f(x) = \Theta\big(\|x - x^*\|^\alpha\big) \qquad \text{as} \quad x \to x^*.$$

This means that there exist $c_1, c_2, \delta > 0$ such that for all x satisfying $\|x - x^*\| \leqslant \delta$,

$$c_2 \|x - x^*\|^\alpha \leqslant f(x^*) - f(x) \leqslant c_1 \|x - x^*\|^\alpha .$$

In particular, one can check that Assumption A2' is satisfied for the dissimilarity defined by $\ell_{c,\beta}(x,y) = c\|x-y\|^\beta$, where $\beta \leqslant \alpha$ (and $c \geqslant c_1$ when $\beta = \alpha$). We further assume that HOO is run with parameters ν_1 and ρ and a tree of dyadic partitions such that Assumption A1 is satisfied as well (see Examples 1 and 2 for explicit values of these parameters in the case of the Euclidean or the supremum norms over the unit cube). The following statements can then be formulated on the expected regret of HOO.

- **Known smoothness:** If we know the true smoothness of f around its maxima, then we set $\beta = \alpha$ and $c \geqslant c_1$. This choice $\ell_{c_1,\alpha}$ of a dissimilarity is such that f is locally weak-Lipschitz with respect to it and the near-optimality dimension is $d = 0$ (cf., Example 3). Theorem 10 thus implies that the expected regret of local-HOO is $\widetilde{O}(\sqrt{n})$, that is, *the rate of the bound is independent of the dimension D*.

- **Smoothness underestimated:** Here, we assume that the true smoothness of f around its maxima is unknown and that it is underestimated by choosing $\beta < \alpha$ (and some c). Then f is still locally weak-Lipschitz with respect to the dissimilarity $\ell_{c,\beta}$ and the near-optimality dimension is $d = D(1/\beta - 1/\alpha)$, as shown in Example 3; the regret of HOO is $\widetilde{O}\big(n^{(d+1)/(d+2)}\big)$.

- **Smoothness overestimated:** Now, if the true smoothness is overestimated by choosing $\beta > \alpha$ or $\alpha = \beta$ and $c < c_1$, then the assumption of weak Lipschitzness is violated and we are unable to provide any guarantee on the behavior of HOO. The latter, when used with an overestimated smoothness parameter, may lack exploration and exploit too heavily from the beginning. As a consequence, it may get stuck in some local optimum of f, missing the global one(s) for a very long time (possibly indefinitely). Such a behavior is illustrated in the example provided in Coquelin and Munos (2007) and showing the possible problematic behavior of the closely related algorithm UCT of Kocsis and Szepesvari (2006). UCT is an example of an algorithm overestimating the smoothness of the function; this is because the B-values of UCT are defined similarly to the ones of the HOO algorithm but without the third term in the definition (3) of the U-values. This corresponds to an assumed infinite degree of smoothness (that is, to a locally constant mean-payoff function).

5.2 Relation to Previous Works

Several works (Agrawal, 1995b; Kleinberg, 2004; Cope, 2009; Auer et al., 2007; Kleinberg et al., 2008a) have considered continuum-armed bandits in Euclidean or, more generally, normed or metric spaces and provided upper and lower bounds on the regret for given classes of environments.

- Cope (2009) derived a $\widetilde{O}(\sqrt{n})$ bound on the regret for compact and convex subsets of $\mathbb{R}^d$ and mean-payoff functions with a unique minimum and second-order smoothness.

- Kleinberg (2004) considered mean-payoff functions f on the real line that are Hölder continuous with degree $0 < \alpha \leqslant 1$. The derived regret bound is $\Theta\big(n^{(\alpha+1)/(\alpha+2)}\big)$.

- Auer et al. (2007) extended the analysis to classes of functions that are equivalent to $\|x - x^*\|^\alpha$ around their maxima x^*, where the allowed smoothness degree is also larger: $\alpha \in [0, \infty)$. They derived the regret bound
$$\Theta\left(n^{\frac{1+\alpha-\alpha\beta}{1+2\alpha-\alpha\beta}}\right),$$
where the parameter β is such that the Lebesgue measure of ε-optimal arm is $O(\varepsilon^\beta)$.

- Another setting is the one of Kleinberg et al. (2008a) and Kleinberg et al. (2008b), who considered a space $(\mathcal{X}, \ell)$ equipped with some dissimilarity ℓ and assumed that f is Lipschitz w.r.t. ℓ at some maximum x^* (when the latter exists and a relaxed condition otherwise), that is,
$$\forall x \in \mathcal{X}, \qquad f(x^*) - f(x) \leqslant \ell(x, x^*)\,. \tag{8}$$

> The obtained regret bound is $\widetilde{O}\big(n^{(d+1)/(d+2)}\big)$, where d is the *zooming dimension*. The latter is defined similarly to our near-optimality dimension with the exceptions that in the definition of zooming dimension *(i)* covering numbers instead of packing numbers are used and *(ii)* sets of the form $\mathcal{X}_\varepsilon \setminus \mathcal{X}_{\varepsilon/2}$ are considered instead of the set $\mathcal{X}_{c\varepsilon}$. When $(\mathcal{X},\ell)$ is a metric space, covering and packing numbers are within a constant factor to each other, and therefore, one may prove that the zooming and near-optimality dimensions are also equal.

For an illustration, consider again the example of Section 5.1. The result of Auer et al. (2007) shows that for $D = 1$, the regret is $\Theta(\sqrt{n})$ (since here $\beta = 1/\alpha$, with the notation above). Our result extends the $\sqrt{n}$ rate of the regret bound to any dimension D.

On the other hand the analysis of Kleinberg et al. (2008b) does not apply because in this example $f(x^*) - f(x)$ is controlled only when x is close in some sense to x^* (i.e., when $\|x - x^*\| \leqslant \delta$), while (8) requires such a control over the whole set $\mathcal{X}$. However, note that the local weak-Lipschitz assumption A2' requires an extra condition in the vicinity of x^* compared to (8) as it is based on the notion of weak Lipschitzness. Thus, A2' and (8) are in general incomparable (both capture a different phenomenon at the maxima).

We now compare our results to those of Kleinberg et al. (2008a) and Kleinberg et al. (2008b) under Assumption A2 (which does not cover the example of Section 5.1 unless δ is large). Under this assumption, our algorithms enjoy essentially the same theoretical guarantees as the zooming algorithm of Kleinberg et al. (2008a,b). Further, the following hold.

- Our algorithms do not require the oracle needed by the zooming algorithm.
- Our truncated HOO algorithm achieves a computational complexity of order $O(n \log n)$, whereas the complexity of a naive implementation of the zooming algorithm is likely to be much larger.[6]
- Both truncated HOO and the zooming algorithms use the doubling trick. The basic HOO algorithm, however, avoids the doubling trick, while meeting the computational complexity of the zooming algorithm.

The fact that the doubling trick can be avoided is good news since an algorithm that uses the doubling trick must start from *tabula rasa* time to time, which results in predictable, yet inevitable, sharp performance drops—a quite unpleasant property. In particular, for this reason algorithms that rely on the doubling trick are often neglected by practitioners. In addition, the fact that we avoid the oracle needed by the zooming algorithm is attractive as this oracle might be difficult to implement for general (non-metric) dissimilarities.

Acknowledgments

We thank one of the anonymous referee for his valuable comments, which helped us to provide a fair and detailed comparison of our work to prior contributions.

This work was supported in part by French National Research Agency (ANR, project EXPLO-RA, ANR-08-COSI-004), the Alberta Ingenuity Centre of Machine Learning, Alberta Innovates

6. The zooming algorithm requires a covering oracle that is able to return a point which is not covered by the set of active strategies, if there exists one. Thus a straightforward implementation of this covering oracle might be computationally expensive in (general) continuous spaces and would require a 'global' search over the whole space.

Technology Futures (formerly iCore and AIF), NSERC and the PASCAL2 Network of Excellence under EC grant no. 216886.

Appendix A. Proofs

We provide here the proofs of the results stated above.

A.1 Proof of Theorem 6 (Main Upper Bound on the Regret of HOO)

We begin with three lemmas. The proofs of Lemmas 15 and 16 rely on concentration-of-measure techniques, while the one of Lemma 14 follows from a simple case study. Let us fix some path $(0,1), (1,i_1^*), (2,i_2^*), \ldots$ of optimal nodes, starting from the root. That is, denoting $i_0^* = 1$, we mean that for all $j \geqslant 1$, the suboptimality of (j,i_j^*) equals $\Delta_{j,i_j^*} = 0$ and (j,i_j^*) is a child of $(j-1,i_{j-1}^*)$.

Lemma 14 *Let (h,i) be a suboptimal node. Let $0 \leqslant k \leqslant h-1$ be the largest depth such that (k,i_k^*) is on the path from the root $(0,1)$ to (h,i). Then for all integers $u \geqslant 0$, we have*

$$\mathbb{E}\big[T_{h,i}(n)\big] \leqslant u + \sum_{t=u+1}^{n} \mathbb{P}\Big\{ \big[U_{s,i_s^*}(t) \leqslant f^* \text{ for some } s \in \{k+1,\ldots,t-1\}\big] \\ \text{or } \big[T_{h,i}(t) > u \text{ and } U_{h,i}(t) > f^*\big] \Big\}.$$

Proof Consider a given round $t \in \{1,\ldots,n\}$. If $(H_t,I_t) \in \mathcal{C}(h,i)$, then this is because the child $(k+1,i')$ of (k,i_k^*) on the path to (h,i) had a better B-value than its brother $(k+1,i_{k+1}^*)$. Since by definition, B-values can only increase on a chosen path, this entails that $B_{k+1,i_{k+1}^*} \leqslant B_{k+1,i'}(t) \leqslant B_{h,i}(t)$. This is turns implies, again by definition of the B-values, that $B_{k+1,i_{k+1}^*}(t) \leqslant U_{h,i}(t)$. Thus,

$$\big\{(H_t,I_t) \in \mathcal{C}(h,i)\big\} \subset \big\{U_{h,i}(t) \geqslant B_{k+1,i_{k+1}^*}(t)\big\} \subset \big\{U_{h,i}(t) > f^*\big\} \cup \big\{B_{k+1,i_{k+1}^*}(t) \leqslant f^*\big\}.$$

But, once again by definition of B-values,

$$\big\{B_{k+1,i_{k+1}^*}(t) \leqslant f^*\big\} \subset \big\{U_{k+1,i_{k+1}^*}(t) \leqslant f^*\big\} \cup \big\{B_{k+2,i_{k+2}^*}(t) \leqslant f^*\big\},$$

and the argument can be iterated. Since up to round t no more than t nodes have been played (including the suboptimal node (h,i)), we know that (t,i_t^*) has not been played so far and thus has a B-value equal to $+\infty$. (Some of the previous optimal nodes could also have had an infinite U-value, if not played so far.) We thus have proved the inclusion

$$\big\{(H_t,I_t) \in \mathcal{C}(h,i)\big\} \subset \big\{U_{h,i}(t) > f^*\big\} \cup \Big(\big\{U_{k+1,i_{k+1}^*}(t) \leqslant f^*\big\} \cup \ldots \cup \big\{U_{t-1,i_{t-1}^*}(t) \leqslant f^*\big\}\Big). \tag{9}$$

Now, for any integer $u \geqslant 0$ it holds that

$$\begin{aligned} T_{h,i}(n) &= \sum_{t=1}^{n} \mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i),\, T_{h,i}(t)\leqslant u\}} + \sum_{t=1}^{n} \mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i),\, T_{h,i}(t)>u\}} \\ &\leqslant u + \sum_{t=u+1}^{n} \mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i),\, T_{h,i}(t)>u\}}, \end{aligned}$$

where we used for the inequality the fact that the quantities $T_{h,i}(t)$ are constant from t to $t+1$, except when $(H_t,I_t)\in\mathcal{C}(h,i)$, in which case, they increase by 1; therefore, on the one hand, at most u of the $T_{h,i}(t)$ can be smaller than u and on the other hand, $T_{h,i}(t)>u$ can only happen if $t>u$. Using (9) and then taking expectations yields the result. ■

Lemma 15 *Let Assumptions A1 and A2 hold. Then, for all optimal nodes (h,i) and for all integers $n\geqslant 1$,*

$$\mathbb{P}\big\{U_{h,i}(n)\leqslant f^*\big\}\leqslant n^{-3}.$$

Proof On the event that (h,i) was not played during the first n rounds, one has, by convention, $U_{h,i}(n)=+\infty$. In the sequel, we therefore restrict our attention to the event $\big\{T_{h,i}(n)\geqslant 1\big\}$.

Lemma 3 with $c=0$ ensures that $f^*-f(x)\leqslant\nu_1\rho^h$ for all arms $x\in\mathcal{P}_{h,i}$. Hence,

$$\sum_{t=1}^{n}\big(f(X_t)+\nu_1\rho^h-f^*\big)\,\mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i)\}}\geqslant 0$$

and therefore,

$$\begin{aligned}
&\mathbb{P}\big\{U_{h,i}(n)\leqslant f^*\quad\text{and}\quad T_{h,i}(n)\geqslant 1\big\}\\
&=\;\mathbb{P}\left\{\widehat{\mu}_{h,i}(n)+\sqrt{\frac{2\ln n}{T_{h,i}(n)}}+\nu_1\rho^h\leqslant f^*\quad\text{and}\quad T_{h,i}(n)\geqslant 1\right\}\\
&=\;\mathbb{P}\left\{T_{h,i}(n)\,\widehat{\mu}_{h,i}(n)+T_{h,i}(n)\,\big(\nu_1\rho^h-f^*\big)\leqslant-\sqrt{2\,T_{h,i}(n)\ln n}\quad\text{and}\quad T_{h,i}(n)\geqslant 1\right\}\\
&=\;\mathbb{P}\Bigg\{\sum_{t=1}^{n}\big(Y_t-f(X_t)\big)\mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i)\}}+\sum_{t=1}^{n}\big(f(X_t)+\nu_1\rho^h-f^*\big)\mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i)\}}\\
&\qquad\qquad\leqslant-\sqrt{2\,T_{h,i}(n)\ln n}\quad\text{and}\quad T_{h,i}(n)\geqslant 1\Bigg\}\\
&\leqslant\;\mathbb{P}\left\{\sum_{t=1}^{n}\big(f(X_t)-Y_t\big)\mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i)\}}\geqslant\sqrt{2\,T_{h,i}(n)\ln n}\quad\text{and}\quad T_{h,i}(n)\geqslant 1\right\}.
\end{aligned}$$

We take care of the last term with a union bound and the Hoeffding-Azuma inequality for martingale differences.

To do this in a rigorous manner, we need to define a sequence of (random) stopping times when arms in $\mathcal{C}(h,i)$ were pulled:

$$T_j=\min\big\{t:\;\;T_{h,i}(t)=j\big\},\quad j=1,2,\ldots.$$

Note that $1 \leqslant T_1 < T_2 < \ldots$, hence it holds that $T_j \geqslant j$. We denote by $\widetilde{X}_j = X_{T_j}$ the j^{th} arm pulled in the region corresponding to $\mathcal{C}(h,i)$. Its associated corresponding reward equals $\widetilde{Y}_j = Y_{T_j}$ and

$$\begin{aligned}
&\mathbb{P}\left\{\sum_{t=1}^{n}\big(f(X_t)-Y_t\big)\mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i)\}} \geqslant \sqrt{2\,T_{h,i}(n)\ln n} \ \text{ and } \ T_{h,i}(n)\geqslant 1\right\} \\
= \ &\mathbb{P}\left\{\sum_{j=1}^{T_{h,i}(n)}\Big(f\big(\widetilde{X}_j\big)-\widetilde{Y}_j\Big) \geqslant \sqrt{2\,T_{h,i}(n)\ln n} \ \text{ and } \ T_{h,i}(n)\geqslant 1\right\} \\
\leqslant \ &\sum_{t=1}^{n}\mathbb{P}\left\{\sum_{j=1}^{t}\Big(f\big(\widetilde{X}_j\big)-\widetilde{Y}_j\Big) \geqslant \sqrt{2t\ln n}\right\},
\end{aligned}$$

where we used a union bound to get the last inequality.

We claim that

$$Z_t = \sum_{j=1}^{t}\Big(f\big(\widetilde{X}_j\big)-\widetilde{Y}_j\Big)$$

is a martingale w.r.t. the filtration $\mathcal{G}_t = \sigma\big(\widetilde{X}_1, Z_1, \ldots, \widetilde{X}_t, Z_t, \widetilde{X}_{t+1}\big)$. This follows, via optional skipping (see Doob, 1953, Chapter VII, adaptation of Theorem 2.3), from the facts that

$$\sum_{t=1}^{n}\big(f(X_t)-Y_t\big)\mathbb{I}_{\{(H_t,I_t)\in\mathcal{C}(h,i)\}}$$

is a martingale w.r.t. the filtration $\mathcal{F}_t = \sigma(X_1, Y_1, \ldots, X_t, Y_t, X_{t+1})$ and that the events $\{T_j = k\}$ are $\mathcal{F}_{k-1}$-measurable for all $k \geqslant j$.

Applying the Hoeffding-Azuma inequality for martingale differences (see Hoeffding, 1963), using the boundedness of the ranges of the induced martingale difference sequence, we then get, for each $t \geqslant 1$,

$$\mathbb{P}\left\{\sum_{j=1}^{t}\Big(f\big(\widetilde{X}_j\big)-\widetilde{Y}_j\Big) \geqslant \sqrt{2t\ln n}\right\} \leqslant \exp\left(-\frac{2\Big(\sqrt{2t\ln n}\Big)^2}{t}\right) = n^{-4},$$

which concludes the proof. ■

Lemma 16 *For all integers $t \leqslant n$, for all suboptimal nodes (h,i) such that $\Delta_{h,i} > \nu_1\rho^h$, and for all integers $u \geqslant 1$ such that*

$$u \geqslant \frac{8\ln n}{(\Delta_{h,i}-\nu_1\rho^h)^2},$$

one has

$$\mathbb{P}\big\{U_{h,i}(t) > f^* \ \text{ and } \ T_{h,i}(t) > u\big\} \leqslant t\,n^{-4}.$$

Proof The u mentioned in the statement of the lemma are such that

$$\frac{\Delta_{h,i}-\nu_1\rho^h}{2} \geqslant \sqrt{\frac{2\ln n}{u}}, \qquad \text{thus} \qquad \sqrt{\frac{2\ln t}{u}}+\nu_1\rho^h \leqslant \frac{\Delta_{h,i}+\nu_1\rho^h}{2}.$$

Therefore,

$$\begin{aligned}
&\mathbb{P}\big\{U_{h,i}(t) > f^* \quad \text{and} \quad T_{h,i}(t) > u\big\} \\
&\quad = \mathbb{P}\left\{\widehat{\mu}_{h,i}(t) + \sqrt{\frac{2\ln t}{T_{h,i}(t)}} + \nu_1\rho^h > f^*_{h,i} + \Delta_{h,i} \quad \text{and} \quad T_{h,i}(t) > u\right\} \\
&\quad \leqslant \mathbb{P}\left\{\widehat{\mu}_{h,i}(t) > f^*_{h,i} + \frac{\Delta_{h,i} - \nu_1\rho^h}{2} \quad \text{and} \quad T_{h,i}(t) > u\right\} \\
&\quad \leqslant \mathbb{P}\left\{T_{h,i}(t)\,\big(\widehat{\mu}_{h,i}(t) - f^*_{h,i}\big) > \frac{\Delta_{h,i} - \nu_1\rho^h}{2}\, T_{h,i}(t) \quad \text{and} \quad T_{h,i}(t) > u\right\} \\
&\quad = \mathbb{P}\left\{\sum_{s=1}^{t} \big(Y_s - f^*_{h,i}\big)\mathbb{I}_{\{(H_s,I_s)\in C(h,i)\}} > \frac{\Delta_{h,i} - \nu_1\rho^h}{2}\, T_{h,i}(t) \quad \text{and} \quad T_{h,i}(t) > u\right\} \\
&\quad \leqslant \mathbb{P}\left\{\sum_{s=1}^{t} \big(Y_s - f(X_s)\big)\mathbb{I}_{\{(H_s,I_s)\in C(h,i)\}} > \frac{\Delta_{h,i} - \nu_1\rho^h}{2}\, T_{h,i}(t) \quad \text{and} \quad T_{h,i}(t) > u\right\}.
\end{aligned}$$

Now it follows from the same arguments as in the proof of Lemma 15 (optional skipping, the Hoeffding-Azuma inequality, and a union bound) that

$$\begin{aligned}
&\mathbb{P}\left\{\sum_{s=1}^{t} \big(Y_s - f(X_s)\big)\,\mathbb{I}_{\{(H_s,I_s)\in C(h,i)\}} > \frac{\Delta_{h,i} - \nu_1\rho^h}{2}\, T_{h,i}(t) \quad \text{and} \quad T_{h,i}(t) > u\right\} \\
&\quad \leqslant \sum_{s'=u+1}^{t} \exp\left(-\frac{2}{s'}\left(\frac{(\Delta_{h,i} - \nu_1\rho^h)}{2}\, s'\right)^2\right) \leqslant \sum_{s'=u+1}^{t} \exp\left(-\frac{1}{2}\, s'\,(\Delta_{h,i} - \nu_1\rho^h)^2\right) \\
&\quad \leqslant t\,\exp\left(-\frac{1}{2}\, u\,\big(\Delta_{h,i} - \nu_1\rho^h\big)^2\right) \leqslant t\,n^{-4},
\end{aligned}$$

where we used the stated bound on u to obtain the last inequality. ■

Combining the results of Lemmas 14, 15, and 16 leads to the following key result bounding the expected number of visits to descendants of a "poor" node.

Lemma 17 *Under Assumptions A1 and A2, for all suboptimal nodes (h,i) with $\Delta_{h,i} > \nu_1\rho^h$, we have, for all $n \geqslant 1$,*

$$\mathbb{E}[T_{h,i}(n)] \leqslant \frac{8\,\ln n}{(\Delta_{h,i} - \nu_1\rho^h)^2} + 4\,.$$

Proof We take u as the upper integer part of $(8\,\ln n)/(\Delta_{h,i} - \nu_1\rho^h)^2$ and use union bounds to get from Lemma 14 the bound

$$\begin{aligned}
\mathbb{E}\big[T_{h,i}(n)\big] &\leqslant \frac{8\,\ln n}{(\Delta_{h,i} - \nu_1\rho^h)^2} + 1 \\
&\qquad + \sum_{t=u+1}^{n}\left(\mathbb{P}\big\{T_{h,i}(t) > u \ \text{ and } \ U_{h,i}(t) > f^*\big\} + \sum_{s=1}^{t-1}\mathbb{P}\big\{U_{s,i_s^*}(t) \leqslant f^*\big\}\right).
\end{aligned}$$

Lemmas 15 and 16 further bound the quantity of interest as

$$\mathbb{E}\big[T_{h,i}(n)\big] \leqslant \frac{8\ln n}{(\Delta_{h,i}-\nu_1\rho^h)^2}+1+\sum_{t=u+1}^{n}\left(t\,n^{-4}+\sum_{s=1}^{t-1}t^{-3}\right)$$

and we now use the crude upper bounds

$$1+\sum_{t=u+1}^{n}\left(t\,n^{-4}+\sum_{s=1}^{t-1}t^{-3}\right)\leqslant 1+\sum_{t=1}^{n}\big(n^{-3}+t^{-2}\big)\leqslant 2+\pi^2/6\leqslant 4$$

to get the proposed statement. ■

Proof (of Theorem 6) First, let us fix $d' > d$. The statement will be proven in four steps.

First step. For all $h = 0,1,2,\ldots$, denote by I_h the set of those nodes at depth h that are $2\nu_1\rho^h$-optimal, that is, the nodes (h,i) such that $f^*_{h,i} \geqslant f^* - 2\nu_1\rho^h$. (Of course, $I_0 = \{(0,1)\}$.) Then, let I be the union of these sets when h varies. Further, let $\mathcal{J}$ be the set of nodes that are not in I but whose parent is in I. Finally, for $h = 1,2,\ldots$ we denote by $\mathcal{J}_h$ the nodes in $\mathcal{J}$ that are located at depth h in the tree (i.e., whose parent is in I_{h-1}).

Lemma 17 bounds in particular the expected number of times each node $(h,i) \in \mathcal{J}_h$ is visited. Since for these nodes $\Delta_{h,i} > 2\nu_1\rho^h$, we get

$$\mathbb{E}\big[T_{h,i}(n)\big] \leqslant \frac{8\ln n}{\nu_1^2\rho^{2h}}+4\,.$$

Second step. We bound the cardinality $|I_h|$ of I_h. We start with the case $h \geqslant 1$. By definition, when $(h,i) \in I_h$, one has $\Delta_{h,i} \leqslant 2\nu_1\rho^h$, so that by Lemma 3 the inclusion $\mathcal{P}_{h,i} \subset \mathcal{X}_{4\nu_1\rho^h}$ holds. Since by Assumption A1, the sets $\mathcal{P}_{h,i}$ contain disjoint balls of radius $\nu_2\rho^h$, we have that

$$|I_h| \leqslant \mathcal{N}\big(\cup_{(h,i)\in I_h}\mathcal{P}_{h,i},\,\ell,\,\nu_2\rho^h\big) \leqslant \mathcal{N}\big(\mathcal{X}_{4\nu_1\rho^h},\,\ell,\,\nu_2\rho^h\big) = \mathcal{N}\big(\mathcal{X}_{(4\nu_1/\nu_2)\,\nu_2\rho^h},\,\ell,\,\nu_2\rho^h\big)\,.$$

We prove below that there exists a constant C such that for all $\varepsilon \leqslant \nu_2$,

$$\mathcal{N}\big(\mathcal{X}_{(4\nu_1/\nu_2)\,\varepsilon},\,\ell,\,\varepsilon\big) \leqslant C\,\varepsilon^{-d'}\,. \tag{10}$$

Thus we obtain the bound $|I_h| \leqslant C\left(\nu_2\rho^h\right)^{-d'}$ for all $h \geqslant 1$. We note that the obtained bound $|I_h| \leqslant C\left(\nu_2\rho^h\right)^{-d'}$ is still valid for $h = 0$, since $|I_0| = 1$.

It only remains to prove (10). Since $d' > d$, where d is the near-optimality of f, we have, by definition, that

$$\limsup_{\varepsilon\to 0}\frac{\ln \mathcal{N}\big(\mathcal{X}_{(4\nu_1/\nu_2)\,\varepsilon},\,\ell,\,\varepsilon\big)}{\ln\big(\varepsilon^{-1}\big)} \leqslant d\,,$$

and thus, there exists $\varepsilon_{d'} > 0$ such that for all $\varepsilon \leqslant \varepsilon_{d'}$,

$$\frac{\ln \mathcal{N}\big(\mathcal{X}_{(4\nu_1/\nu_2)\,\varepsilon},\,\ell,\,\varepsilon\big)}{\ln\big(\varepsilon^{-1}\big)} \leqslant d'\,,$$

which in turn implies that for all $\varepsilon \leqslant \varepsilon_{d'}$,

$$\mathcal{N}\big(\mathcal{X}_{(4\nu_1/\nu_2)\,\varepsilon},\,\ell,\,\varepsilon\big) \leqslant \varepsilon^{-d'}\,.$$

The result is proved with $C = 1$ if $\varepsilon_{d'} \geqslant \nu_2$. Now, consider the case $\varepsilon_{d'} < \nu_2$. Given the definition of packing numbers, it is straightforward that for all $\varepsilon \in \big[\varepsilon_{d'}, \nu_2\big]$,

$$\mathcal{N}\big(\mathcal{X}_{(4\nu_1/\nu_2)\,\varepsilon}, \ell, \varepsilon\big) \leqslant u_{d'} \overset{\text{def}}{=} \mathcal{N}\big(\mathcal{X}, \ell, \varepsilon_{d'}\big) \ ;$$

therefore, for all $\varepsilon \in \big[\varepsilon_{d'}, \nu_2\big]$,

$$\mathcal{N}\big(\mathcal{X}_{(4\nu_1/\nu_2)\,\varepsilon}, \ell, \varepsilon\big) \leqslant u_{d'} \frac{\nu_2^{d'}}{\varepsilon^{d'}} = C\varepsilon^{-d'}$$

for the choice $C = \max\big\{1,\ u_{d'}\nu_2^{d'}\big\}$. Because we take the maximum with 1, the stated inequality also holds for $\varepsilon \leqslant \varepsilon^{-d'}$, which concludes the proof of (10).

Third step. Let $H \geqslant 1$ be an integer to be chosen later. We partition the nodes of the infinite tree $\mathcal{T}$ into three subsets, $\mathcal{T} = \mathcal{T}^1 \cup \mathcal{T}^2 \cup \mathcal{T}^3$, as follows. Let the set $\mathcal{T}^1$ contain the descendants of the nodes in I_H (by convention, a node is considered its own descendant, hence the nodes of I_H are included in $\mathcal{T}^1$); let $\mathcal{T}^2 = \cup_{0 \leqslant h < H} I_h$; and let $\mathcal{T}^3$ contain the descendants of the nodes in $\cup_{1 \leqslant h \leqslant H} \mathcal{J}_h$. Thus, $\mathcal{T}^1$ and $\mathcal{T}^3$ are potentially infinite, while $\mathcal{T}^2$ is finite.

We recall that we denote by (H_t, I_t) the node that was chosen by HOO in round t. From the definition of the algorithm, each node is played at most once, thus no two such random variables are equal when t varies. We decompose the regret according to which of the sets $\mathcal{T}^j$ the nodes (H_t, I_t) belong to:

$$\mathbb{E}\big[R_n\big] = \mathbb{E}\left[\sum_{t=1}^{n}(f^* - f(X_t))\right] = \mathbb{E}\big[R_{n,1}\big] + \mathbb{E}\big[R_{n,2}\big] + \mathbb{E}\big[R_{n,3}\big]\,,$$

$$\text{where} \qquad R_{n,i} = \sum_{t=1}^{n}\big(f^* - f(X_t)\big)\mathbb{I}_{\{(H_t,I_t)\in\mathcal{T}^i\}}\,, \qquad \text{for } i = 1,2,3.$$

The contribution from $\mathcal{T}^1$ is easy to bound. By definition any node in I_H is $2\nu_1\rho^H$-optimal. Hence, by Lemma 3, the corresponding domain is included in $\mathcal{X}_{4\nu_1\rho^H}$. By definition of a tree of coverings, the domains of the descendants of these nodes are still included in $\mathcal{X}_{4\nu_1\rho^H}$. Therefore,

$$\mathbb{E}\big[R_{n,1}\big] \leqslant 4\nu_1\rho^H n\,.$$

For $h \geqslant 0$, consider a node $(h,i) \in \mathcal{T}^2$. It belongs to I_h and is therefore $2\nu_1\rho^h$-optimal. By Lemma 3, the corresponding domain is included in $\mathcal{X}_{4\nu_1\rho^h}$. By the result of the second step of this proof and using that each node is played at most once, one gets

$$\mathbb{E}\big[R_{n,2}\big] \leqslant \sum_{h=0}^{H-1} 4\nu_1\rho^h\,|I_h| \leqslant 4C\nu_1\nu_2^{-d'}\sum_{h=0}^{H-1}\rho^{h(1-d')}\,.$$

We finish by bounding the contribution from $\mathcal{T}^3$. We first remark that since the parent of any element $(h,i) \in \mathcal{J}_h$ is in I_{h-1}, by Lemma 3 again, we have that $\mathcal{P}_{h,i} \subset \mathcal{X}_{4\nu_1\rho^{h-1}}$. We now use the first step of this proof to get

$$\mathbb{E}\big[R_{n,3}\big] \leqslant \sum_{h=1}^{H} 4\nu_1\rho^{h-1} \sum_{i\,:\,(h,i)\in\mathcal{J}_h} \mathbb{E}\big[T_{h,i}(n)\big] \leqslant \sum_{h=1}^{H} 4\nu_1\rho^{h-1}\,|\mathcal{J}_h|\left(\frac{8\ln n}{\nu_1^2\rho^{2h}} + 4\right).$$

Now, it follows from the fact that the parent of $\mathcal{J}_h$ is in I_{h-1} that $|\mathcal{J}_h| \leqslant 2|I_{h-1}|$ when $h \geqslant 1$. Substituting this and the bound on $|I_{h-1}|$ obtained in the second step of this proof, we get

$$\begin{aligned}\mathbb{E}\big[R_{n,3}\big] &\leqslant \sum_{h=1}^{H} 4\nu_1\rho^{h-1}\left(2C\left(\nu_2\rho^{h-1}\right)^{-d'}\right)\left(\frac{8\ln n}{\nu_1^2\rho^{2h}}+4\right)\\ &\leqslant 8C\nu_1\nu_2^{-d'}\sum_{h=1}^{H}\rho^{h(1-d')+d'-1}\left(\frac{8\ln n}{\nu_1^2\rho^{2h}}+4\right).\end{aligned}$$

Fourth step. Putting the obtained bounds together, we get

$$\begin{aligned}\mathbb{E}\big[R_n\big] &\leqslant 4\nu_1\rho^H n + 4C\nu_1\nu_2^{-d'}\sum_{h=0}^{H-1}\rho^{h(1-d')} + 8C\nu_1\nu_2^{-d'}\sum_{h=1}^{H}\rho^{h(1-d')+d'-1}\left(\frac{8\ln n}{\nu_1^2\rho^{2h}}+4\right)\\ &= O\left(n\rho^H + (\ln n)\sum_{h=1}^{H}\rho^{-h(1+d')}\right) = O\Big(n\rho^H + \rho^{-H(1+d')}\ln n\Big)\end{aligned}$$

(recall that $\rho < 1$). Note that all constants hidden in the O symbol only depend on ν_1, ν_2, ρ and d'.

Now, by choosing H such that $\rho^{-H(d'+2)}$ is of the order of $n/\ln n$, that is, ρ^H is of the order of $(n/\ln n)^{-1/(d'+2)}$, we get the desired result, namely,

$$\mathbb{E}\big[R_n\big] = O\Big(n^{(d'+1)/(d'+2)}\,(\ln n)^{1/(d'+2)}\Big).$$

■

A.2 Proof of Theorem 8 (Regret Bound for Truncated HOO)

The proof follows from an adaptation of the proof of Theorem 6 and of its associated lemmas; for the sake of clarity and precision, we explicitly state the adaptations of the latter.

Adaptations of the lemmas. Remember that D_{n_0} denotes the maximum depth of the tree, given horizon n_0. The adaptation of Lemma 14 is done as follows. Let (h,i) be a suboptimal node with $h \leqslant D_{n_0}$ and let $0 \leqslant k \leqslant h-1$ be the largest depth such that (k, i_k^*) is on the path from the root $(0,1)$ to (h,i). Then, for all integers $u \geqslant 0$, one has

$$\begin{aligned}\mathbb{E}\big[T_{h,i}(n_0)\big] \leqslant u + \sum_{t=u+1}^{n_0}\mathbb{P}\Big\{&\big[U_{s,i_s^*}(t) \leqslant f^* \text{ for some } s \text{ with } k+1 \leqslant s \leqslant \min\{D_{n_0}, n_0\}\big]\\ &\text{or}\ \ \big[T_{h,i}(t) > u \ \text{ and } \ U_{h,i}(t) > f^*\big]\Big\}.\end{aligned}$$

As for Lemma 15, its straightforward adaptation states that under Assumptions A1 and A2, for all optimal nodes (h,i) with $h \leqslant D_{n_0}$ and for all integers $1 \leqslant t \leqslant n_0$,

$$\mathbb{P}\big\{U_{h,i}(t) \leqslant f^*\big\} \leqslant t\,(n_0)^{-4} \leqslant (n_0)^3.$$

Similarly, the same changes yield from Lemma 16 the following result for truncated HOO. For all integers $t \leqslant n_0$, for all suboptimal nodes (h,i) such that $h \leqslant D_{n_0}$ and $\Delta_{h,i} > \nu_1\rho^h$, and for all integers $u \geqslant 1$ such that

$$u \geqslant \frac{8\ln n_0}{(\Delta_{h,i} - \nu_1\rho^h)^2},$$

one has

$$\mathbb{P}\big\{U_{h,i}(t) > f^* \text{ and } T_{h,i}(t) > u\big\} \leqslant t\,(n_0)^{-4}\,.$$

Combining these three results (using the same methodology as in the proof of Lemma 17) shows that under Assumptions A1 and A2, for all suboptimal nodes (h,i) such that $h \leqslant D_{n_0}$ and $\Delta_{h,i} > \nu_1\rho^h$, one has

$$\begin{aligned}\mathbb{E}[T_{h,i}(n_0)] &\leqslant \frac{8\,\ln n_0}{(\Delta_{h,i}-\nu_1\rho^h)^2}+1+\sum_{t=u+1}^{n_0}\left(t\,(n_0)^4+\sum_{s=1}^{\min\{D_{n_0},n_0\}}(n_0)^{-3}\right)\\ &\leqslant \frac{8\,\ln n_0}{(\Delta_{h,i}-\nu_1\rho^h)^2}+3\,.\end{aligned}$$

(We thus even improve slightly the bound of Lemma 17.)

Adaptation of the proof of Theorem 6. The main change here comes from the fact that trees are cut at the depth D_{n_0}. As a consequence, the sets I_h, I, $\mathcal{J}$, and $\mathcal{J}_h$ are defined only by referring to nodes of depth smaller than D_{n_0}. All steps of the proof can then be repeated, except the third step; there, while the bounds on the regret resulting from nodes of $\mathcal{T}^1$ and $\mathcal{T}^3$ go through without any changes (as these sets were constructed by considering all descendants of some base nodes), the bound on the regret $R_{n,2}$ associated with the nodes $\mathcal{T}^2$ calls for a modified proof since at this stage we used the property that each node is played at most once. But this is not true anymore for nodes (h,i) located at depth D_{n_0}, which can be played several times. Therefore the proof is modified as follows.

Consider a node at depth $h = D_{n_0}$. Then, by definition of D_{n_0},

$$h \geqslant D_{n_0} = \frac{(\ln n_0)/2 - \ln(1/\nu_1)}{\ln(1/\rho)}\,, \qquad \text{that is,} \qquad \nu_1\,\rho^h \leqslant \frac{1}{\sqrt{n_0}}\,.$$

Since the considered nodes are $2\nu_1\rho^{D_{n_0}}$-optimal, the corresponding domains are $4\nu_1\rho^{D_{n_0}}$-optimal by Lemma 3, thus also $4/\sqrt{n_0}$-optimal. The instantaneous regret incurred when playing any of these nodes is therefore bounded by $4/\sqrt{n_0}$; and the associated cumulative regret (over n_0 rounds) can be bounded by $4\sqrt{n_0}$. In conclusion, with the notations of Theorem 6, we get the new bound

$$\mathbb{E}\big[R_{n,2}\big] \leqslant \sum_{h=0}^{H-1} 4\nu_1\rho^h\,|I_h| + 4\sqrt{n_0} \leqslant 4\sqrt{n_0} + 4C\nu_1\nu_2^{-d'}\sum_{h=0}^{H-1}\rho^{h(1-d')}\,.$$

The rest of the proof goes through and only this additional additive factor of $4\sqrt{n_0}$ is suffered in the final regret bound. (The additional factor can be included in the O notation.)

A.3 Proof of Theorem 9 (Regret Bound for z-HOO)

We start with the following equivalent of Lemma 3 in this new local context. Remember that h_0 is the smallest integer such that

$$2\nu_1\rho^{h_0} < \varepsilon_0\,.$$

Lemma 18 *Under Assumptions A1 and A2', for all $h \geqslant h_0$, if the suboptimality factor $\Delta_{h,i}$ of a region $\mathcal{P}_{h,i}$ is bounded by $c\nu_1\rho^h$ for some $c \in [0,2]$, then all arms in $\mathcal{P}_{h,i}$ are $L\max\{2c,\, c+1\}\nu_1\rho^h$-optimal, that is,*

$$\mathcal{P}_{h,i} \subset \mathcal{X}_{L\max\{2c,c+1\}\nu_1\rho^h}\,.$$

When $c = 0$, that is, the node (h,i) is optimal, the bound improves to

$$\mathcal{P}_{h,i} \subset \mathcal{X}_{\nu_1\rho^h}\,.$$

Proof We first deal with the general case of $c \in [0,2]$. By the hypothesis on the suboptimality of $\mathcal{P}_{h,i}$, for all $\delta > 0$, there exists an element $x \in \mathcal{X}_{c\nu_1\rho^h+\delta} \cap \mathcal{P}_{h,i}$. If δ is small enough, for example, $\delta \in \big(0,\, \varepsilon_0 - 2\nu_1\rho^{h_0}\big]$, then this element satisfies $x \in \mathcal{X}_{\varepsilon_0}$. Let $y \in \mathcal{P}_{h,i}$. By Assumption A1, $\ell(x,y) \leqslant \operatorname{diam}(\mathcal{P}_{h,i}) \leqslant \nu_1\rho^h$, which entails, by denoting $\varepsilon = \max\big\{0, \nu_1\rho^h - (f^* - f(x))\big\}$,

$$\ell(x,y) \leqslant \nu_1\rho^h \leqslant f^* - f(x) + \varepsilon\,, \qquad \text{that is,} \qquad y \in \mathcal{B}\big(x,\, f^* - f(x) + \varepsilon\big)\,.$$

Since $x \in \mathcal{X}_{\varepsilon_0}$ and $\varepsilon \leqslant \nu_1\rho^h \leqslant \nu_1\rho^{h_0} < \varepsilon_0$, the second part of Assumption A2' then yields

$$y \in \mathcal{B}\big(x,\, f^* - f(x) + \varepsilon\big) \;\subset \mathcal{X}_{L\left(2(f^*-f(x))+\varepsilon\right)}\,.$$

It follows from the definition of ε that $f^* - f(x) + \varepsilon = \max\big\{f^* - f(x),\, \nu_1\rho^h\big\}$, and this implies

$$y \in \mathcal{B}\big(x,\, f^* - f(x) + \varepsilon\big) \;\subset \mathcal{X}_{L\left(f^*-f(x)+\max\{f^*-f(x),\nu_1\rho^h\}\right)}\,.$$

But $x \in \mathcal{X}_{c\nu_1\rho^h+\delta}$, that is, $f^* - f(x) \leqslant c\nu_1\rho^h + \delta$, we thus have proved

$$y \in \mathcal{X}_{L\left(\max\{2c,c+1\}\nu_1\rho^h+2\delta\right)}\,.$$

In conclusion, $\mathcal{P}_{h,i} \subset \mathcal{X}_{L\max\{2c,c+1\}\nu_1\rho^h+2L\delta}$ for all sufficiently small $\delta > 0$. Letting $\delta \to 0$ concludes the proof.

In the case of $c = 0$, we resort to the first part of Assumption A2', which can be applied since $\operatorname{diam}(\mathcal{P}_{h,i}) \leqslant \nu_1\rho^h \leqslant \varepsilon_0$ as already noted above, and can exactly be restated as indicating that for all $y \in \mathcal{P}_{h,i}$,

$$f^* - f(y) \leqslant \operatorname{diam}(\mathcal{P}_{h,i}) \leqslant \nu_1\rho^h\,;$$

that is, $\mathcal{P}_{h,i} \subset \mathcal{X}_{\nu_1\rho^h}$. ■

We now provide an adaptation of Lemma 17 (actually based on adaptations of Lemmas 14 and 15), providing the same bound under local conditions that relax the assumptions of Lemma 17 to some extent.

Lemma 19 *Consider a depth $z \geqslant h_0$. Under Assumptions A1 and A2', the algorithm z-HOO satisfies that for all $n \geqslant 1$ and all suboptimal nodes (h,i) with $\Delta_{h,i} > \nu_1\rho^h$ and $h \geqslant z$,*

$$\mathbb{E}\big[T_{h,i}(n)\big] \leqslant \frac{8\,\ln n}{(\Delta_{h,i} - \nu_1\rho^h)^2} + 4\,.$$

Proof We consider some path (z,i^*_z), $(z+1,i^*_{z+1}), \ldots$ of optimal nodes, starting at depth z. We distinguish two cases, depending on whether there exists $z \leqslant k' \leqslant h-1$ such that $(h,i) \in \mathcal{C}(k',i^*_{k'})$ or not.

In the first case, we denote k' the largest such k. The argument of Lemma 14 can be used without any change and shows that for all integers $u \geqslant 0$,

$$\mathbb{E}\big[T_{h,i}(n)\big] \leqslant u + \sum_{t=u+1}^{n} \mathbb{P}\Big\{ \big[U_{s,i^*_s}(t) \leqslant f^* \text{ for some } s \in \{k+1,\ldots,t-1\}\big] \\ \text{or } \big[T_{h,i}(t) > u \text{ and } U_{h,i}(t) > f^*\big] \Big\}.$$

In the second case, we denote by (z,i_h) the ancestor of (h,i) located at depth z. By definition of z-HOO, $(H_t,I_t) \in \mathcal{C}(h,i)$ at some round $t \geqslant 1$ only if $B_{z,i^*_z}(t) \leqslant B_{z,i_h}(t)$ and since B-values can only increase on a chosen path, $(H_t,I_t) \in \mathcal{C}(h,i)$ can only happen if $B_{z,i^*_z}(t) \leqslant B_{h,i}(t)$. Repeating again the argument of Lemma 14, we get that for all integers $u \geqslant 0$,

$$\mathbb{E}\big[T_{h,i}(n)\big] \leqslant u + \sum_{t=u+1}^{n} \mathbb{P}\Big\{ \big[U_{s,i^*_s}(t) \leqslant f^* \text{ for some } s \in \{z,\ldots,t-1\}\big] \\ \text{or } \big[T_{h,i}(t) > u \text{ and } U_{h,i}(t) > f^*\big] \Big\}.$$

Now, notice that Lemma 16 is valid without any assumption. On the other hand, with the modified assumptions, Lemma 15 is still true but only for optimal nodes (h,i) with $h \geqslant h_0$. Indeed, the only point in its proof where the assumptions were used was in the fourth line, when applying Lemma 3; here, Lemma 18 with $c = 0$ provides the needed guarantee.

The proof is concluded with the same computations as in the proof of Lemma 17. ■

Proof (of Theorem 9) We follow the four steps in the proof of Theorem 6 with some slight adjustments. In particular, for $h \geqslant z$, we use the sets of nodes I_h and $\mathcal{J}_h$ defined therein.

First step. Lemma 19 bounds the expected number of times each node $(h,i) \in \mathcal{J}_h$ is visited. Since for these nodes $\Delta_{h,i} > 2\nu_1\rho^h$, we get

$$\mathbb{E}\big[T_{h,i}(n)\big] \leqslant \frac{8 \ln n}{\nu_1^2 \rho^{2h}} + 4 .$$

Second step. We bound here the cardinality $|I_h|$. By Lemma 18 with $c = 2$, when $(h,i) \in I_h$ and $h \geqslant z$, one has $\mathcal{P}_{h,i} \subset \mathcal{X}_{4L\nu_1\rho^h}$.

Now, by Assumption A1 and by using the same argument as in the second step of the proof of Theorem 6,

$$|I_h| \leqslant \mathcal{N}\big(\mathcal{X}_{(4L\nu_1/\nu_2)\nu_2\rho^h}, \ell, \nu_2\rho^h\big) .$$

Assumption A3 can be applied since $\nu_2\rho^h \leqslant 2\nu_1\rho^h \leqslant 2\nu_1\rho^{h_0} \leqslant \varepsilon_0$ and yields the inequality $|I_h| \leqslant C\big(\nu_2\rho^h\big)^{-d}$.

Third step. We consider some integer $H \geqslant z$ to be defined by the analysis in the fourth step. We define a partition of the nodes located at a depth equal to or larger than z; more precisely,

- $\mathcal{T}^1$ contains the nodes of I_H and their descendants,

- $\mathcal{T}^2 = \bigcup_{z \leqslant h \leqslant H-1} I_h$,
- $\mathcal{T}^3$ contains the nodes $\bigcup_{z+1 \leqslant h \leqslant H} \mathcal{J}_h$ and their descendants,
- $\mathcal{T}^4$ is formed by the nodes (z,i) located at depth z not belonging to I_z, that is, such that $\Delta_{z,i} > 2\nu_1\rho^z$, and their descendants.

As in the proof of Theorem 6 we denote by $R_{n,i}$ the regret resulting from the selection of nodes in $\mathcal{T}^i$, for $i \in \{1,2,3,4\}$.

Lemma 18 with $c = 2$ yields the bound $\mathbb{E}\big[R_{n,1}\big] \leqslant 4L\nu_1\rho^H n$, where we crudely bounded by n the number of times that nodes in $\mathcal{T}^1$ were played. Using that by definition each node of $\mathcal{T}^2$ can be played only once, we get

$$\mathbb{E}\big[R_{n,2}\big] \leqslant \sum_{h=z}^{H-1} \big(4L\nu_1\rho^h\big)\,|I_h| \leqslant 4CL\nu_1\nu_2^{-d} \sum_{h=z}^{H-1} \rho^{h(1-d)}\,.$$

As for $R_{n,3}$, we also use here that nodes in $\mathcal{T}^3$ belong to some $\mathcal{J}_h$, with $z+1 \leqslant h \leqslant H$; in particular, they are the child of some element of I_{h-1} and as such, firstly, they are $4L\nu_1\rho^{h-1}$-optimal (by Lemma 18) and secondly, their number is bounded by $|\mathcal{J}_h| \leqslant 2|I_{h-1}| \leqslant 2C\big(\nu_2\rho^{h-1}\big)^{-d}$. Thus,

$$\mathbb{E}\big[R_{n,3}\big] \leqslant \sum_{h=z+1}^{H} \big(4L\nu_1\rho^{h-1}\big) \sum_{i:(h,i)\in\mathcal{J}_h} \mathbb{E}\big[T_{h,i}(n)\big] \leqslant 8CL\nu_1\nu_2^{-d} \sum_{h=z+1}^{H} \rho^{(h-1)(1-d)} \left(\frac{8\,\ln n}{\nu_1^2\rho^{2h}} + 4\right),$$

where we used the bound of Lemma 19. Finally, for $\mathcal{T}^4$, we use that it contains at most $2^z - 1$ nodes, each of them being associated with a regret controlled by Lemma 19; therefore,

$$\mathbb{E}\big[R_{n,4}\big] \leqslant \big(2^z - 1\big) \left(\frac{8\,\ln n}{\nu_1^2\rho^{2z}} + 4\right).$$

Fourth step. Putting things together, we have proved that

$$\mathbb{E}\big[R_n\big] \leqslant 4L\nu_1\rho^H n + \mathbb{E}\big[R_{n,2}\big] + \mathbb{E}\big[R_{n,3}\big] + \big(2^z - 1\big) \left(\frac{8\,\ln n}{\nu_1^2\rho^{2z}} + 4\right),$$

where (using that $\rho < 1$ in the second inequality)

$$\begin{aligned}
&\mathbb{E}\big[R_{n,2}\big] + \mathbb{E}\big[R_{n,3}\big] \\
&\leqslant\; 4CL\nu_1\nu_2^{-d} \sum_{h=z}^{H-1} \rho^{h(1-d)} + 8CL\nu_1\nu_2^{-d} \sum_{h=z+1}^{H} \rho^{(h-1)(1-d)} \left(\frac{8\,\ln n}{\nu_1^2\rho^{2h}} + 4\right) \\
&=\; 4CL\nu_1\nu_2^{-d} \sum_{h=z}^{H-1} \rho^{h(1-d)} + 8CL\nu_1\nu_2^{-d} \sum_{h=z}^{H-1} \rho^{h(1-d)} \left(\frac{8\,\ln n}{\nu_1^2\rho^2\rho^{2h}} + 4\right) \\
&\leqslant\; 4CL\nu_1\nu_2^{-d} \sum_{h=z}^{H-1} \rho^{h(1-d)} \frac{1}{\rho^{2h}} + 8CL\nu_1\nu_2^{-d} \sum_{h=z}^{H-1} \rho^{h(1-d)} \left(\frac{8\,\ln n}{\nu_1^2\rho^2\rho^{2h}} + \frac{4}{\rho^{2h}}\right) \\
&=\; CL\nu_1\nu_2^{-d} \left(\sum_{h=z}^{H-1} \rho^{-h(1+d)}\right) \left(36 + \frac{64}{\nu_1^2\rho^2}\,\ln n\right).
\end{aligned}$$

Denoting

$$\gamma = \frac{4CL\nu_1\nu_2^{-d}}{(1/\rho)^{d+1}-1}\left(\frac{16}{\nu_1^2\rho^2}+9\right),$$

it follows that for $n \geqslant 2$

$$\mathbb{E}\big[R_{n,2}\big]+\mathbb{E}\big[R_{n,3}\big] \leqslant \gamma\,\rho^{-H(d+1)}\ln n\,.$$

It remains to define the parameter $H \geqslant z$. In particular, we propose to choose it such that the terms

$$4L\nu_1\rho^H n \qquad \text{and} \qquad \rho^{-H(d+1)}\ln n$$

are balanced. To this end, let H be the smallest integer k such that $4L\nu_1\rho^k n \leqslant \gamma\rho^{-k(d+1)}\ln n$; in particular,

$$\rho^H \leqslant \left(\frac{\gamma\ln n}{4L\nu_1 n}\right)^{1/(d+2)}$$

and

$$4L\nu_1\rho^{H-1}n > \gamma\rho^{-(H-1)(d+1)}\ln n, \qquad \text{implying} \qquad \gamma\,\rho^{-H(d+1)}\ln n \leqslant 4L\nu_1\rho^H n\,\rho^{-(d+2)}\,.$$

Note from the inequality that this H is such that

$$H \geqslant \frac{1}{d+2}\,\frac{\ln(4L\nu_1 n)-\ln(\gamma\ln n)}{\ln(1/\rho)}$$

and thus this H satisfies $H \geqslant z$ in view of the assumption of the theorem indicating that n is large enough. The final bound on the regret is then

$$\begin{aligned}
\mathbb{E}\big[R_n\big] &\leqslant 4L\nu_1\rho^H n+\gamma\,\rho^{-H(d+1)}\ln n+\big(2^z-1\big)\left(\frac{8\,\ln n}{\nu_1^2\rho^{2z}}+4\right)\\
&\leqslant \left(1+\frac{1}{\rho^{d+2}}\right)4L\nu_1\rho^H n+\big(2^z-1\big)\left(\frac{8\,\ln n}{\nu_1^2\rho^{2z}}+4\right)\\
&\leqslant \left(1+\frac{1}{\rho^{d+2}}\right)4L\nu_1 n\left(\frac{\gamma\ln n}{4L\nu_1 n}\right)^{1/(d+2)}+\big(2^z-1\big)\left(\frac{8\,\ln n}{\nu_1^2\rho^{2z}}+4\right)\\
&= \left(1+\frac{1}{\rho^{d+2}}\right)\big(4L\nu_1 n\big)^{(d+1)/(d+2)}(\gamma\ln n)^{1/(d+2)}+\big(2^z-1\big)\left(\frac{8\,\ln n}{\nu_1^2\rho^{2z}}+4\right).
\end{aligned}$$

This concludes the proof. ■

A.4 Proof of Theorem 10 (Regret Bound for Local-HOO)

Proof We use the notation of the proof of Theorem 9. Let r_0 be a positive integer such that for $r \geqslant r_0$, one has

$$z_r \overset{\text{def}}{=} \lceil\log_2 r\rceil \geqslant h_0 \qquad \text{and} \qquad z_r \leqslant \frac{1}{d+2}\,\frac{\ln(4L\nu_1 2^r)-\ln(\gamma\ln 2^r)}{\ln(1/\rho)}\,;$$

we can therefore apply the result of Theorem 9 in regimes indexed by $r \geqslant r_0$. For previous regimes, we simply upper bound the regret by the number of rounds, that is, $2^{r_0}-2 \leqslant 2^{r_0}$. For round n, we denote by r_n the index of the regime where n lies in (regime $r_n = \lfloor \log_2(n+1) \rfloor$). Since regime r_n terminates at round $2^{r_n+1} - 2$, we have

$$\begin{aligned}
\mathbb{E}\big[R_n\big] &\leqslant \mathbb{E}\big[R_{2^{r_n+1}-2}\big] \\
&\leqslant 2^{r_0} + \sum_{r=r_0}^{r_n} \left(\left(1 + \frac{1}{\rho^{d+2}}\right) \left(4L\nu_1 2^r\right)^{(d+1)/(d+2)} (\gamma \ln 2^r)^{1/(d+2)} + \left(2^{z_r} - 1\right) \left(\frac{8 \ln 2^r}{\nu_1^2 \rho^{2z_r}} + 4 \right) \right) \\
&\leqslant 2^{r_0} + C_1 (\ln n) \sum_{r=r_0}^{r_n} \left(\left(2^{(d+1)/(d+2)}\right)^r + \left(2/\rho^2\right)^{z_r} \right) \\
&\leqslant 2^{r_0} + C_2 (\ln n) \left(\left(2^{(d+1)/(d+2)}\right)^{r_n} + r_n \left(2/\rho^2\right)^{z_{r_n}} \right) \; = (\ln n)\, O\big(n^{(d+1)/(d+2)}\big),
\end{aligned}$$

where $C_1, C_2 > 0$ denote some constants depending only on the parameters but not on n. Note that for the last equality we used that the first term in the sum of the two terms that depend on n dominates the second term. ■

A.5 Proof of Theorem 12 (Uniform Upper Bound on the Regret of HOO against the Class of all Weak Lipschitz Environments)

Equations (6) and (7), which follow from Assumption A2, show that Assumption A2' is satisfied for $L = 2$ and all $\varepsilon_0 > 0$. We take, for instance, $\varepsilon_0 = 3\nu_1$. Moreover, since $\mathcal{X}$ has a packing dimension of D, all environments have a near-optimality dimension less than D. In particular, for all $D' > D$ (as shown in the second step of the proof of Theorem 6 in Section A.1), there exists a constant C (depending only on ℓ, $\mathcal{X}$, $\varepsilon_0 = 3\nu_1$, ν_2, and D') such that Assumption A3 is satisfied. We can therefore take $h_0 = 0$ and apply Theorem 9 with $z = 0$ and $M \in \mathcal{F}_{\mathcal{X},\ell}$; the fact that all the quantities involved in the bound depend only on $\mathcal{X}$, ℓ, ν_2, D', and the parameters of HOO, but not on a particular environment in $\mathcal{F}$, concludes the proof.

A.6 Proof of Theorem 13 (Minimax Lower Bound in Metric Spaces)

Let $K \geqslant 2$ an integer to be defined later. We provide first an overview of the proof. Here, we exhibit a set $\mathcal{A}$ of environments for the $\{1, \dots, K+1\}$-armed bandit problem and a subset $\mathcal{F}' \subset \mathcal{F}_{\mathcal{X},\ell}$ which satisfy the following properties.

(i) The set $\mathcal{A}$ contains "difficult" environments for the $\{1, \dots, K+1\}$-armed bandit problem.

(ii) For any strategy $\varphi^{(\mathcal{X})}$ suited to the $\mathcal{X}$-armed bandit problem, one can construct a strategy $\psi^{(K+1)}$ for the $\{1, \dots, K+1\}$-armed bandit problem such that

$$\forall M \in \mathcal{F}', \; \exists \nu \in \mathcal{A}, \qquad \mathbb{E}_M\big[R_n(\varphi^{(\mathcal{X})})\big] = \mathbb{E}_\nu\big[R_n(\psi^{(K+1)})\big] .$$

We now provide the details.

Proof We only deal with the case of deterministic strategies. The extension to randomized strategies can be done using Fubini's theorem (by integrating also w.r.t. the auxiliary randomizations used).

First step. Let $\eta \in (0,1/2)$ be a real number and $K \geqslant 2$ be an integer, both to be defined during the course of the analysis. The set $\mathcal{A}$ only contains K elements, denoted by $\nu^1, \ldots, \nu^K$ and given by product distributions. For $1 \leqslant j \leqslant K$, the distribution ν^j is obtained as the product of the ν_i^j when $i \in \{1, \ldots, K+1\}$ and where

$$\nu_i^j = \begin{cases} \text{Ber}(1/2), & \text{if } i \neq j; \\ \text{Ber}(1/2 + \eta), & \text{if } i = j. \end{cases}$$

One can extract the following result from the proof of the lower bound of Cesa-Bianchi and Lugosi (2006, Section 6.9).

Lemma 20 *For all strategies $\psi^{(K+1)}$ for the $\{1, \ldots, K+1\}$-armed bandit (where $K \geqslant 2$), one has*

$$\max_{j=1,\ldots,K} \mathbb{E}_{\nu^j}\left[R_n(\psi^{(K+1)})\right] \geqslant n\eta \left(1 - \frac{1}{K} - \eta\sqrt{4\ln(4/3)}\sqrt{\frac{n}{K}}\right).$$

Second step. We now need to construct $\mathcal{F}'$ such that item (ii) is satisfied. We assume that K is such that $\mathcal{X}$ contains K disjoint balls with radius η. (We shall quantify later in this proof a suitable value of K.) Denoting by $x_1, \ldots, x_K$ the corresponding centers, these disjoint balls are then $\mathcal{B}(x_1, \eta), \ldots, \mathcal{B}(x_K, \eta)$.

With each of these balls we now associate a bandit environment over $\mathcal{X}$, in the following way. For all $x^* \in \mathcal{X}$, we introduce a mapping $g_{x^*,\eta}$ on $\mathcal{X}$ defined by

$$g_{x^*,\eta}(x) = \max\{0, \ \eta - \ell(x, x^*)\}$$

for all $x \in \mathcal{X}$. This mapping is used to define an environment $M_{x^*,\eta}$ over $\mathcal{X}$, as follows. For all $x \in \mathcal{X}$,

$$M_{x^*,\eta}(x) = \text{Ber}\left(\frac{1}{2} + g_{x^*,\eta}(x)\right).$$

Let $f_{x^*,\eta}$ be the corresponding mean-payoff function; its values equal

$$f_{x^*,\eta}(x) = \frac{1}{2} + \max\{0, \ \eta - \ell(x, x^*)\}$$

for all $x \in \mathcal{X}$. Note that the mean payoff is maximized at $x = x^*$ (with value $1/2 + \eta$) and is minimal for all points lying outside $\mathcal{B}(x^*, \eta)$, with value $1/2$. In addition, that ℓ is a metric entails that these mean-payoff functions are 1-Lipschitz and thus are also weakly Lipschitz. (This is the only point in the proof where we use that ℓ is a metric.) In conclusion, we consider

$$\mathcal{F}' = \{M_{x_1,\eta}, \ldots, M_{x_K,\eta}\} \subset \mathcal{F}_{\mathcal{X},\ell}.$$

Third step. We describe how to associate with each (deterministic) strategy $\varphi^{(\mathcal{X})}$ on $\mathcal{X}$ a (random) strategy $\psi^{(K+1)}$ on the finite set of arms $\{1, \ldots, K+1\}$. Each of these strategies is indeed given by a sequence of mappings,

$$\varphi_1^{(\mathcal{X})}, \varphi_2^{(\mathcal{X})}, \ldots \qquad \text{and} \qquad \psi_1^{(K+1)}, \psi_2^{(K+1)}, \ldots$$

where for $t \geqslant 1$, the mappings $\varphi_t^{(\mathcal{X})}$ and $\psi_t^{(K+1)}$ should only depend on the past up to the beginning of round t. Since the strategy $\varphi^{(\mathcal{X})}$ is deterministic, the mapping $\varphi_t^{(\mathcal{X})}$ takes only into account the past rewards $Y_1,\dots,Y_{t-1}$ and is therefore a mapping $[0,1]^{t-1} \to \mathcal{X}$. (In particular, $\varphi_1^{(\mathcal{X})}$ equals a constant.)

We use the notations I'_t and Y'_t for, respectively, the arms pulled and the rewards obtained by the strategy $\psi^{(K+1)}$ at each round t. The arms I'_t are drawn at random according to the distributions

$$\psi_t^{(K+1)}\big(I'_1,\dots,I'_{t-1},Y'_1,\dots,Y'_{t-1}\big)\,,$$

which we now define. (Actually, they will depend on the obtained payoffs $Y'_1,\dots,Y'_{t-1}$ only.) To do that, we need yet another mapping T that links elements in $\mathcal{X}$ to probability distributions over $\{1,\dots,K+1\}$. Denoting by δ_k the Dirac probability on $k \in \{1,\dots,K+1\}$, the mapping T is defined as

$$T(x) = \begin{cases} \delta_{K+1}\,, & \text{if } x \notin \displaystyle\bigcup_{j=1,\dots,K} \mathcal{B}(x_j,\eta); \\ \left(1-\dfrac{\ell(x,x_j)}{\eta}\right)\delta_j + \dfrac{\ell(x,x_j)}{\eta}\,\delta_{K+1}\,, & \text{if } x \in \mathcal{B}(x_j,\eta) \text{ for some } j \in \{1,\dots,K\}, \end{cases}$$

for all $x \in \mathcal{X}$. Note that this definition is legitimate because the balls $\mathcal{B}(x_j,\eta)$ are disjoint when j varies between 1 and K.

Finally, $\psi^{(K+1)}$ is defined as follows. For all $t \geqslant 1$,

$$\psi_t^{(K+1)}\big(I'_1,\dots,I'_{t-1},Y'_1,\dots,Y'_{t-1}\big) = \psi_t^{(K+1)}\big(Y'_1,\dots,Y'_{t-1}\big) = T\Big(\varphi_t^{(\mathcal{X})}\big(Y'_1,\dots,Y'_{t-1}\big)\Big)\,.$$

Before we proceed, we study the distribution of the reward Y' obtained under ν^i (for $i \in \{1,\dots,K\}$) by the choice of a random arm I' drawn according to $T(x)$, for some $x \in \mathcal{X}$. Since Y' can only take the values 0 or 1, its distribution is a Bernoulli distribution whose parameter $\mu_i(x)$ we compute now. The computation is based on the fact that under ν^i, the Bernoulli distribution corresponding to arm j has $1/2$ as an expectation, except if $j = i$, in which case it is $1/2+\eta$. Thus, for all $x \in \mathcal{X}$,

$$\mu_i(x) = \begin{cases} 1/2\,, & \text{if } x \notin \mathcal{B}(x_i,\eta); \\ \left(1-\dfrac{\ell(x,x_i)}{\eta}\right)\left(\dfrac{1}{2}+\eta\right) + \dfrac{\ell(x,x_i)}{\eta}\,\dfrac{1}{2} = \dfrac{1}{2}+\eta-\ell(x,x_i)\,, & \text{if } x \in \mathcal{B}(x_i,\eta). \end{cases}$$

That is, $\mu_i = f_{x_i,\eta}$ on $\mathcal{X}$.

Fourth step. We now prove that the distributions of the regrets of $\varphi^{(\mathcal{X})}$ under $M_{x_j,\eta}$ and of $\psi^{(K+1)}$ under ν^j are equal for all $j = 1,\dots,K$. On the one hand, the expectations of rewards associated with the best arms equal $1/2+\eta$ under the two environments. On the other hand, one can prove by induction that the sequences $Y_1,Y_2,\dots$ and $Y'_1,Y'_2,\dots$ have the same distribution. (In the argument below, conditioning by empty sequences means no conditioning. This will be the case only for $t = 1$.)

For all $t \geqslant 1$, we denote

$$X'_t = \varphi_t^{(\mathcal{X})}\big(Y'_1,\dots,Y'_{t-1}\big)\,.$$

Under ν^j and given $Y'_1,\dots,Y'_{t-1}$, the distribution of Y'_t is obtained by definition as the two-step random draw of $I'_t \sim T(X'_t)$ and then, conditionally on this first draw, $Y'_t \sim \nu^j_{I'_t}$. By the above results, the distribution of Y'_t is thus a Bernoulli distribution with parameter $\mu_j(X'_t)$.

At the same time, under $M_{x_j,\eta}$ and given $Y_1,\ldots,Y_{t-1}$, the choice of

$$X_t = \varphi_t^{(\mathcal{X})}\big(Y_1,\ldots,Y_{t-1}\big)$$

yields a reward Y_t distributed according to $M_{x_j,\eta}(X_t)$, that is, by definition and with the notations above, a Bernoulli distribution with parameter $f_{x_j,\eta}(X_t) = \mu_j(X_t)$.

The argument is concluded by induction and by using the fact that rewards are drawn independently in each round.

Fifth step. We summarize what we proved so far. For $\eta \in (0,1/2)$, provided that there exist $K \geqslant 2$ disjoint balls $\mathcal{B}(x_j,\eta)$ in $\mathcal{X}$, we could construct, for all strategies $\varphi^{(\mathcal{X})}$ for the $\mathcal{X}$-armed bandit problem, a strategy $\psi^{(K+1)}$ for the $\{1,\ldots,K+1\}$-armed bandit problem such that, for all $j=1,\ldots,K$ and all $n \geqslant 1$,

$$\mathbb{E}_{M_{x_j,\eta}}\big[R_n(\varphi^{(\mathcal{X})})\big] = \mathbb{E}_{\nu^j}\big[R_n(\psi^{(K+1)})\big]\,.$$

But by the assumption on the packing dimension, there exists $c>0$ such that for all $\eta < 1/2$, the choice of $K_\eta = \lceil c\,\eta^{-D}\rceil \geqslant 2$ guarantees the existence of such K_η disjoint balls. Substituting this value, and using the results of the first and fourth steps of the proof, we get

$$\max_{j=1,\ldots,K_\eta} \mathbb{E}_{M_{x_j,\eta}}\big[R_n(\varphi^{(\mathcal{X})})\big] = \max_{j=1,\ldots,K_\eta} \mathbb{E}_{\nu^j}\big[R_n(\psi^{(K+1)})\big] \geqslant n\eta\left(1-\frac{1}{K_\eta}-\eta\sqrt{4\ln(4/3)}\sqrt{\frac{n}{K_\eta}}\right).$$

The proof is concluded by noting that

- the left-hand side is smaller than the maximal regret w.r.t. all weak Lipschitz environments;
- the right-hand side can be lower bounded and then optimized over $\eta < 1/2$ in the following way.

By definition of K_η and the fact that it is larger than 2, one has

$$\begin{aligned} &n\eta\left(1-\frac{1}{K_\eta}-\eta\sqrt{4\ln(4/3)}\sqrt{\frac{n}{K_\eta}}\right) \\ &\qquad\qquad \geqslant n\eta\left(1-\frac{1}{2}-\eta\sqrt{4\ln(4/3)}\sqrt{\frac{n}{c\eta^{-D}}}\right) = n\eta\left(\frac{1}{2}-C\eta^{1+D/2}\sqrt{n}\right)\end{aligned}$$

where $C=\sqrt{\big(4\ln(4/3)\big)/c}$. We can optimize the final lower bound over $\eta \in [0,\,1/2]$.

To that end, we choose, for instance, η such that $C\eta^{1+D/2}\sqrt{n} = 1/4$, that is,

$$\eta = \left(\frac{1}{4C\sqrt{n}}\right)^{1/(1+D/2)} = \left(\frac{1}{4C}\right)^{1/(1+D/2)} n^{-1/(D+2)}\,.$$

This gives the lower bound

$$\frac{1}{4}\left(\frac{1}{4C}\right)^{1/(1+D/2)} n^{1-1/(D+2)} = \underbrace{\frac{1}{4}\left(\frac{1}{4C}\right)^{1/(1+D/2)}}_{=\gamma(c,D)} n^{(D+1)/(D+2)}\,.$$

To ensure that this choice of η is valid we need to show that $\eta \leqslant 1/2$. Since the latter requirement is equivalent to

$$n \geqslant \left(2\left(\frac{1}{4C}\right)^{1/(1+D/2)}\right)^{D+2},$$

it suffices to choose the right-hand side to be $N(c,D)$; we then get that $\eta \leqslant 1/2$ indeed holds for all $n \geqslant N(c,D)$, thus concluding the proof of the theorem. ■

References

J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: an efficient algorithm for bandit linear optimization. In *Proceedings of the 21st International Conference on Learning Theory*. Omnipress, 2008.

R. Agrawal. Sample mean based index policies with $o(\log n)$ regret for the multi-armed bandit problem. *Advances in Applied Mathematics*, 27:1054–1078, 1995a.

R. Agrawal. The continuum-armed bandit problem. *SIAM Journal on Control and Optimization*, 33:1926–1951, 1995b.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning Journal*, 47(2-3):235–256, 2002a.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002b.

P. Auer, R. Ortner, and C. Szepesvári. Improved rates for the stochastic continuum-armed bandit problem. In *Proceedings of the 20th Conference on Learning Theory*, pages 454–468, 2007.

S. Bubeck and R. Munos. Open loop optimistic planning. In *Proceedings of the 23rd International Conference on Learning Theory*. Omnipress, 2010.

S. Bubeck, R. Munos, G. Stoltz, and Cs. Szepesvari. Online optimization in $\mathcal{X}$–armed bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 201–208, 2009.

S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. *Theoretical Computer Science*, 412:1832–1852, 2011.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

G.M.J. Chaslot, M.H.M. Winands, H. Herik, J. Uiterwijk, and B. Bouzy. Progressive strategies for Monte-Carlo tree search. *New Mathematics and Natural Computation*, 4(3):343–357, 2008.

E. Cope. Regret and convergence bounds for immediate-reward reinforcement learning with continuous action spaces. *IEEE Transactions on Automatic Control*, 54(6):1243–1253, 2009.

P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 67–74, 2007.

J. L. Doob. *Stochastic Processes*. John Wiley & Sons, 1953.

H. Finnsson and Y. Bjornsson. Simulation-based approach to general game playing. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 259–264, 2008.

S. Gelly and D. Silver. Combining online and offline knowledge in UCT. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM New York, NY, USA, 2007.

S. Gelly and D. Silver. Achieving master level play in $9\times$ 9 computer go. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1537–1540, 2008.

S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with patterns in Monte-Carlo go. Technical Report RR-6062, INRIA, 2006.

J. C. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley-Interscience Series in Systems and Optimization. Wiley, Chichester, NY, 1989.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

R. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems 18*, 2004.

R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the 40th ACM Symposium on Theory of Computing*, 2008a.

R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces, September 2008b. URL `http://arxiv.org/abs/0809.4882`.

L. Kocsis and Cs. Szepesvari. Bandit based Monte-carlo planning. In *Proceedings of the 15th European Conference on Machine Learning*, pages 282–293, 2006.

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society*, 58:527–535, 1952.

M.P.D. Schadd, M.H.M. Winands, H.J. van den Herik, and H. Aldewereld. Addressing NP-complete puzzles with Monte-Carlo methods. In *Proceedings of the AISB 2008 Symposium on Logic and the Simulation of Interaction and Reasoning*, volume 9, pages 55—61. The Society for the study of Artificial Intelligence and Simulation of Behaviour, 2008.

Y. Yang. How powerful can any regression learning procedure be? In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2, pages 636–643, 2007.

Journal of Machine Learning Research 12 (2011) 1697-1728 Submitted 4/10; Revised 10/10; Published 5/11

Domain Decomposition Approach for Fast Gaussian Process Regression of Large Spatial Data Sets

Chiwoo Park CHIWOO.PARK@TAMU.EDU
Department of Industrial and Systems Engineering
Texas A&M University
3131 TAMU, College Station, TX 77843-3131, USA

Jianhua Z. Huang JIANHUA@STAT.TAMU.EDU
Department of Statistics
Texas A&M University
3143 TAMU, College Station, TX 77843-3143, USA

Yu Ding YUDING@IEMAIL.TAMU.EDU
Department of Industrial and Systems Engineering
Texas A&M University
3131 TAMU, College Station, TX 77843-3131, USA

Editor: Neil Lawrence

Abstract

Gaussian process regression is a flexible and powerful tool for machine learning, but the high computational complexity hinders its broader applications. In this paper, we propose a new approach for fast computation of Gaussian process regression with a focus on large spatial data sets. The approach decomposes the domain of a regression function into small subdomains and infers a local piece of the regression function for each subdomain. We explicitly address the mismatch problem of the local pieces on the boundaries of neighboring subdomains by imposing continuity constraints. The new approach has comparable or better computation complexity as other competing methods, but it is easier to be parallelized for faster computation. Moreover, the method can be adaptive to non-stationary features because of its local nature and, in particular, its use of different hyperparameters of the covariance function for different local regions. We illustrate application of the method and demonstrate its advantages over existing methods using two synthetic data sets and two real spatial data sets.

Keywords: domain decomposition, boundary value problem, Gaussian process regression, parallel computation, spatial prediction

1. Introduction

This paper is concerned about fast computation of Gaussian process regression, hereafter called GP regression. With its origin in geostatistics, well known as *kriging*, the GP regression has recently developed to be a useful tool in machine learning (Rasmussen and Williams, 2006). A GP regression provides the best unbiased linear estimator computable by a simple closed form expression and is a popular method for interpolation or extrapolation. A major limitation of GP regression is its computational complexity, scaled by $O(N^3)$, where N is the number of training observations. Many approximate computation methods have been introduced in the literature to relieve the computation burden. A new computation scheme is developed in this paper with a focus on large spatial data sets.

Existing approximation methods may be categorized into three schools: matrix approximation, likelihood approximation and localized regression. The first school is motivated by the observation that the inversion of a big covariance matrix is the major part of the expensive computation, and thus, approximating the matrix by a lower rank version will help reduce the computational demand. Williams and Seeger (2000) approximated the covariance matrix by the Nyström extension of a smaller covariance matrix evaluated on M training observations ($M \ll N$). This helps reduce the computation cost from $O(N^3)$ to $O(NM^2)$, but this method does not guarantee the positivity of the prediction variance (Quiñonero-Candela and Rasmussen, 2005, page 1954).

The second school approximates the likelihood of training and testing points by assuming conditional independence of training and testing points, given M artificial points, known as "*inducing inputs*." Under this assumption, one only needs to invert matrices of rank M for GP predictions rather than the original big matrix of rank N. Depending on the specific independence assumed, there are a number of variants to the approach: deterministic training conditional (DTC, Seeger et al., 2003), full independent conditional (FIC, Snelson and Ghahramani, 2006), partial independent conditional (PIC, Snelson and Ghahramani, 2007). DTC assumes a deterministic relation between the inducing inputs and the regression function values at training sample locations. An issue in DTC is how to choose the inducing inputs; a greedy strategy has been used to choose the inducing inputs among the training data. FIC assumes that each individual training or test point is conditionally independent of one another once given all the inducing inputs. Under this assumption, FIC enjoys a reduced computation cost of $O(NM^2)$ for training and $O(M^2)$ for testing. However, FIC will have difficulty in fitting data having abrupt local changes or non-stationary features; see Snelson and Ghahramani (2007). PIC makes a relaxed conditional independence assumption in order to better reflect localized features. PIC first groups all data points into several blocks and assumes that all the data points within a block could still be dependent but the data points between blocks are conditional independent once given the inducing inputs. Suppose that B is the number of data points in a block, PIC entertains a reduced computation cost of $O(N(M+B)^2)$ for training and $O((M+B)^2)$ for testing.

The last school is localized regression. It starts from the belief that a pair of observations far away from each other are almost uncorrelated. As such, prediction at a test location can be performed by using only a small number, say B, of neighboring points. One way to implement this idea, called *local kriging*, is to decompose the entire domain into smaller subdomains and to predict at a test site using the training points only in the subdomain which the test site belongs to. It is well known that local kriging suffers from having discontinuities in prediction on the boundaries of subdomains. On the other hand, the local kriging enjoys many advantages, such as adaptivity to non-stationary changes, efficient computation with $O(NB^2)$ operations for training and $O(B^2)$ for testing, and easiness of being parallelized for faster computation.

Another way for localized regression is to build multiple local predictors and to combine them by taking a weighted average of the local predictions. Differing in the weighting schemes used, several methods have been proposed in the literature, including Bayesian committee machine (BCM, Tresp, 2000; Schwaighofer et al., 2003), local probabilistic regression (LPR, Urtasun and Darrell, 2008), mixture of Gaussian process experts (MGP, Rasmussen and Ghahramani, 2002), and treed Gaussian process model (TGP, Gramacy and Lee, 2008). Because of the averaging mechanism, all these methods avoid the discontinuity problem of local kriging. However, the testing time complexities of all these methods are significantly higher than local kriging, making them less competitive when the number of test locations is large. In particular, BCM is transductive and it requires inversion of a matrix whose size is the same as the number of test locations, and as such, it is very slow when the number of test locations is large. Mixture models such as MGP and TGP involve complicated integrations which in turn are approximated by sampling or Monte Carlo simulation. The use of Monte Carlo simulation makes these methods less effective for large data sets.

Being aware of the advantages and disadvantages of the local kriging along with computational limitations of the averaging-based localized regression, we propose a new local kriging approach that explicitly addresses the problem of discontinuity in prediction on the boundaries of subdomains. The basic idea is to formulate the GP regression as an optimization problem and to decompose the optimization problem into smaller local optimization problems that provide local predictions. By imposing continuity constraints on the local predictions at the boundaries, we are able to produce a continuous global prediction for 1-d data and significantly control the degrees of discontinuities for 2-d data. Our new local kriging approach is motivated by the domain decomposition method widely used for solving partial differential equations (PDE, Quarteroni and Valli, 1999). To obtain a numerical solution of a PDE, the finite element method discretizes the problem and approximates the PDE by a big linear system whose computation cost grows with the number of discretizing points over the big domain. In order to attain an efficient solution, the domain decomposition method decomposes the domain of the PDE solution into small pieces, solves small linear systems for local approximations of the PDE solution, and smoothly connects the local approximations through imposing continuity and smoothness conditions on boundaries.

Our method has, in a regular (sequential) computing environment, at least similar computational complexity as the most efficient existing methods such as FIC, PIC, BCM, and LPR, but it can be parallelized easily for faster computation, resulting a much reduced computational cost of $O(B^3)$. Furthermore, each local predictor in our approach is allowed to use a different hyperparameter for the covariance function and thus the method is adaptive to non-stationary changes in the data, a feature not enjoyed by FIC and PIC. The averaging-based localized regressions also allow local hyperparameters, but our method is computationally more attractive for large test data sets. Overall, our approach achieves a good balance of computational speed and accuracy, as demonstrated empirically using synthetic and real spatial data sets (Section 6). Methods applying a compactly supported covariance function (Gneiting, 2002; Furrer et al., 2006) can be considered as a variant of localized regression, which essentially uses *moving boundaries* to define neighborhoods. These methods can produce continuous predictions but cannot be easily modified to adapt to non-stationarity.

The rest of the paper is organized as follows. In Sections 2 and 3, we formulate the new local kriging as a constrained optimization problem and provide solution approaches for the optimization problem. Section 4 presents the numerical algorithm of our method. Section 5 discusses the hyperparameter learning issue. Section 6 provides numerical comparisons of our method with several existing methods, including local kriging, FIC, PIC, BCM, and LPR, using two synthetic data sets (1-d and 2-d) and two real data sets (both 2-d). Finally Section 7 concludes the paper, followed by additional discussions on possible improvement of the proposed method.

2. GP Regression as an Optimization Problem

Before formulating the problem, we define notational convention. Boldface capital letters represent matrices and boldface lowercase letters represent vectors. One exception is a notation for spatial locations. A spatial location is a two-dimensional vector, but for notational simplicity, we do not use boldface for it. Instead, we use boldface lowercase to represent a set of spatial locations.

A spatial GP regression is usually formulated as follows: given a training data set $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, N\}$ of n pairs of locations x_i and noisy observations y_i, obtain the predictive distribution for the realization of a latent function at a test location x_*, denoted by $f_* = f(x_*)$. We assume that the latent function comes from a zero-mean Gaussian random field with a covariance function $k(\cdot, \cdot)$ on a domain $\Omega \subset \mathcal{R}^d$ and the noisy observations y_i are given by

$$y_i = f(x_i) + \varepsilon_i, \qquad i = 1, \ldots, N,$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ are white noises independent of $f(x_i)$. Denote $\mathbf{x} = [x_1, x_2, .., x_N]^t$ and $\boldsymbol{y} = [y_1, y_2, .., y_N]^t$. The joint distribution of $(f_*, \boldsymbol{y})$ is

$$P(f_*, \boldsymbol{y}) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k_{**} & \boldsymbol{k}_{\mathbf{x}*}^t \\ \boldsymbol{k}_{\mathbf{x}*} & \sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}} \end{bmatrix}\right),$$

where $k_{**} = k(x_*, x_*)$, $\boldsymbol{k}_{\mathbf{x}*} = (k(x_1, x_*), \ldots, k(x_N, x_*))^t$ and $\boldsymbol{K}_{\mathbf{xx}}$ is an $N \times N$ matrix with $(i, j)^{th}$ entity $k(x_i, x_j)$. The subscripts of $k_{**}, \boldsymbol{k}_{\mathbf{x}*}$ and $\boldsymbol{K}_{\mathbf{xx}}$ represent two sets of the locations between which the covariance is computed, and x_* is abbreviated as $*$. By the conditional distribution for Gaussian variables, the predictive distribution of f_* given $\boldsymbol{y}$ is

$$P(f_*|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{k}_{\mathbf{x}*}^t (\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1} \boldsymbol{y}, k_{**} - \boldsymbol{k}_{\mathbf{x}*}^t (\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1} \boldsymbol{k}_{\mathbf{x}*}). \tag{1}$$

The predictive mean $\boldsymbol{k}_{\mathbf{x}*}^t (\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1} \boldsymbol{y}$ gives the point prediction of $f(x)$ at location x_*, whose uncertainty is measured by the predictive variance $k_{**} - \boldsymbol{k}_{\mathbf{x}*}^t (\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1} \boldsymbol{k}_{\mathbf{x}*}$.

The point prediction given above is the best linear unbiased predictor (BLUP) in the following sense. Consider all linear predictors

$$\mu(x_*) = \boldsymbol{u}(x_*)^t \boldsymbol{y},$$

satisfying the unbiasedness requirement $E[\mu(x_*)] = 0$. We want to find the vector $\boldsymbol{u}(x_*)$ that minimizes the mean squared prediction error $E[\mu(x_*) - f(x_*)]^2$. Since $E[\mu(x_*)] = 0$ and $E[f(x_*)] = 0$, the mean squared prediction error equals the error variance $\mathrm{var}[\mu(x_*) - f(x_*)]$ and can be expressed as

$$\begin{aligned} \sigma(x_*) &= \boldsymbol{u}(x_*)^t E(\boldsymbol{y}\boldsymbol{y}^t) \boldsymbol{u}(x_*) - 2\boldsymbol{u}(x_*)^t E(\boldsymbol{y} f_*) + E(f_*^2) \\ &= \boldsymbol{u}(x_*)^t (\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}}) \boldsymbol{u}(x_*) - 2\boldsymbol{u}(x_*)^t \boldsymbol{k}_{\mathbf{x}*} + k_{**}, \end{aligned} \tag{2}$$

which is a quadratic form in $\boldsymbol{u}(x_*)$. It is easy to see $\sigma(x_*)$ is minimized if and only if $\boldsymbol{u}(x_*)$ is chosen to be $(\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1} \boldsymbol{k}_{\mathbf{x}*}$.

Based on the above discussion, the mean of the predictive distribution in (1) or the best linear unbiased predictor can be obtained by solving the following minimization problem: for $x_* \in \Omega$,

$$\underset{\boldsymbol{u}(x_*) \in \mathbb{R}^N}{\text{Minimize}} \quad z[\boldsymbol{u}(x_*)] := \boldsymbol{u}(x_*)^t (\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}}) \boldsymbol{u}(x_*) - 2\boldsymbol{u}(x_*)^t \boldsymbol{k}_{\mathbf{x}*}, \tag{3}$$

where the constant term k_{**} in $\sigma(x_*)$ is dropped from the objective function. Given the solution $\boldsymbol{u}(x_*) = (\sigma^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1} \boldsymbol{k}_{\mathbf{x}*}$, the predictive mean is given by $\boldsymbol{u}(x_*)^t \boldsymbol{y}$ and the predictive variance is $z[\boldsymbol{u}(x_*)] + k_{**}$, the optimal objective value plus the variance of f_* at the location x_*.

The optimization problem (3) and its solution depends on the special location x_* that we seek prediction. However, we are usually interested in predicting at multiple test locations. This motivates us to define another class of optimization problem whose solutions are independent of the location. Note that the optimal solution of (3) has the form of $\boldsymbol{A}\boldsymbol{k}_{\mathbf{x}*}$ for an $N \times N$ matrix $\boldsymbol{A}$ and thus, we can restrict the feasible region for $\boldsymbol{u}(x_*)$ to $\mathbb{V} = \{\boldsymbol{A}\boldsymbol{k}_{\mathbf{x}*}; \boldsymbol{A} \in \mathbb{R}^{N\times N}\}$. This leads to the following optimization problem: for $x_* \in \Omega$,

$$\underset{\boldsymbol{A}\in\mathbb{R}^{N\times N}}{\text{Minimize}} \quad z[\boldsymbol{A}] := \boldsymbol{k}_{\mathbf{x}*}^t \boldsymbol{A}^t(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})\boldsymbol{A}\boldsymbol{k}_{\mathbf{x}*} - 2\boldsymbol{k}_{\mathbf{x}*}^t\boldsymbol{A}^t\boldsymbol{k}_{\mathbf{x}*}. \tag{4}$$

The first order necessary condition (FONC) for solving (4) is

$$\frac{dz[\boldsymbol{A}]}{d\boldsymbol{A}} = 2(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})\boldsymbol{A}\boldsymbol{k}_{\mathbf{x}*}\boldsymbol{k}_{\mathbf{x}*}^t - 2\boldsymbol{k}_{\mathbf{x}*}\boldsymbol{k}_{\mathbf{x}*}^t = 0. \tag{5}$$

To obtain $\boldsymbol{A}$, we need $N \times N$ equations with respect to $\boldsymbol{A}$. However, the FONC only provides N equations since $\boldsymbol{k}_{\mathbf{x}*}\boldsymbol{k}_{\mathbf{x}*}^t$ is a rank one matrix, and thus it cannot uniquely determine the optimal $\boldsymbol{A}$. In fact, $\boldsymbol{A} = (\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}(\boldsymbol{I} + \boldsymbol{B})$, where $\boldsymbol{B}$ is any matrix satisfying $\boldsymbol{B}\boldsymbol{k}_{\mathbf{x}*} = 0$, all satisfies the FONC in (5).

Recall that our intention of the reformulation is to produce location-independent solutions. Yet those $\boldsymbol{A}$'s satisfying the FONC as mentioned above are still dependent on x_*, except for $\widehat{\boldsymbol{A}} = (\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}$, which becomes the one we propose to choose as the solution to the FONC. It is also easy to verify that $\widehat{\boldsymbol{A}} = (\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{xx}})^{-1}$ is indeed the optimal solution to (4). The formulation (4) and the above-mentioned rationale for choosing its solution will serve as the basis for the development of local kriging in the next section.

3. Domain Decomposition: Globally Connected Local Kriging

The reformulation of the GP regression as the optimization problem in (4) does not reduce the computational complexity. We still need to compute the matrix inversion in $\widehat{\boldsymbol{A}}$ which requires $O(N^3)$ computations. To ease the computational burden, our strategy is to approximate the optimization problem (4) by a collection of local optimization problems, each of which is computationally cheap to solve. The local optimization problems are connected in a way ensuring that the spatial prediction function is globally continuous. We present the basic idea in this section and derive the numerical algorithm in the next section.

We decompose the domain Ω into m disjoint subdomains $\{\Omega_j\}_{j=1,\dots,m}$. Let $\mathbf{x}_j$ be the subset of locations of observed data that belong to Ω_j and let $\boldsymbol{y}_j$ denote the corresponding values of the response variable. Denote by n_j the number of data points in Ω_j. Consider an initial local problem as follows: for $x_* \in \Omega_j$,

$$\underset{\boldsymbol{A}_j\in\mathbb{R}^{n_j\times n_j}}{\text{Minimize}} \quad \boldsymbol{k}_{\mathbf{x}_j*}^t\boldsymbol{A}_j^t(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} - 2\boldsymbol{k}_{\mathbf{x}_j*}^t\boldsymbol{A}_j^t\boldsymbol{k}_{\mathbf{x}_j*}, \tag{6}$$

where we introduced the subdomain-dependent noise variance σ_j^2. The minimizer $\boldsymbol{A}_j = (\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}$ provides a local predictor, $\mu_j(x_*) = \boldsymbol{k}_{\mathbf{x}_j*}^t\boldsymbol{A}_j^t\boldsymbol{y}_j$, for $x_* \in \Omega_j$. Computing the local predictor requires only $O(n_j^3)$ operations for each j. By making $n_j \ll N$, the saving in computation could be substantial.

As we mentioned in the introduction, the above local kriging will suffer from discontinuities in prediction on boundaries of subdomains. While the prediction on the interior of each subdomain is independently governed by the corresponding local predictor, the prediction on a boundary comes from the local predictors of at least two subdomains that intersect on the boundary, which provide different predictions. For simplicity, in this paper, we suppose that a boundary is shared by at most two subdomains. In the language of finite element analysis, our subdomains $\{\Omega_j\}_{j=1,\ldots,m}$ form a '*conforming*' mesh of the domain Ω (Ern and Guermond, 2004). Suppose that two neighboring subdomains Ω_j and Ω_k have a common boundary $\Gamma_{jk} := \overline{\Omega}_j \cap \overline{\Omega}_k$, where $\overline{\Omega}_j$ means the closure of Ω_j. Using $\boldsymbol{k}_{\mathbf{x}_j\circ}$ as the abbreviation of $\boldsymbol{k}_{\mathbf{x}_j x_\circ}$, we have discontinuities on Γ_{jk}, that is,

$$\boldsymbol{k}^t_{\mathbf{x}_j\circ}\boldsymbol{A}^t_j\boldsymbol{y}_j \neq \boldsymbol{k}^t_{\mathbf{x}_k\circ}\boldsymbol{A}^t_k\boldsymbol{y}_k \text{ for } x_\circ \in \Gamma_{jk}.$$

The discontinuity problem of local kriging has been well documented in the literature; see Snelson and Ghahramani (2007, Figure 1).

To fix the problem, we impose continuity constraints on subdomain boundaries when combining the local predictors. Specifically, we impose

$$\text{(Continuity)} \quad \boldsymbol{k}^t_{\mathbf{x}_j\circ}\boldsymbol{A}^t_j\boldsymbol{y}_j = \boldsymbol{k}^t_{\mathbf{x}_k\circ}\boldsymbol{A}^t_k\boldsymbol{y}_k \text{ for } x_\circ \in \Gamma_{jk}.$$

This continuity requirement implies that two mean predictions obtained from local predictors of two neighboring subdomains are the same on the common subdomain boundary. According to (2), the predictive variance is in a quadratic form of the predictive mean. Thus, the continuity of the predictive mean across boundary imply the continuity of the predictive variance.

To incorporate the continuity condition to the local kriging problems, define $r_{jk}(x_\circ)$ as a consistent prediction at $x_\circ$ on Γ_{jk}. The continuity condition is converted to the following two separate conditions:

$$\boldsymbol{k}^t_{\mathbf{x}_j\circ}\boldsymbol{A}^t_j\boldsymbol{y}_j = r_{jk}(x_\circ) \text{ and } \boldsymbol{k}^t_{\mathbf{x}_k\circ}\boldsymbol{A}^t_k\boldsymbol{y}_k = r_{jk}(x_\circ) \text{ for } x_\circ \in \Gamma_{jk}.$$

Adding these conditions as constraints, we revise the initial local problem (6) to the following constrained local problem: for $x_* \in \Omega_j$

$$\begin{aligned}\textbf{LP(j)}: \quad \underset{\boldsymbol{A}_j\in\mathbb{R}^{n_j\times n_j}}{\text{Minimize}} \quad & \boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} - 2\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j\boldsymbol{k}_{\mathbf{x}_j*} \\ \text{s.t.} \quad & \boldsymbol{k}^t_{\mathbf{x}_j\circ}\boldsymbol{A}^t_j\boldsymbol{y}_j = r_{jk}(x_\circ) \quad \text{for } x_\circ \in \Gamma_{jk} \text{ and } k \in N(j),\end{aligned} \tag{7}$$

where $N(j) := \{k : \Omega_k \text{ is next to } \Omega_j\}$. Note that $r_{jk}(x_\circ)$ is a function of $x_\circ$ and is referred to as a *boundary value function* on Γ_{jk}. We ensure the continuity of the prediction across the boundary Γ_{jk} by using a common boundary value function r_{jk} for two neighboring subdomains Ω_j and Ω_k. Solving of the constrained local problem **LP(j)** will be discussed in the subsequent section. Since the solution depends on a set of r_{jk}'s, denoted collectively as $r_j = \{r_{jk}; k \in N(j)\}$, we denote the solution of (7) as $\boldsymbol{A}_j(r_j)$. Note that, if r_j is given, we can solve the constrained local problem **LP(j)** for each subdomain independently. In reality, r_j is unknown unless additional conditions are used.

To obtain the boundary value functions, we propose to minimize the predictive variances on the subdomain boundaries. The predictive variance at a boundary point $x_\circ$ is given by the objective function of (7), which depends on r_j and can be written as

$$\boldsymbol{k}^t_{\mathbf{x}_j\circ}\boldsymbol{A}_j(r_j)^t(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j(r_j)\boldsymbol{k}_{\mathbf{x}_j\circ} - 2\boldsymbol{k}^t_{\mathbf{x}_j\circ}\boldsymbol{A}_j(r_j)^t\boldsymbol{k}_{\mathbf{x}_j\circ}.$$

To obtain the collection of all boundary value functions, $\{r_j\}^m_{j=1}$, we solve the following optimization problem

$$\underset{\{r_j\}^m_{j=1}}{\text{Minimize}} \quad \sum_{j=1}^m\sum_{k\in N(j)}\sum_{x_\circ\in\Gamma_{jk}} \boldsymbol{k}^t_{\mathbf{x}_j\circ}\boldsymbol{A}_j(r_j)^t(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j(r_j)\boldsymbol{k}_{\mathbf{x}_j\circ} - 2\boldsymbol{k}^t_{\mathbf{x}_j\circ}\boldsymbol{A}_j(r_j)^t\boldsymbol{k}_{\mathbf{x}_j\circ}. \tag{8}$$

Note that we cannot solve optimization over each r_j separately since r_{jk} in r_j is equivalent to r_{kj} in r_k so the optimization over r_j is essentially linked to the optimization over r_k. In other words, the equations for obtaining the optimized r_j's are entangled. We call (8) an *interface equation*, since it solves the boundary values on the interfaces between subdomains. Details of solving these equations will be given in the next section.

To summarize, we reformulate the spatial prediction problem as a collection of local prediction optimization problems, and impose continuity restrictions to these local problems. Our solution strategy is to first solve the interface equations to obtain the boundary value functions, and then to solve the constrained local problems to build the globally connected local predictors. The implementation of this basic idea is given in the next section.

4. Numerical Algorithm Based on Domain Decomposition

To solve (7) and (8) numerically, we make one simplification that restricts the boundary value functions r_{jk}'s to be polynomials of a certain degree. Since we want our predictions to be continuous and smooth, and polynomials are dense in the space of continuous functions, such restriction to polynomials does not sacrifice much accuracy. To facilitate computation, we use Lagrange basis polynomials as the basis functions to represent the boundary value functions.

Suppose that we use p Lagrange basis polynomials defined at p interpolation points that are uniformly spaced on Γ_{jk}. We refer to p as the *degrees of freedom*. Let $\boldsymbol{r}_{jk}$ be a $p \times 1$ vector that denotes the boundary function r_{jk} evaluated at the p interpolation points. Then $r_{jk}(x_\circ)$ can be written as a linear combination

$$r_{jk}(x_\circ) = \boldsymbol{T}_{jk}(x_\circ)^t \boldsymbol{r}_{jk}, \tag{9}$$

where $\boldsymbol{T}_{jk}(x_\circ)$ is a $p \times 1$ vector with the values of p Lagrange basis polynomials at $x_\circ$ as its elements. Plugging (9) into (7), the local prediction problem becomes for $x_* \in \Omega_j$,

$$\begin{aligned} \textbf{LP(j)}: \quad & \underset{\boldsymbol{A}_j \in \mathbb{R}^{n_j \times n_j}}{\text{Minimize}} \quad \boldsymbol{k}_{\mathbf{x}_j *}^t \boldsymbol{A}_j^t (\sigma_j^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j \mathbf{x}_j}) \boldsymbol{A}_j \boldsymbol{k}_{\mathbf{x}_j *} - 2 \boldsymbol{k}_{\mathbf{x}_j *}^t \boldsymbol{A}_j^t \boldsymbol{k}_{\mathbf{x}_j *} \\ & \text{s.t.} \quad \boldsymbol{k}_{\mathbf{x}_j \circ}^t \boldsymbol{A}_j^t \boldsymbol{y}_j = \boldsymbol{T}_{jk}(x_\circ)^t \boldsymbol{r}_{jk} \quad \text{for } x_\circ \in \Gamma_{jk} \text{ and } k \in N(j). \end{aligned} \tag{10}$$

Since the constraint in (10) must hold for all points on Γ_{jk}, there are infinite number of constraints to check. One way to handle these constraints is to merge the infinitely many constraints into one constraint by considering the following integral equation:

$$\int_{\Gamma_{jk}} [\boldsymbol{k}_{\mathbf{x}_j \circ}^t \boldsymbol{A}_j^t \boldsymbol{y}_j - \boldsymbol{T}_{jk}(x_\circ)^t \boldsymbol{r}_{jk}]^2 dx_\circ = 0.$$

The integral depends on the covariance function used and is usually intractable for general covariance functions. Even when the integral has a closed form expression, the expression can be too complicated to ensure a simple solution to the constrained optimization. Consider, for example, the covariance function is a squared exponential covariance function. In this case, the integration can be written as a combination of Gaussian error functions, but still we could not easily have the first order optimal solution for $\boldsymbol{A}$ with the integral constraint. We thus adopt another simplification, which is to check the constraint only at q uniformly spaced points on Γ_{jk}; these constraint-checking points on a boundary are referred to as *control points*. Although this approach does not guarantee that the continuity constraint is met at all points on Γ_{jk}, we find that the difference of $\boldsymbol{k}_{\mathbf{x}_j \circ}^t \boldsymbol{A}_j^t \boldsymbol{y}_j$ and

$r_{jk}(x_\circ)$ is small for all $x_\circ$ on Γ_{jk} when q is chosen to be reasonably large; see Section 6.2 for some empirical evidence.

Specifically, let $\mathbf{x}^b_{jk}$ denote the q uniformly spaced points on Γ_{jk}. Evaluate $\boldsymbol{k}_{\mathbf{x}_j\circ}$ and $\boldsymbol{T}_{jk}(x_\circ)$ when $x_\circ$ is taken to be an element of $\mathbf{x}^b_{jk}$ and denote the results collectively as the $n_j \times q$ matrix $\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}^b_{jk}}$ and the $q \times p$ matrix $\boldsymbol{T}_{jk}$, respectively. Then, the continuity constraints at the q points are expressed as follows: for $x_* \in \Omega_j$,

$$\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_{jk}}\boldsymbol{A}^t_j\boldsymbol{y}_j = \boldsymbol{T}^t_{jk}\boldsymbol{r}_{jk}.$$

Consequently, the optimization problem (10) can be rewritten as: for $x_* \in \Omega_j$,

$$\begin{aligned} \mathbf{LP(j)}': \quad \underset{\boldsymbol{A}_j\in\mathbb{R}^{n_j\times n_j}}{\text{Minimize}} \quad & \boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} - 2\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j\boldsymbol{k}_{\mathbf{x}_j*} \\ \text{s.t.} \quad & \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_{jk}}\boldsymbol{A}^t_j\boldsymbol{y}_j = \boldsymbol{T}^t_{jk}\boldsymbol{r}_{jk} \quad \text{for } k \in N(j). \end{aligned} \tag{11}$$

Introducing Lagrange multipliers $\boldsymbol{\lambda}_{jk}(x_*)$ (a $q \times 1$ vector), the problem becomes an unconstrained problem to minimize the Lagrangian: for $x_* \in \Omega_j$,

$$\begin{aligned} L(\boldsymbol{A}_j, \boldsymbol{\lambda}_{jk}(x_*)) := \boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} - 2\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j\boldsymbol{k}_{\mathbf{x}_j*} \\ - \sum_{k\in N(j)} \boldsymbol{\lambda}_{jk}(x_*)^t[\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_{jk}}\boldsymbol{A}^t_j\boldsymbol{y}_j - \boldsymbol{T}^t_{jk}\boldsymbol{r}_{jk}]. \end{aligned} \tag{12}$$

Let $\boldsymbol{\lambda}_j(x_*)$ denote a $q_j \times 1$ vector formed by stacking those $\boldsymbol{\lambda}_{jk}(x_*)$'s for $k \in N(j)$ where $q_j := q|N(j)|$. Let $\mathbf{x}^b_j$ denote the collection of $\mathbf{x}^b_{jk}$ for all $k \in N(j)$. We form $\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}^b_j}$ by columnwise binding $\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}^b_{jk}}$ and form $\boldsymbol{T}^t_j\boldsymbol{r}_j$ by row-wise binding $\boldsymbol{T}^t_{jk}\boldsymbol{r}_{jk}$. The Lagrangian becomes: for $x_* \in \Omega_j$,

$$\begin{aligned} L(\boldsymbol{A}_j, \boldsymbol{\lambda}_{jk}(x_*)) := \boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} - 2\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j\boldsymbol{k}_{\mathbf{x}_j*} \\ - \boldsymbol{\lambda}_j(x_*)^t[\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j}\boldsymbol{A}^t_j\boldsymbol{y}_j - \boldsymbol{T}^t_j\boldsymbol{r}_j]. \end{aligned}$$

The first order necessary conditions (FONC) for local optima are: for $x_* \in \Omega_j$,

$$\frac{d}{d\boldsymbol{A}_j}L(\boldsymbol{A}_j, \boldsymbol{\lambda}_{jk}) = 2(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*}\boldsymbol{k}^t_{\mathbf{x}_j*} - 2\boldsymbol{k}_{\mathbf{x}_j*}\boldsymbol{k}^t_{\mathbf{x}_j*} - \boldsymbol{y}_j\boldsymbol{\lambda}^t_j(x_*)\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j} = 0, \tag{13}$$

$$\frac{d}{d\boldsymbol{\lambda}_j}L(\boldsymbol{A}_j, \boldsymbol{\lambda}_j) = \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j}\boldsymbol{A}^t_j\boldsymbol{y}_j - \boldsymbol{T}^t_j\boldsymbol{r}_j = 0. \tag{14}$$

As in the unconstrained optimization case, that is, (4) and (5), the FONC (13) provides insufficient number of equations to uniquely determine the optimal $\boldsymbol{A}_j$ and $\boldsymbol{\lambda}_j(x_*)$. To see this, note that we have $n_j \times n_j$ unknowns from $\boldsymbol{A}_j$ and q_j unknowns from $\boldsymbol{\lambda}_j(x_*)$. Equation (13) provides only n_j distinguishable equations due to the matrix of rank one, $\boldsymbol{k}_{\mathbf{x}_j*}\boldsymbol{k}^t_{\mathbf{x}_j*}$, and Equation (14) adds q_j $(= q|N(j)|)$ linear equations. Thus, in order to find a sensible solution, we will follow our solution-choosing rationale stated in Section 2, which is to look for the location-independent solution.

To proceed, first, we change our target of obtaining the optimal $\boldsymbol{A}_j$ to an easier task of obtaining $\boldsymbol{u}(x_*) = \boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*}$, which is the quantity directly needed for the local predictor $\boldsymbol{u}(x_*)^t\boldsymbol{y}_j$. From Equation (13), we have that

$$\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} = (\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\left(\boldsymbol{k}_{\mathbf{x}_j*} + \frac{1}{2}\boldsymbol{y}_j\boldsymbol{\lambda}_j(x_*)^t\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j}(\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{k}_{\mathbf{x}_j*})^{-1}\boldsymbol{k}_{\mathbf{x}_j*}\right). \tag{15}$$

From here, the only thing that needs to be determined is the $q_j \times 1$ vector $\boldsymbol{\lambda}_j(x_*)$, which comes out from the Lagrangian (12). We have q_j equations from (14) to determine $\boldsymbol{\lambda}_j(x_*)$, so we restrict the solution space for $\boldsymbol{\lambda}_j(x_*)$ to a space fully specified by q_j unknowns. Specifically, we let $\boldsymbol{\lambda}_j(x_*)$ be proportional to $\boldsymbol{k}_{\mathbf{x}_j^b *}$, which is inversely related to the distance of x_* from the boundary points in $\mathbf{x}_j^b$. More precisely, we set $\boldsymbol{\lambda}_j(x_*) = \boldsymbol{\Lambda}_j(\boldsymbol{k}^t_{\mathbf{x}_j^b *}\boldsymbol{k}_{\mathbf{x}_j^b *})^{-1/2}\boldsymbol{k}_{\mathbf{x}_j^b *}$, where $(\boldsymbol{k}^t_{\mathbf{x}_j^b *}\boldsymbol{k}_{\mathbf{x}_j^b *})^{-1/2}$ is a scaling factor to normalize the vector $\boldsymbol{k}_{\mathbf{x}_j^b *}$ to unit length, and $\boldsymbol{\Lambda_j}$ is a $q_j \times q_j$ unknown diagonal matrix whose diagonal elements are collectively denoted as a column vector $\boldsymbol{\lambda}_j$. Note that the newly defined $\boldsymbol{\lambda}_j$ no longer depends on locations.

The optimal $\boldsymbol{\lambda}_j$ is obtained by using (15) to evaluate $\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*}$ at the q_j points $\mathbf{x}_{jk}^b$ ($k \in N(j)$) on the boundaries and then solving (14). The optimal solution is (derivation in Appendix A)

$$\begin{aligned}
\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} &= (\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1} \\
&\quad \left(\boldsymbol{k}_{\mathbf{x}_j*} + \frac{1}{2}\boldsymbol{y}_j\boldsymbol{\lambda}_j^t[(\boldsymbol{k}^t_{\mathbf{x}_j^b *}\boldsymbol{k}_{\mathbf{x}_j^b *})^{-1/2}\boldsymbol{k}_{\mathbf{x}_j^b *}] \circ [\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{k}_{\mathbf{x}_j*}(\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{k}_{\mathbf{x}_j*})^{-1}]\right), \\
\boldsymbol{\lambda}_j &= 2\boldsymbol{G}_j\frac{\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{y}_j}{\boldsymbol{y}_j^t(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{y}_j}, \\
\boldsymbol{G}_j^{-1} &= \{\mathrm{diag}_{1/2}[(\boldsymbol{K}^t_{\mathbf{x}_j^b\mathbf{x}_j^b}\boldsymbol{K}_{\mathbf{x}_j^b\mathbf{x}_j^b})^{-1}]\boldsymbol{K}_{\mathbf{x}_j^b\mathbf{x}_j^b}\} \circ \{\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}\mathrm{diag}[(\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b})^{-1}]\},
\end{aligned} \tag{16}$$

where $\boldsymbol{A} \circ \boldsymbol{B}$ is a Hadamard product of matrix $\boldsymbol{A}$ and $\boldsymbol{B}$, $\mathrm{diag}_{1/2}[\boldsymbol{A}]$ is a diagonal matrix with its diagonal elements the same as the square root of the diagonal elements of $\boldsymbol{A}$, and note that $\boldsymbol{G}_j^{-1}$ is symmetric. To simplify the expression, we define

$$\boldsymbol{h}_j := (\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{y}_j \text{ and } \bar{\boldsymbol{k}}_{\mathbf{x}_j^b *} := [(\boldsymbol{k}^t_{\mathbf{x}_j^b *}\boldsymbol{k}_{\mathbf{x}_j^b *})^{-1/2}\boldsymbol{k}_{\mathbf{x}_j^b *}] \circ [\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{k}_{\mathbf{x}_j*}(\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{k}_{\mathbf{x}_j*})^{-1}].$$

The optimal solution becomes

$$\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} = (\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\left(\boldsymbol{k}_{\mathbf{x}_j*} + \frac{\boldsymbol{y}_j(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{h}_j)^t\boldsymbol{G}_j}{\boldsymbol{y}_j^t\boldsymbol{h}_j}\bar{\boldsymbol{k}}_{\mathbf{x}_j^b *}\right). \tag{17}$$

It follows from (17) that the local mean predictor is

$$\hat{p}_j(x_*;\boldsymbol{r}_j) := \boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}_j^t\boldsymbol{y}_j = \boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{h}_j + \bar{\boldsymbol{k}}^t_{\mathbf{x}_j^b *}\boldsymbol{G}_j(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{h}_j), \tag{18}$$

for $x_* \in \Omega_j$. The local mean predictor is the sum of two terms: the first term, $\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{h}_j$, is the standard local kriging predictor without any boundary constraints; the second term is a scaled version of $\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{h}_j$, that is, the mismatches between the boundary value function and the unconstrained local kriging prediction. If x_* is one of the control points, then the local mean predictor given in (18) matches exactly the value given by the boundary value function.

The use of the local mean predictor in (18) relies on the knowledge of vector $\boldsymbol{r}_j$ which identifies $|N(j)|$ boundary value functions defined on $|N(j)|$ boundaries surrounding Ω_j. The $\boldsymbol{r}_j$ is equivalent to mean prediction (18) at $\mathbf{x}_j^b$. We choose the solution of $\boldsymbol{r}_j$ such that it minimizes the predictive variance at $\mathbf{x}_j^b$. The local predictive variance is computed by

$$\begin{aligned}\hat{\sigma}_j(x_*;\boldsymbol{r}_j) &= k_{**} + \boldsymbol{k}_{\mathbf{x}_j*}^t \boldsymbol{A}_j^t(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*} - 2\boldsymbol{k}_{\mathbf{x}_j*}^t\boldsymbol{A}_j^t\boldsymbol{k}_{\mathbf{x}_j*}\\ &= k_{**} - \boldsymbol{k}_{\mathbf{x}_j*}^t(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{k}_{\mathbf{x}_j*}\\ &\quad + \frac{\bar{\boldsymbol{k}}_{\mathbf{x}_j^b*}^t\boldsymbol{G}_j(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)^t\boldsymbol{G}_j\bar{\boldsymbol{k}}_{\mathbf{x}_j^b*}}{\boldsymbol{h}_j^t\boldsymbol{y}_j}.\end{aligned} \tag{19}$$

The second equality of (19) is obtained by plugging (17) into the first equality of (19). Since evaluating $\bar{\boldsymbol{k}}_{\mathbf{x}_j^b*}$ at each point in $\mathbf{x}_j^b$ and combining them in columnwise results in $\boldsymbol{G}_j^{-1}$, the predictive variances at $\mathbf{x}_j^b$ can be simplified as

$$\hat{\boldsymbol{\sigma}}_j(\mathbf{x}_j^b;\boldsymbol{r}_j) = \text{diag}_c[(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)^t]/(\boldsymbol{h}_j^t\boldsymbol{y}_j) + \text{constant},$$

where $\text{diag}_c[\boldsymbol{A}]$ is a column vector of the diagonal elements of matrix $\boldsymbol{A}$. Omitting a constant, the summation of the predictive variances at $\mathbf{x}_j^b$ is

$$\begin{aligned}S_j(\boldsymbol{r}_j) &= \mathbf{1}^t\text{diag}_c[(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)^t]/(\boldsymbol{h}_j^t\boldsymbol{y}_j)\\ &= \text{trace}[(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)^t]/(\boldsymbol{h}_j^t\boldsymbol{y}_j)\\ &= (\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)^t(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j)/(\boldsymbol{h}_j^t\boldsymbol{y}_j).\end{aligned}$$

We propose to minimize with respect to $\{\boldsymbol{r}_j\}_{j=1}^m$ the summation of predictive variances at all boundary points over all subdomains, that is,

$$\underset{\{\boldsymbol{r}_j\}_{j=1}^m}{\text{Minimize}} \quad \sum_{j=1}^m S_j(\boldsymbol{r}_j). \tag{20}$$

This is the realized version of (8) in our numerical solution procedure. Because this is a quadratic programming with respect to $\boldsymbol{r}_j$, we can easily see that the optimal boundary values $\boldsymbol{r}_{jk}$ at $\mathbf{x}_{jk}^b$ are given by (derivation in Appendix B)

$$\boldsymbol{r}_{jk} = (\boldsymbol{T}_{jk}^t\boldsymbol{T}_{jk})^{-1}\boldsymbol{T}_{jk}^t\left[\frac{\boldsymbol{h}_k^t\boldsymbol{y}_k}{\boldsymbol{h}_j^t\boldsymbol{y}_j + \boldsymbol{h}_k^t\boldsymbol{y}_k}\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_{jk}^b}^t\boldsymbol{h}_j + \frac{\boldsymbol{h}_j^t\boldsymbol{y}_j}{\boldsymbol{h}_j^t\boldsymbol{y}_j + \boldsymbol{h}_k^t\boldsymbol{y}_k}\boldsymbol{K}_{\mathbf{x}_k\mathbf{x}_{jk}^b}^t\boldsymbol{h}_k\right]. \tag{21}$$

Apparently, the minimizer of (20) is a weighted average of the mean predictions from two standard local GP predictors of neighboring subdomains.

In summary, we first solve the interface Equation (20) for all Γ_{kj} to obtain $\boldsymbol{r}_j$'s so that its choice makes local predictors continuous across internal boundaries. Given $\boldsymbol{r}_j$'s, we solve each local problem $\mathbf{LP(j)}'$, whose solution is given by (16) and yields the local mean predictors in (18) and the local predictive variance in (19). To simplify the expression of local predictive variance, we define a $\boldsymbol{u}_j$ as

$$\boldsymbol{u}_j := \boldsymbol{G}_j(\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}^t\boldsymbol{h}_j),$$

so that the predictive variance in (19) can be written as

$$\hat{\sigma}_j(x_*;\boldsymbol{r}_j) = k_{**} - \boldsymbol{k}_{\mathbf{x}_j*}^t(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{k}_{\mathbf{x}_j*} + \bar{\boldsymbol{k}}_{\mathbf{x}_j^b*}^t\boldsymbol{u}_j\boldsymbol{u}_j^t\bar{\boldsymbol{k}}_{\mathbf{x}_j^b*}/(\boldsymbol{h}_j^t\boldsymbol{y}_j),$$

A summary of the algorithm (labeled as DDM), including the steps of making the mean prediction and computing the predictive variance, is given in Algorithm 1.

Algorithm 1. Domain Decomposition Method (DDM).

```
1. Partition domain $\Omega$ into subdomains $\Omega_1,\ldots,\Omega_m$.
2. Precompute $\boldsymbol{H}_j, \boldsymbol{h}_j, \boldsymbol{G}_j$ and $c_j$ for each subdomain $\Omega_j$ using
     $\boldsymbol{H}_j \leftarrow (\sigma_j^2\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}, \boldsymbol{h}_j \leftarrow \boldsymbol{H}_j\boldsymbol{y}_j, c_j \leftarrow \boldsymbol{y}_j^t\boldsymbol{h}_j$,
     and $\boldsymbol{G}_j \leftarrow (\{\mathrm{diag}_{1/2}[(\boldsymbol{K}^t_{\mathbf{x}_j^b\mathbf{x}_j^b}\boldsymbol{K}_{\mathbf{x}_j^b\mathbf{x}_j^b})^{-1}]\boldsymbol{K}_{\mathbf{x}_j^b\mathbf{x}_j^b}\}$
          $\circ\{\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b}\mathrm{diag}[(\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j^b})^{-1}]\})^{-1}$.
3. Solve the interface equation for $j=1,\ldots,m$ and $k\in N(j)$:
     $\boldsymbol{r}_{jk} = (\boldsymbol{T}_{jk}^t\boldsymbol{T}_{jk})^{-1}\boldsymbol{T}_{jk}^t\left[\frac{c_k}{c_j+c_k}\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_{jk}^b}\boldsymbol{h}_j+\frac{c_j}{c_j+c_k}\boldsymbol{K}^t_{\mathbf{x}_k\mathbf{x}_{jk}^b}\boldsymbol{h}_k\right]$.
4. Compute the quantities in the local predictor. For each $\Omega_j$,
     i) $\boldsymbol{u}_j \leftarrow \boldsymbol{G}_j(\boldsymbol{T}_j^t\boldsymbol{r}_j-\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{h}_j)$.
5. Predict at location $x_*$. If $x_*$ is in $\Omega_j$,
     i) $\bar{\boldsymbol{k}}_{\mathbf{x}_j^b*} = [(\boldsymbol{k}^t_{\mathbf{x}_j^b*}\boldsymbol{k}_{\mathbf{x}_j^b*})^{-1/2}\boldsymbol{k}_{\mathbf{x}_j^b*}]\circ[\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}_j^b}\boldsymbol{k}_{\mathbf{x}_j*}(\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{k}_{\mathbf{x}_j*})^{-1}]$.
     ii) $\hat{p}_j(x_*;\boldsymbol{r}_j) \leftarrow \boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{h}_j+\bar{\boldsymbol{k}}^t_{\mathbf{x}_j^b*}\boldsymbol{u}_j$.
     iii) $\hat{\sigma}_j(x_*;\boldsymbol{r}_j) \leftarrow k_{**}-\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{H}_j\boldsymbol{k}_{\mathbf{x}_j*}+\bar{\boldsymbol{k}}^t_{\mathbf{x}_j^b*}\boldsymbol{u}_j\boldsymbol{u}_j^t\bar{\boldsymbol{k}}_{\mathbf{x}_j^b*}/c_j$.
```

Remark 1 Analysis of computational complexity. *Suppose that $n_j = B$ for $j = 1,\ldots,m$. The computational complexity for the precomputation step in Algorithm 1 is $O(mB^3)$, or equivalently, $O(NB^2)$. If we denote the number of sharing boundaries by w, the complexity for solving the interface equation is $O(wqB+q^3)$, where the inversion in $(\boldsymbol{T}_{jk}^t\boldsymbol{T}_{jk})^{-1}\boldsymbol{T}_{jk}^t$ is counted once because the $\boldsymbol{T}_{jk}$ matrix is essentially the same for all subdomains if we use the same polynomial evaluated at the same number of equally spaced locations. Since w is no more than dm for rectangle-shaped subdomains, the computation required to solve the interface equation is dominated by $O(dmqB)$, or equivalently, $O(dqN)$. Since computing $\boldsymbol{u}_j$'s requires only $O(mq^2)$ operations, the total complexity for performing Step 1 through 4 is $O(NB^2+dqN)$. We call this the 'training time complexity'. For small q and d, the complexity can be simplified to $O(NB^2+N)$, which is clearly dominated by NB^2. The existence of the dqN term also indicates that it does not help with computational saving to use too many control points on boundaries. On the other hand, we also observe empirically that using q greater than eight does not render significant gain in terms of reduction in boundary prediction mismatches (see Figure 2 and related discussion). Hence, we believe that q should, and could, be kept at a small value.*

The prediction step requires $O(B)$ computation for predictive mean and $O(B^2)$ for predictive variance after pre-computing $\boldsymbol{h}_j$ and $\boldsymbol{u}_j$. The complexities for training and prediction is summarized in Table 1 with a comparison to several other methods including FIC, PIC, BCM, and LPR. Note that the second row in Table 1 is the computational complexity for a fully parallelized domain decomposition approach (denoted by P-DDM), which will be explained later in Section 6.7, and BGP in the sixth row refers to the Bagged Gaussian Process, to be explained in Section 6.

Remark 2 One dimensional case. *The derivation in this section simplifies significantly in the one dimensional case. In fact, all results hold with the simplification $p = q = 1$. When $d = 1$, Γ_{jk} has only one point and there is no need to define a polynomial boundary value function. Denote by r_{jk}*

the prediction at the boundary Γ_{jk}*, the local prediction problem* $\boldsymbol{LP(j)}'$ *is simply*

$$\begin{aligned} \underset{A\in\mathbb{R}^{n_j\times n_j}}{\text{Minimize}} \quad & \boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})\boldsymbol{A}_j\boldsymbol{k}_{\mathbf{x}_j*}-2\boldsymbol{k}^t_{\mathbf{x}_j*}\boldsymbol{A}^t_j\boldsymbol{k}_{\mathbf{x}_j*} \\ \text{s.t.} \quad & \boldsymbol{K}^t_{\mathbf{x}_{jk}}\boldsymbol{A}^t_j\boldsymbol{y}_j = r_{jk} \quad \textit{for } k\in N(j). \end{aligned}$$

The local mean predictor is straightforwardly obtained from expression (18) *by replacing* $\boldsymbol{T}_{jk}$ *with 1.*

5. Hyperparameter Learning

By far, our discussions were made when using fixed hyperparameters. We discuss in this section how to estimate hyperparameters from data. Inheriting the advantage of local kriging, DDM can choose different hyperparameters for each subdomain. We refer to such subdomain-specific hyperparameters as “local hyperparameters.” Since varying hyperparameters means varying covariance functions across subdomains, nonstationary variation in the data can be captured by using local hyperparameters. On the other hand, if one set of hyperparameters is used for the whole domain, we refer to these hyperparameters as “global hyperparameters.” Using global hyperparameters is desirable when the data are spatially stationary.

Maximizing a marginal likelihood is a popular approach for hyperparameter learning in likelihood approximation based methods (Seeger et al., 2003; Snelson and Ghahramani, 2006, 2007). Obtaining the optimal hyperparameter values is generally difficult since the likelihood maximization is usually a non-linear and non-convex optimization problem. The method has nonetheless successfully provided reasonable choices for hyperparameters. We propose to learn local hyperparameters by maximizing the local marginal likelihood functions. Specifically, the local hyperparameters, denoted by θ_j associated with each Ω_j, are selected such that they minimize the negative log marginal likelihood:

$$ML_j(\theta_j) := -\log p(\boldsymbol{y}_j;\theta_j) = \frac{n_j}{2}\log(2\pi)+\frac{1}{2}\log|\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j}|+\frac{1}{2}\boldsymbol{y}^t_j(\sigma^2_j\boldsymbol{I}+\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{y}_j, \quad (22)$$

where $\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j}$ depends on θ_j. Note that (22) is the marginal likelihood of the standard local kriging model. One might want to replace $\sigma^2_j\boldsymbol{I}+\boldsymbol{K}^{-1}_{\mathbf{x}_j\mathbf{x}_j}$ in (22) by the optimal $\boldsymbol{A}_j$ that solves (11). However, doing so needs to solve for $\boldsymbol{A}_j$, $\boldsymbol{r}_{jk}$ and θ_j iteratively, which is computationally more costly. Our simple strategy above disentangles the hyperparameter learning and the prediction problem, and works well empirically (see Section 6).

When we want to have the global hyperparameters for the whole domain, we choose θ such that it minimizes

$$ML(\theta) = \sum_{j=1}^{m} ML_j(\theta), \quad (23)$$

where the summation of the negative log local marginal likelihoods is over all subdomains. The above treatment assumes that the data from each subdomain are mutually independent. This is certainly an approximation to solving the otherwise computationally expensive global marginal likelihood.

In the likelihood approximation based methods like FIC, the time complexity to evaluate a marginal likelihood is the same as their training computation, that is, $O(NM^2)$. However, numerically optimizing the marginal likelihood runs such evaluation a number of iterations, usually, 50–200 times. For this reason, the total training time (counting the hyperparameter learning as well) is

Methods	Hyperparameters Learning	Training	Prediction	
			Mean	Variance
DDM	$O(LNB^2)$	$O(NB^2)$	$O(B)$	$O(B^2)$
P-DDM	$O(LB^3)$	$O(B^3)$	$O(B)$	$O(B^2)$
FIC	$O(LNM^2)$	$O(NM^2)$	$O(M)$	$O(M^2)$
PIC	$O(LN(M+B)^2)$	$O(N(M+B)^2)$	$O(M+B)$	$O((M+B)^2)$
BCM	$O(LNM^2)$	$O(NM^2+N_q^3)$	$O(NM)$	$O(NM)$
BGP	$O(LKM^3)$	$O(KM^3)$	$O(KM^2)$	$O(KM^2)$
LPR	$O(R(LM^3+N))$		$O(KM^3+KN)$	$O(KM^3+KN)$

Table 1: Comparison of computational complexities: we suppose that L iterations are required for learning hyperparameters; for DDM, the number of control points q on a boundary is kept to be a small constant as discussed in Remark 1; for BCM, N_q is the number of testing points; for BGP, K is the number of bootstrap samples, M is the size of each bootstrap sample; for LPR, R is the number of the subsets of training points used for estimating local hyperparameters and K is the number of local experts of size M.

much slower than expected. The computational complexity of DDM is similar to FIC, as shown in Table 1. One way that can significantly improve the computation is through parallelization, which is easier to conduct for DDM because the m local predictions can be performed simultaneously. If a full parallelization can be done, the computational complexity for one iteration using DDM reduces to $O(B^3)$, where $n_j = B$ is assumed for all j's. For more comparison results, see Table 1.

6. Experimental Results

In this section, we present some experimental results for evaluating the performance of DDM. First, we show how DDM works as the tuning parameters of DDM (p, q and m) change, and provide some guidance on setting the tuning parameters when applying DDM. Then, we compare DDM with several competing methods in terms of computation time and prediction accuracy. We also evaluate how well DDM can solve the problem of prediction mismatch on boundaries.

The competing methods are local GP (i.e., local kriging), FIC (Snelson and Ghahramani, 2006), PIC (Snelson and Ghahramani, 2006), BCM (Tresp, 2000), and LPR (Urtasun and Darrell, 2008). We also include in our comparative study the Bagged Gaussian Processes (BGP, Chen and Ren, 2009) as suggested by a referee, because it is another way to provide continuous prediction surfaces by smoothly combining independent GPs. BGP was originally developed for improving the robustness of GP regression, not for the purpose of faster computation. The prediction by BGP is an average of the predictions obtained from multiple bootstrap resamples, each of which has the same size as the training data. Hence, its computational complexity is no better than the full GP regression. But faster computation can be achieved by reducing the bootstrap sample size to a small number $M \ll N$, a strategy used in our comparison.

FIC and PIC does not allow the use of local hyperparameters for reflecting local variations of data, so we used global hyperparameters for both FIC and PIC, and for DDM as well, for the sake of fairness. The remaining methods are designed to allow local hyperparameters, so DDM uses local hyperparameters for comparison with those.

We did not compare DDM with the the mixture of GP experts such as MGP (Rasmussen and Ghahramani, 2002) and TGP (Gramacy and Lee, 2008), because their computation times are far worse than the other compared methods especially for large data sets, due to the use of computationally slow sampling methods. For example, according to our simple experiment, it took more than two hours for TGP to train its predictor for a data set with 1,000 training points and took more than three days (79 hours) for a larger data set with 2,000 training points, while other competing methods took only a few seconds. We did not directly implement and test MGP, but according to Gramacy and Lee (2008, page 1126), MGP's computation efficiency is no better than TGP. In general, the sampling based approaches are not competitive for handling large-scale data sets and thus are inappropriate for comparison with DDM, even though they may be useful on small to medium-sized data sets in high dimension.

6.1 Data Sets and Evaluation Criteria

We considered four data sets: two synthetic data sets (one in 1-d and the other in 2-d) and two real spatial data sets both in 2-d. The synthetic data set in 1-d is packed together with the FIC implementation by Snelson and Ghahramani (2006). It consists of 200 training points and 301 test points. We use this synthetic data set to illustrate that PIC still encounters the prediction mismatch problem at boundaries, while the proposed DDM does solve the problem for 1-d data. The second synthetic data set in 2-d, `synthetic-2d`, was generated from a stationary GP with an isotropic squared exponential function using the R package `RandomFields`, where nugget = 4, scale=4, and variance=8 are set as parameters for the covariance function. It consists of 63,001 sample points.

The first real data set, `TCO`, contains data collected by NIMBUS-7/TOMS satellite to measure the total column of ozone over the globe on Oct 1 1988. This set consists of 48,331 measurements. The second real data set, `MOD08-CL`, was collected by the Moderate Resolution Imaging Spectroradiometer (MODIS) on NASA's Terra satellite that measures the average of cloud fractions over the globe from January to September in 2009. It has 64,800 measurements. Spatial non-stationarity presents in both real data sets.

Using the second synthetic data set and the two real spatial data sets, we compare the computation time and prediction accuracy among the competing methods. We randomly split each data set into a training set containing 90% of the total observations and a test set containing the remaining 10% of the observations. To compare the computational efficiency of methods, we measure two computation times, the training time (including the time for hyperparameter learning) and the prediction (or test) time. For comparison of accuracy, we use three measures on the set of the test data, denoted as $\{(x_t, y_t); t = 1, \ldots, T\}$, where T is the total data amount in the test set. The first measure is the mean squared error (MSE)

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \mu_t)^2,$$

which measures the accuracy of the mean prediction μ_t at location x_t. The second one is the negative log predictive density (NLPD)

$$\text{NLPD} = \frac{1}{T}\sum_{t=1}^{T}\left[\frac{(y_t-\mu_t)^2}{2\sigma_t^2} + \frac{1}{2}\log(2\pi\sigma_t^2)\right],$$

which considers the accuracy of the predictive variance σ_t as well as the mean prediction μ_t. These two criteria were used broadly in the GP regression literature. The last measure, the mean squared mismatch (MSM), measures the mismatches of mean prediction on boundaries. Given a set of test points, $\{x_e; e = 1, \ldots, E\}$, located on the boundary between subdomains Ω_i and Ω_j, the MSM is defined as

$$\text{MSM} = \frac{1}{E}\sum_{e=1}^{E}(\mu_e^{(i)} - \mu_e^{(j)})^2,$$

where $\mu_e^{(i)}$ and $\mu_e^{(j)}$ are mean predictions from Ω_i and Ω_j, respectively. A smaller value of MSE, NLPD or MSM indicates a better performance.

Our implementation of DDM was mostly done in MATLAB. When applying DDM to the 2-d spatial data, one issue is how to partition the whole domain into subdomains, also known as *meshing* in the finite element analysis literature (Ern and Guermond, 2004). A simple idea is just to use a uniform mesh, where each subdomain has roughly the same size. However simple, this idea works surprisingly well in many applications, including our three data sets in 2-d. Thus, we used a uniform mesh with each subdomain shaped rectangularly in our implementation.

For FIC, we used the MATLAB implementation by Snelson and Ghahramani (2006), while for BCM, the implementation by Schwaighofer et al. (2003) was used. Since the implementations of the other methods are not available, we wrote our own codes for PIC, LPR and BGP. Throughout the numerical analysis, we used the anisotropic version of a squared exponential covariance function. All numerical studies were performed on a computer with two 3.16 GHz quadcore CPUs.

6.2 Mismatch of Predictions on Boundaries

DDM puts continuity constraints on local GP regressors so that predictions from neighboring local GP regressors are the same on boundaries for 1-d data and are well controlled for 2-d data. In this section, we show empirically, by using the synthetic 1-d data set and the 2-d (real) `TCO` data set, the effectiveness of having the continuity constraints.

For the synthetic data set, we split the whole domain, $[-1, 7]$, into four subdomains of equal size. The same subdomains are used for local GP, PIC and DDM. PIC is also affected by the number and locations of inducing inputs. To see how the mismatch of prediction is affected by the number of inducing inputs, we considered two choices, five and ten, as the number of inducing inputs for PIC. The locations of inducing inputs along with the hyperparameters are chosen by optimizing the marginal likelihood. For DDM, the local hyperparameters are obtained for each subdomain using the method described in Section 5.

Figure 1 shows for the synthetic data the predictive distributions of the full GP regression, local GP, PIC with $M = 5$, PIC with $M = 10$, and DDM. In the figure, red lines are the predictive means of the predictive distributions. The mean of local GP and the mean of PIC with $M = 5$ have substantial discontinuities at $x = 1.5$ and $x = 4.5$, which correspond to the boundary points of subdomains. As M increases to 10, the discontinuities decrease remarkably but are still visible. In general, the mismatch in prediction on boundaries is partially resolved in PIC by increasing the number of inducing inputs at the expense of longer computing time. By contrast, the mean prediction of DDM is continuous, and close to that of the full GP regression.

Unlike in the 1-d case, DDM cannot perfectly solve the mismatch problem for 2-d data. Our algorithm chooses to enforce continuity at a finite number of control points. A natural question is whether continuity uniformly improves as the number of control points (q) increases. This question

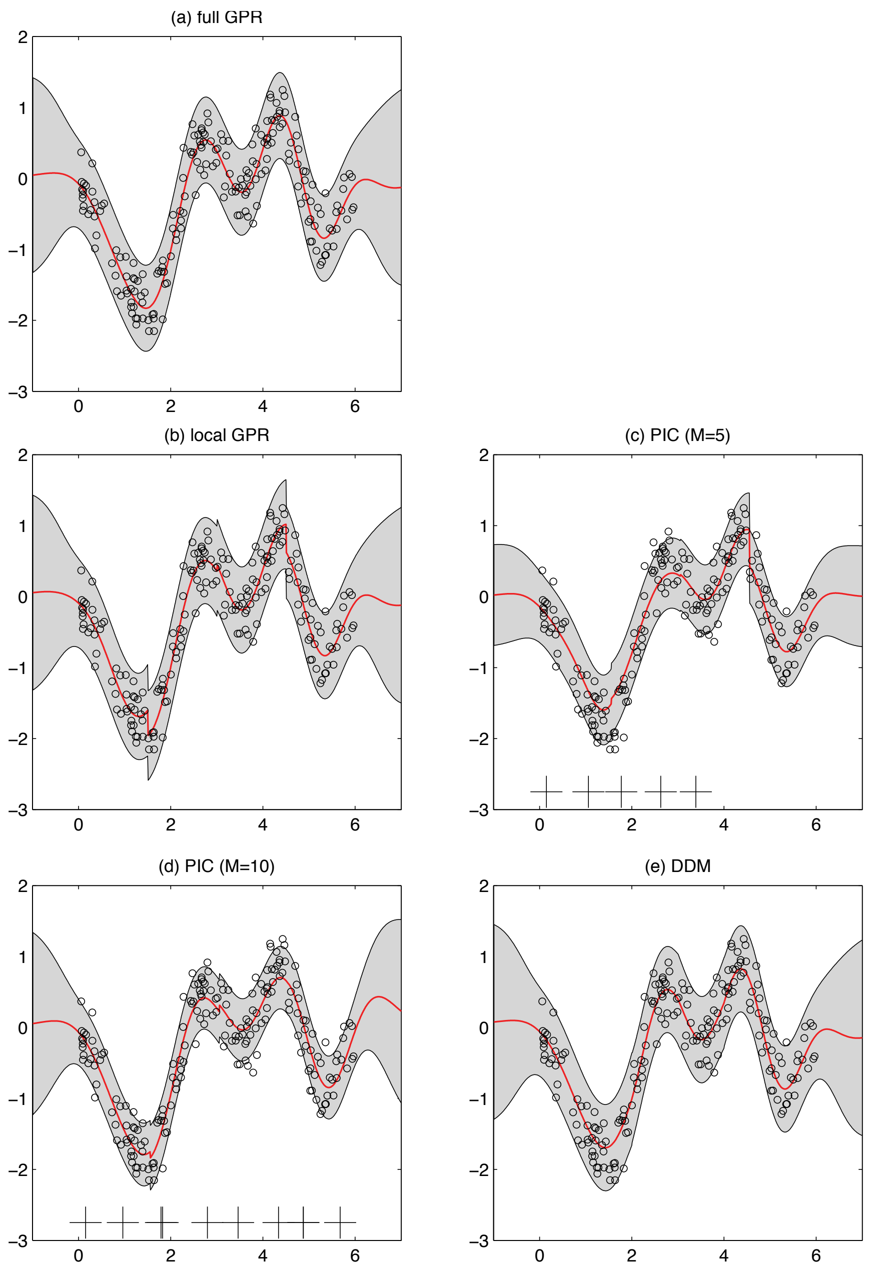


Figure 1: Comparison of predictive distribution in the synthetic data set: circles represent training points; the red lines are predictive means and the gray bands represent deviation from the predictive means by ± 1.5 times of predictive standard deviations; black crosses on the bottom of plots for PIC show the locations of inducing inputs.

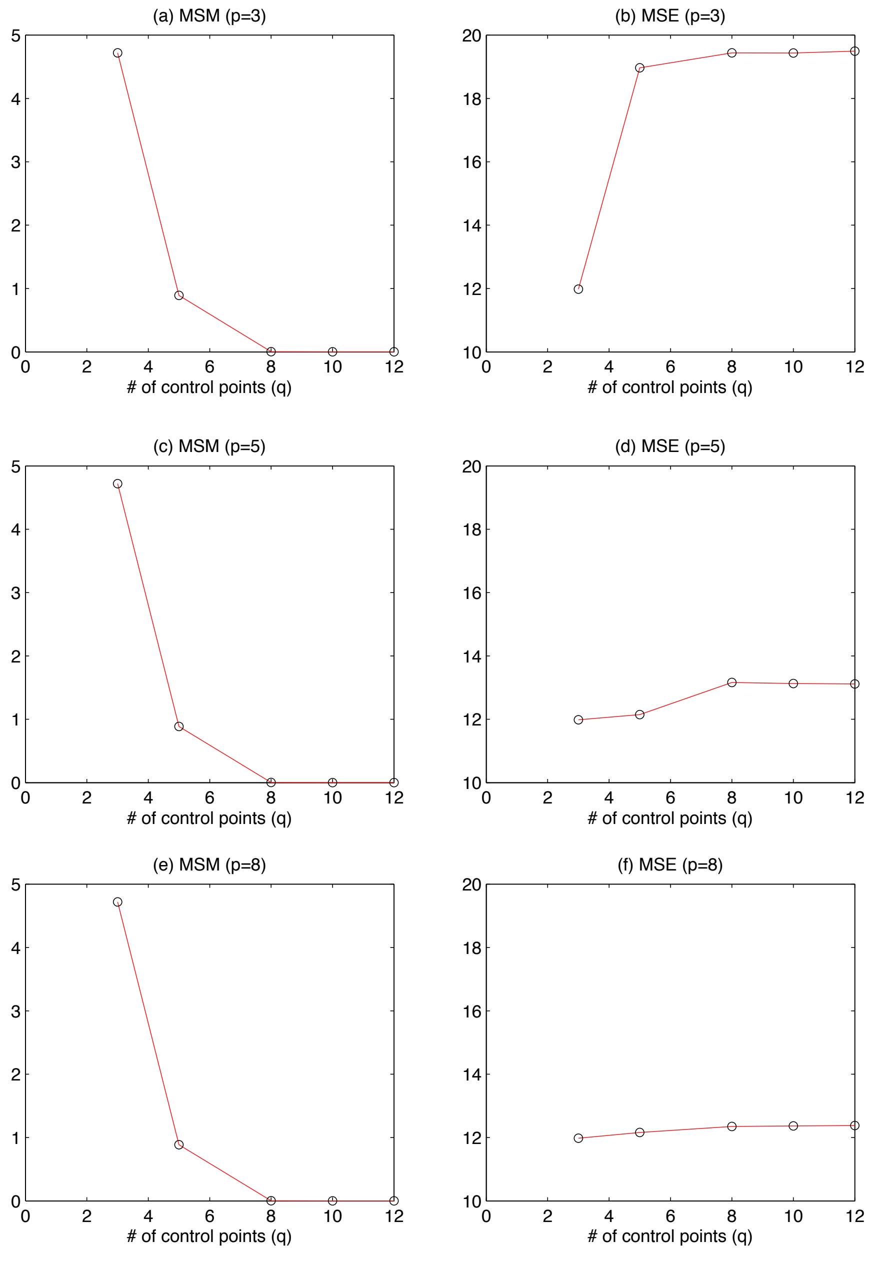


Figure 2: Left column: MSM versus the degrees of freedom p and the number of control points q; Right column: MSE versus p and q.

is related to the stability of the algorithm. Another interesting question is whether the degrees of freedom (p) affects the continuity or other behaviors of DDM. To answer these questions, we traced MSE and MSM with the change of p and q for a fixed regular grid.

We observe from Figure 2 that for the TCO data set, the magnitude of prediction mismatch, measured by MSM, decreases as we increase the number of control points. We also observe that there is no need to use too many control points. For the 2-d data sets at hand, using more than eight control points does not help much in decreasing the MSM further; and the MSM is close to zero with eight (or more) control points. On the other hand, if the degrees of freedom (p) is small but the number of control points is larger, the MSE could increase remarkably (see Figure 2-(b)). This is not surprising, because the degrees of freedom determines the complexity of a boundary function, and if we pursue better match with too simple boundary function, we would distort local predictors a lot, which will in the end hurt the accuracy of the local predictors. If p is large enough to represent the complexity of boundary functions, the MSE stays roughly constant regardless of q (see Figure 2-(d) and 2-(f)). To save space, we do not present here the results for another real data set, MOD08-CL, because they are consistent with those for TCO. Our general recommendation is to use a reasonably large p and let $q = p$.

6.3 Choice of Mesh Size for DDM

An important tuning parameter in DDM is the mesh size. In this section, we provide a guideline for an appropriate choice of the mesh size through some designed experiments. The mesh size is defined by the number of training data points in a subdomain, previously denoted by B. We empirically measure, using the synthetic-2d, TCO and MOD08-CL data sets, how MSE and training/testing times change for different B's. In order to characterize the goodness of B, we introduce in the following a "marginal MSE loss" with respect to the total computation time $Time$, that is, training time + test time, measured in seconds. Given a set of mesh sizes $\mathcal{B} = \{B_1, B_2, ..., B_r\}$,

$$\text{marginal}(B; B^*) := \max\left\{0, \frac{MSE(B) - MSE(B^*)}{1 + Time(B^*) - Time(B)}\right\} \text{ for } B \in \mathcal{B},$$

where $B^* = \max\{B \in \mathcal{B}\}$. The denominator implies how much time saving is obtained for a reduced B, while the numerator implies how much MSE we lost with the time saving. But marginal$(B; B^*)$ alone is not a good enough measure because marginal$(B; B^*)$ is always zero at $B = B^*$. So, we balanced the loss by adding the change in MSE and computation relative to the smallest mesh size in $\mathcal{B}$, namely

$$\text{marginal MSE loss} := \text{marginal}(B; B^*) + \text{marginal}(B; B^\circ),$$

where $B^\circ = \min\{B \in \mathcal{B}\}$. We can interpret the marginal MSE loss as how much MSE is sacrificed for a unit time saving, so smaller values are better.

Figure 3 shows the marginal MSE loss for the three data sets. For all data sets, a B between 200 and 600 gives smaller marginal MSE loss. Based on this empirical evidence, we recommend to choose the mesh size so that the number of training data points in a subdomain ranges from 200 to 600. If the number is too large, DDM will spend too much time for small reduction of MSE. Conversely, if the number is too small, MSE will deteriorate significantly. The latter might be because DDM has too fewer training data points to learn local hyperparameters.

6.4 DDM Versus Local GP

We compared DDM with local GP for different mesh sizes and in terms of overall prediction accuracy and mismatch on boundaries. We considered two versions of DDM, one using global hyperpa-

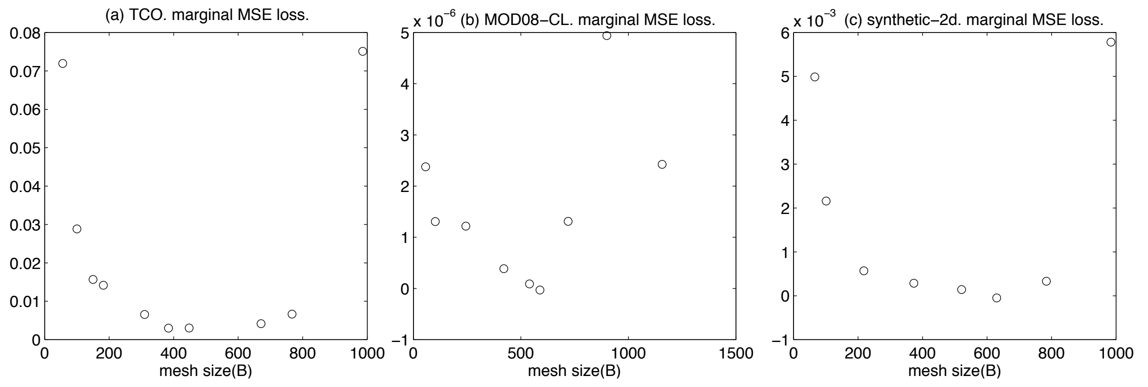


Figure 3: Marginal MSE loss versus mesh size(B). For the three data sets, the marginal loss is small when B is in between 200 and 600.

rameters (G-DDM) and the other using local hyperparameters (L-DDM). For local GP, we always used local hyperparameters.

Figure 4 shows the three performance measures as a function of the number of subdomains for G-DDM, L-DDM and local GP, using the `TCO` data and the synthetic-2d data, respectively. DDM adds more computation to local GP for imposing the continuity on boundaries, but the increased computation is very small relative to the original computation of local GP. Hence, the comparison of DDM with local GP as a function of the number of subdomains is almost equivalent to the comparison in terms of the total time (i.e., training plus test time).

In Figure 4, local GP has bigger MSE and NLPD than the two versions of DDM for both data sets. The better performance of DDM can be contributed to the better prediction accuracy around boundaries of subdomains. The comparison results for two versions of DDM are as expected: In terms of MSE and NLPD, L-DDM is better than G-DDM for the `TCO` data set, which can be explained by nonstationarity of the data. On the other hand, for the synthetic-2d data set, G-DDM is better, which is not surprising since the synthetic-2d data set is generated from a stationary GP so one would expect that global hyperparameters work well.

The left panels of Figure 5 show the comparison results for the actual `MOD08-CL` data set. In terms of MSE and NLPD, L-DDM is appreciably better than local GP when the number of subdomains is small, but the two methods perform comparably when the number of subdomains is large. This message is somewhat different from what we observed for `TCO` data set. One explanation is that `TCO` data set has several big empty spots with no observation over the subregion, but `MOD08-CL` data set does not have such “holes”. Because of the continuity constrains, we believe DDM is able to borrow information from neighboring subdomains, and consequently, to provide better spatial predictions. To verify this, we randomly chose twenty locations within the spatial domain of `MOD08-CL` data set and artificially removed small neighborhoods of each randomly chosen location from the `MOD08-CL` data set; doing so resulted in a new data set called “`MOD08-CL` with holes”. The results of applying three methods on this new data set are shown on the right panels of Figure 5. L-DDM is clearly superior over local GP across different choices of the number of subdomains.

This comparison reveals that when there is nonstationary in data, using local parameters (local GP and L-DDM versus G-DDM) will help adapt to the non-stationary features, and thus, improve the prediction accuracy. More importantly, the improvement in prediction can be further enhanced by a proper effort to smooth out the boundary mismatches in localized methods (L-DDM versus local GP). In all cases, the MSM associated with DDM method is very small.

6.5 G-DDM Versus FIC and PIC

We compared prediction accuracy of G-DDM with FIC and PIC. We only considered global hyperparameters for DDM because FIC and PIC cannot incorporate local hyperparameters. Since each of the compared methods has different tuning parameters, it is hard to compare these methods using prediction accuracy measures (MSE and NLPD) for a fixed set of tunning parameters. Instead, we considered MSE and NLPD as a function of the total computation time required. To obtain the prediction accuracy measures for different computation times, we tried several different settings of experiments and presented the best accuracies of each method for given computation times: for DDM, we varied the number of equally sized subdomains (m) and the number of control points q while keeping the degrees of freedom p the same as q; we tested two versions of PIC having different domain decomposition schemes: k-means clustering (denoted by kPIC) and regular grids meshing (denoted by rPIC), and for each version, we varied the total number of subdomains (m) and the number of inducing inputs (M); for FIC, we varied the number of inducing inputs (M). We see that each of the compared methods has one major tuning parameter mainly affecting their training and test times; it is m for DDM, or M for FIC and PIC. In order to obtain one point in Figure 6, we first fixed the major tuning parameter for each method, and then changed the remaining parameters to get the best accuracy for a given computation time.

In this empirical study, for G-DDM, a set of the global hyperparameters was learned by minimizing (23). In FIC, the global hyperparameters, together with the locations of the inducing inputs, were determined by maximizing its marginal likelihood function. For PIC, we tested several options: learning the hyperparameters and inducing inputs by maximizing the PIC approximated marginal likelihood; learning the hyperparameters by maximizing the PIC approximated marginal likelihood, whereas learning the inducing inputs by the FIC approximated likelihood; or learning the hyperparameters by the FIC approximated marginal likelihood, whereas learning the inducing inputs by the PIC approximated marginal likelihood. Theoretically, the first option should be the best choice. However, as discussed in Section 5, due to the non-linear and non-convex nature of the likelihood function, an optimization algorithm may converge to a local optimum and thus yields a suboptimal solution. Consequently, it is not clear which option's local optimum produces the best performance. According to our empirical studies, for the `TCO` data set, the first option gave the best result, while for the `MOD08-CL` data set, the third option was the best. We present the results based on the empirically best result.

Figure 6 shows MSE and NLPD versus the total computation time. G-DDM exhibits superior performance for the two real data sets. We observe that FIC and PIC need a large number of inducing inputs, at the cost of much longer computation time, in order to lower its MSE or NLPD to a level comparable to G-DDM. Depending on specific context, the difference in computation time could be substantial. For the instance of `TCO` data set, G-DDM using 156 subdomains produced MSE = 17.7 and NLPD = 2.94 with training time = 47 seconds. FIC could not obtain a similar result even with $M = 500$ and computation time = 484 seconds, and rPIC could obtain MSE = 25.8 and NLPD = 3.06 after spending 444 seconds and using 483 subdomains and M = 500. For the synthetic-2d

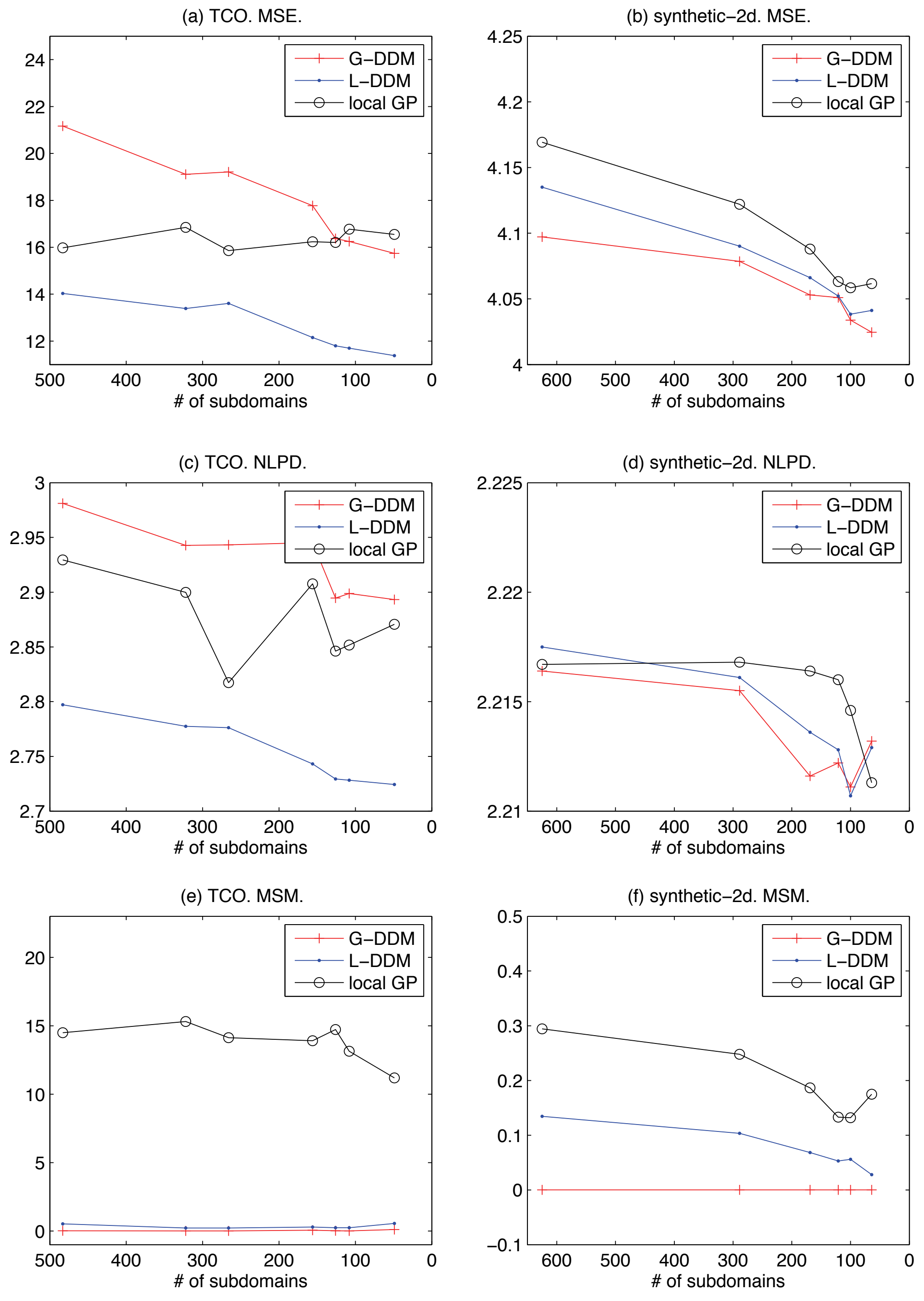


Figure 4: Prediction accuracy of DDM and local GP for different mesh sizes. For `TCO`, p ranged from five to eight for G-DDM and L-DDM. For synthetic-2d data set, p ranged from four to eight.

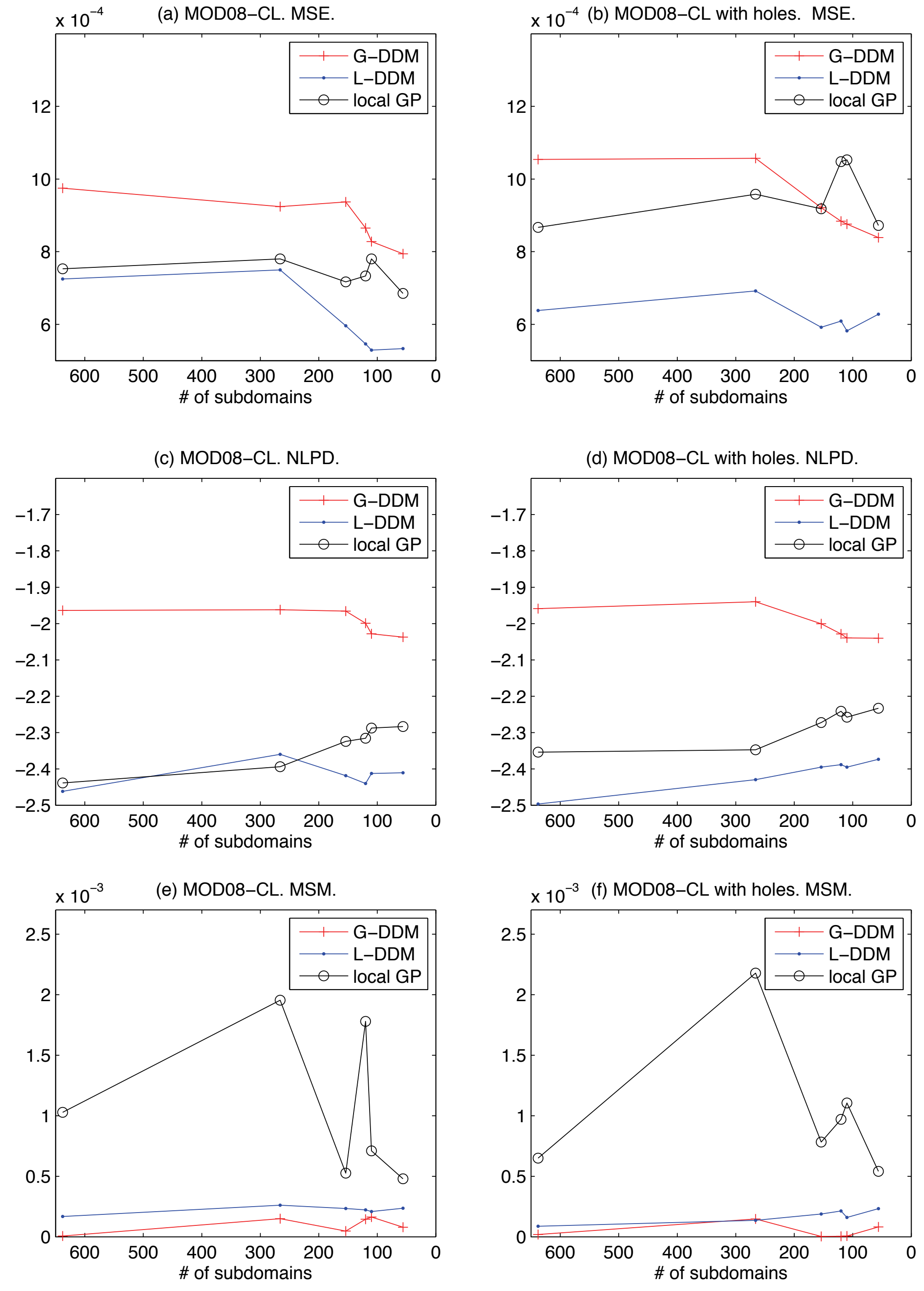


Figure 5: Prediction accuracy of DDM and local GP for the `MOD08-CL` data set. The left panel uses the original `MOD08-CL` data, while the right panel uses the `MOD08-CL` data with observations removed at a number of locations. For the two data sets, p and q ranged from four to eight for G-DDM and L-DDM.

data set from a stationary GP, G-DDM does not have advantage over FIC and PIC but still performs reasonably well.

We here used two versions of PIC: kPIC and rPIC, and the difference between them is how the subdomains are created. Since rPIC uses a regular grid meshing to decompose the domain, the general perception is that rPIC might not perform as well as kPIC, due to this domain-decomposition rigidity. It is interesting, however, to see that this perception is not supported by our empirical studies of using large-size spatial data sets. In Figure 6, kPIC exhibits no appreciable difference with rPIC in terms of MSE and NLPD. Please note that we actually did not count the time for conducting the domain decomposition when we recorded the training time. If we consider the computation complexities of the k-means clustering versus the regular grid meshing, then kPIC would be less attractive. This is because the time complexity for performing the k-means clustering is $O(IkdN)$, much more expensive than that for regular grid meshing, which only requires $O(dN)$ computation, where I is the number of iterations required for the convergence of the clustering algorithm, k is the size of neighborhoods, d is the dimension of data, and N is the number of data points.

Regarding the mismatch of prediction on boundaries as measured by MSM, G-DDM is multifold better than that of rPIC; see Figure 7. This is not surprising, since DDM explicitly controls the mismatch of prediction on boundaries. For kPIC, we could not measure MSM because it is difficult to define boundaries when we use the k-means clustering for the purpose of domain decomposition. FIC does not have the mismatch problem since it does not use subdomains for prediction.

6.6 L-DDM Versus Local Methods

We compared prediction accuracy of L-DDM with three localized regression methods, BCM, BGP, and LPR, all of which partition the original data space for fast computation. BGP uses different hyperparameters for each bootstrap sample, but strictly speaking, these hyperparameters cannot be called "local hyperparameters" since each bootstrap sample is from the whole domain, not a local region. However, BGP can be converted to have local hyperparameters by making bootstrap samples to come from local regions in the same way as BCM, that is, via k-means clustering. We call the "local version" of BGP as L-BGP, and we present the experimental results of both BGP and L-BGP (this L-BGP is in fact suggested by one referee). We present the results in the same way as in the previous section by plotting computation times versus prediction accuracy measures. Since the computational complexity comparison here is significantly different for training and testing (or prediction), the results for training time and test time are presented separately.

To obtain the prediction accuracy measures for different computation times, we tried several different settings of experiments and presented the best accuracies of each method for given computation times: for DDM, we varied the number of equally sized subdomains and the number of control points q while keeping the degrees of freedom p the same as q; for BGP, the number of bootstrap samples (K) ranged from 5 to 30 and the number of data points in each model (M) ranged from 300 to 900; for L-BGP, the number of local regions (K) ranged from for 9 to 64 and the number of data points in each model (M) ranged from 150 to 1500; for LPR, the number of local experts (K) ranged from 5 to 20 and the number of data points used for each expert (M) ranged from 50 to 200 while the number of locations chosen for local hyperparameter learning (R) ranged from 500 to 1500; for BCM, the number of local estimators (M) was varied from 100 to 600. Similar to what we did in Section 6.5, we still use one or two major parameters to determine the computation time first, and then use the remaining parameters to get the best accuracy for each method. The major

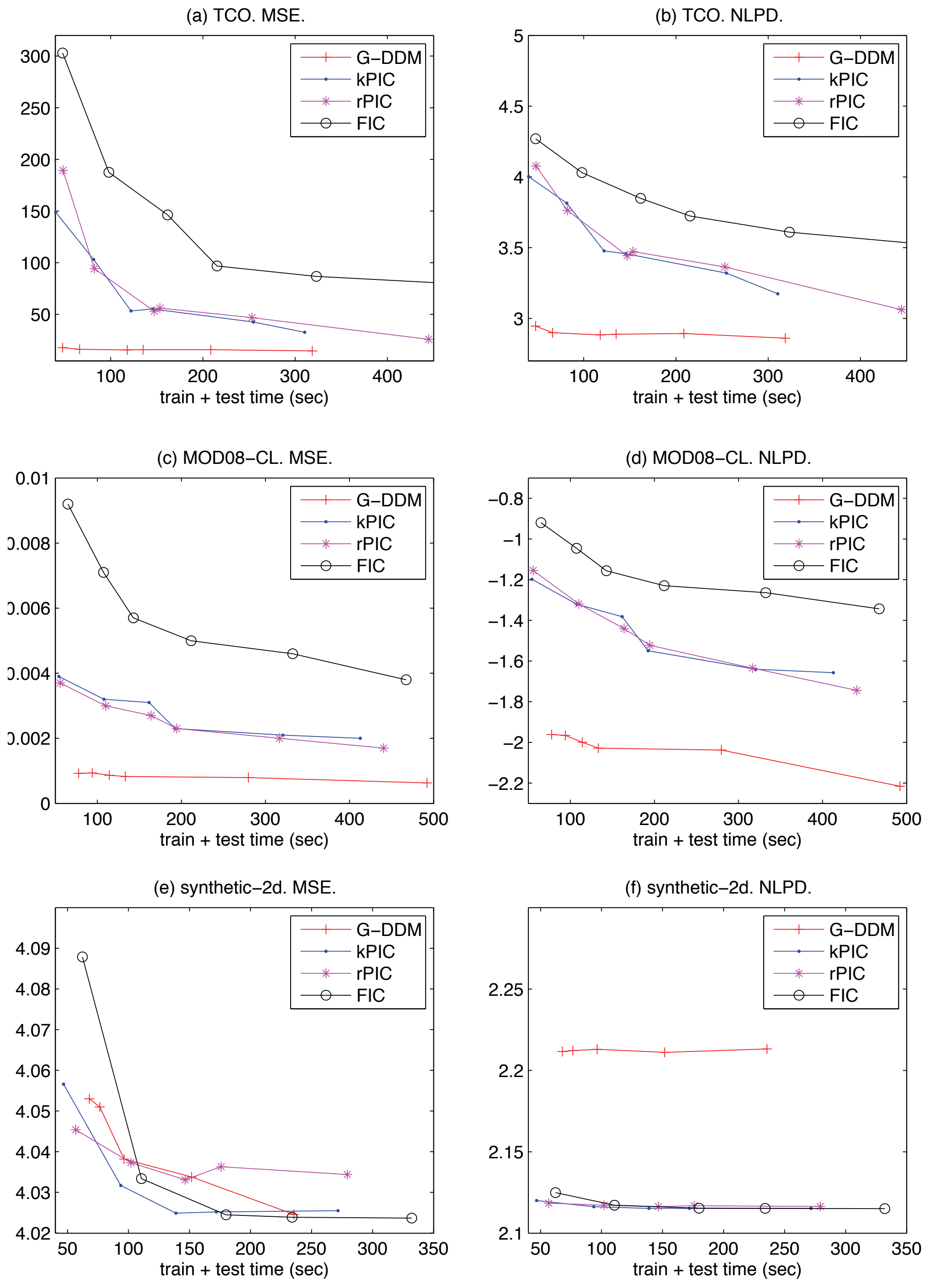


Figure 6: Prediction accuracy versus total computation time. For all three data sets, G-DDM uses $m \in \{36, 56, 110, 156, 266, 638\}$; FIC uses $M \in \{50, 100, 150, 200, 300, 400\}$; kPIC and rPIC use $M \in \{50, 100, 150, 200, 300, 400\}$.

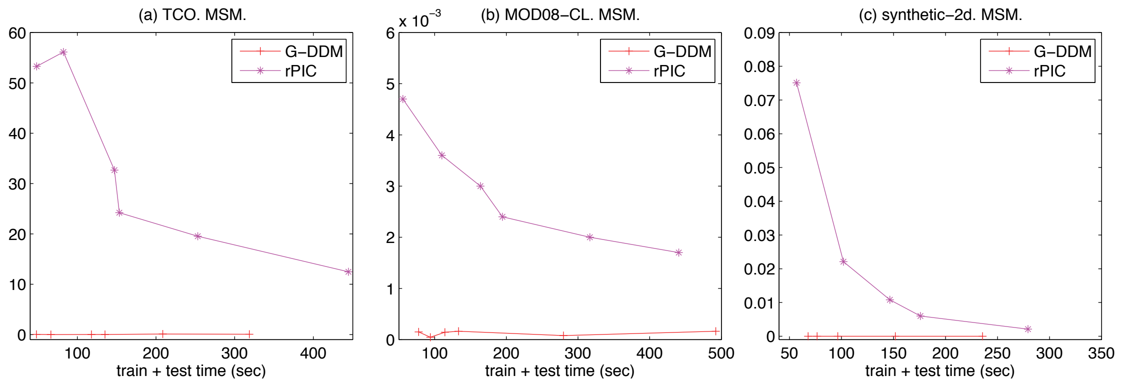


Figure 7: MSM versus total computation time. For three data sets, G-DDM uses $m \in \{36, 56, 110, 156, 266, 638\}$; rPIC uses $M \in \{50, 100, 150, 200, 300, 400\}$.

time-determining parameters are: m for DDM; K and M for BGP; K for L-BGP; T and R for LPR; M for BCM.

The local hyperparameters for the methods in comparison are all learned by minimizing (22). However, for BCM, when we tried to use local hyperparameters for the TCO data set, the implementation by Schwaighofer et al. (2003) always returned "NA" (not available) so we could not obtain valid results with local hyperparameters. Therefore, we applied global hyperparameters to BCM only for the TCO data set. The global hyperparameters were learned by minimizing (23), which is equivalent to the implementation of BCM by Schwaighofer et al. (2003). When we ran our implementation of LPR, we found that the results are sensitive to the setting of its tuning parameters. The reported results for LPR are based on the set of tuning parameters that gives the best MSE, chosen from more than thirty different settings.

Figure 8 traces MSEs and NLPDs as a function of training time for the three data sets. For TCO, BCM and L-DDM have comparably good accuracy (measured using MSE) with similar training costs, but the NLPD of L-DDM is much smaller than that of BCM, implying that the goodness of fit of L-DDM is better. The other methods do not perform as accurately as L-DDM with even much greater training cost. For all of the three data sets, BCM, BGP, L-BGP and LPR have higher, and sometimes much higher, NLPD than L-DDM. By the definition of NLPD, both a big MSE and a small predictive variance will lead to a high NLPD. Thus, we can infer that, for the TCO data set, the differences of NLPD between L-DDM and BCM are mainly caused by too small predictive variances of BCM (i.e., BCM underestimates the predictive variances considerably), since the MSEs produced by the two methods are very close. For other data sets, the differences in NLPD come from both the differences in MSE and differences in predictive variance. For the stationary synthetic data set, BCM has high MSE and NLPD, suggesting that BCM might not be very competitive for stationary data sets. Overall, L-DDM outperforms all other methods for both non-stationary and stationary data sets.

Figure 9 shows MSEs and NLPDs as testing times change. One observes that the testing times are significantly different across methods. In particular, the computation time needed to predict at a new location for BCM and LPR is far longer than that for L-DDM or BGP. This is also supported by the computational complexity analysis presented in Table 1. One also observes that the curves of

L-DDM always locate at the lower-left parts of the plots, implying that L-DDM spent much shorter prediction time but obtained much better prediction accuracy. Note that the x-axis and y-axis of the plots are log-scaled so the difference in computation times is much bigger than what it looks like in the plots. For examples, L-DDM spent less than three seconds for all the data sets for making a prediction, while BCM's prediction time ranged from 100 to 1,000 seconds, and LPR spent from 189 to 650 seconds. BCM and LPR do not look competitive when the number of locations to predict is large, a situation frequently encountered in real spatial prediction problems.

6.7 Benefit of Parallel Processing

As mentioned earlier, one advantage of DDM is that its computation can be parallelized easily. This advantage comes from its domain decomposition formulation. As soon as a solution of interface Equation (8) is available, (11) can be solved simultaneously for individual subdomains. Once fully parallelized, the computational complexity of DDM reduces to $O(B^3)$ for training, and that for hyperparameter learning is reduced to $O(LB^3)$. Since the computation of hyperparameter learning usually accounts for the biggest portion of the entire training time, parallelization could provide a remarkable computational saving. See the second row of Table 1 for a summary of the computational complexity for the parallel version of DDM (P-DDM).

While a full parallelization of DDM needs the support from sophisticated software and hardware and is thus not yet available, we implemented a rudimentary version of P-DDM by using the MATLAB Parallel Processing Toolbox on a computer with two quadcore CPUs. In doing so, we replaced the regular `for`-loop with its parallel version `parfor`-loop and examined how much the training time can be reduced by this simple action.

Denote the training time from the sequential DDM as ST, the training time from P-DDM as PT, and define the "speed-up ratio" as ST/PT. We use the speed-up ratio to summarize the increase of computing power by parallel processing. We varied the mesh size and the number of control points on the boundaries to examine the effect of parallel computing under different settings.

Speed-up ratios for different setups of mesh size are presented in Figure 10. The speed-up ratio is roughly proportional to the number of concurrent processes. With a maximum of eight concurrent processes allowed by the computer, we are able to accelerate training process by a factor of at least three and half. This result does not appear to depend much on data sets, but it depends on mesh sizes. Since a smaller mesh size implies that each subdomain (or computing unit) consumes less time, parallelization works more effectively and the speed-up ratio curve bends less downward as the number of processes increases.

7. Concluding Remarks and Discussions

We develop a fast computation method for GP regression, which revises the local kriging predictor to provide consistent predictions on the boundaries of subdomains. Our DDM method inherits many advantages of the local kriging: fast computation, the use of local hyperparameters to fit spatially nonstationary data sets, and the easiness of parallel computation. Such advantages of the proposed method over other competing methods are supported by our empirical studies. Mismatch of predictions on subdomain boundaries is entirely eliminated in the 1-d case and is significantly controlled in the 2-d cases. Most importantly, DDM shows more accurate prediction using less training and testing time than other methods. Parallelization of computation for DDM also reveals

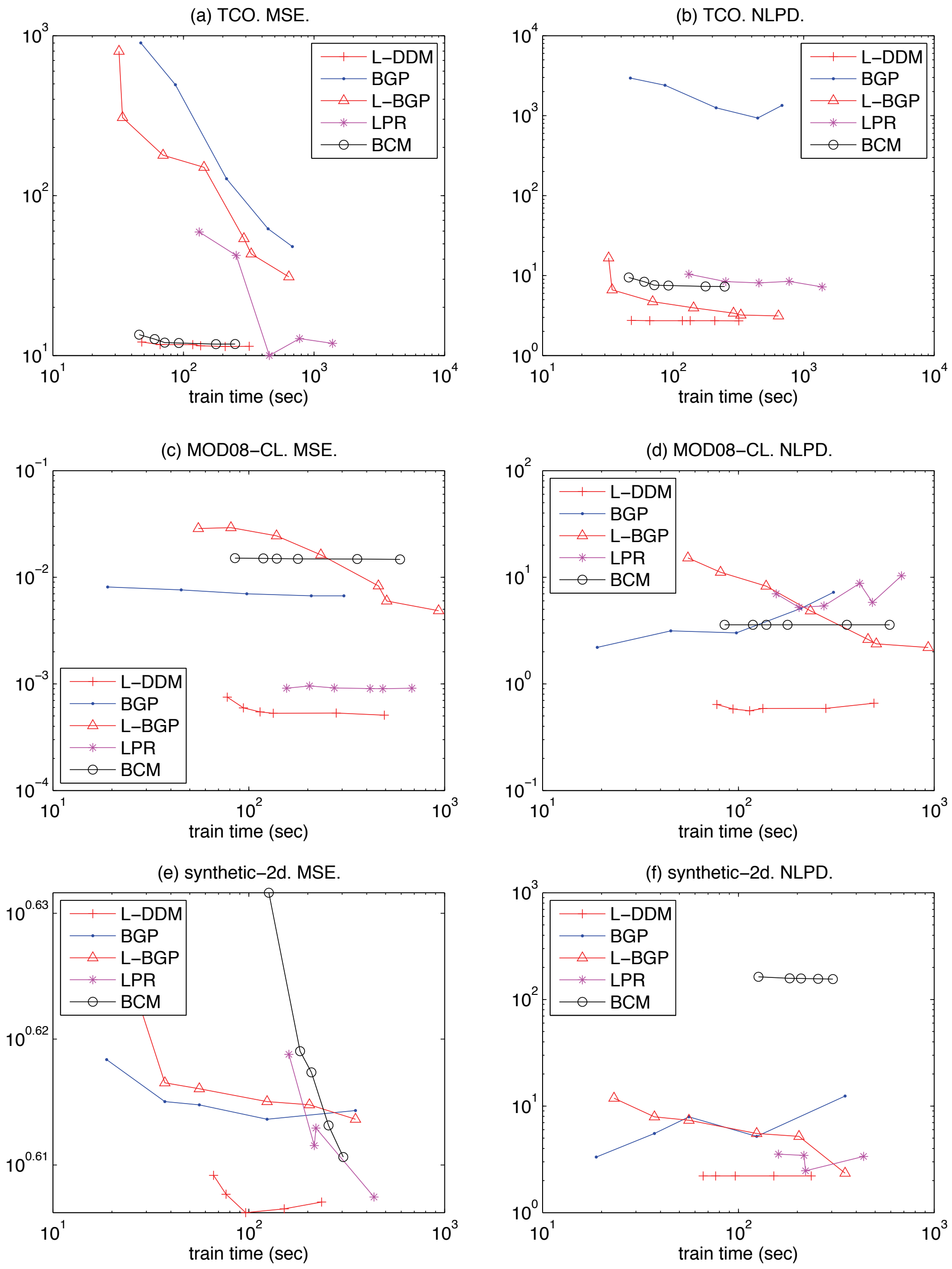


Figure 8: Prediction accuracy versus training time. Both of x-axis and y-axis are log-scaled due to big variations on values from the compared methods. For the three data sets, $m \in \{36, 56, 110, 156, 266, 638\}$ in L-DDM; $(K,M) \in \{(5,700),(10,700),(10,900),(20,900),(30,900)\}$ in BGP; $K \in \{9,16,25,36,49,64\}$ in L-BGP; $(K,R) \in \{5,10,20\} \otimes \{500,1500\}$ in LPR; $M \in \{100,150,200,250,300,600\}$ in BCM.

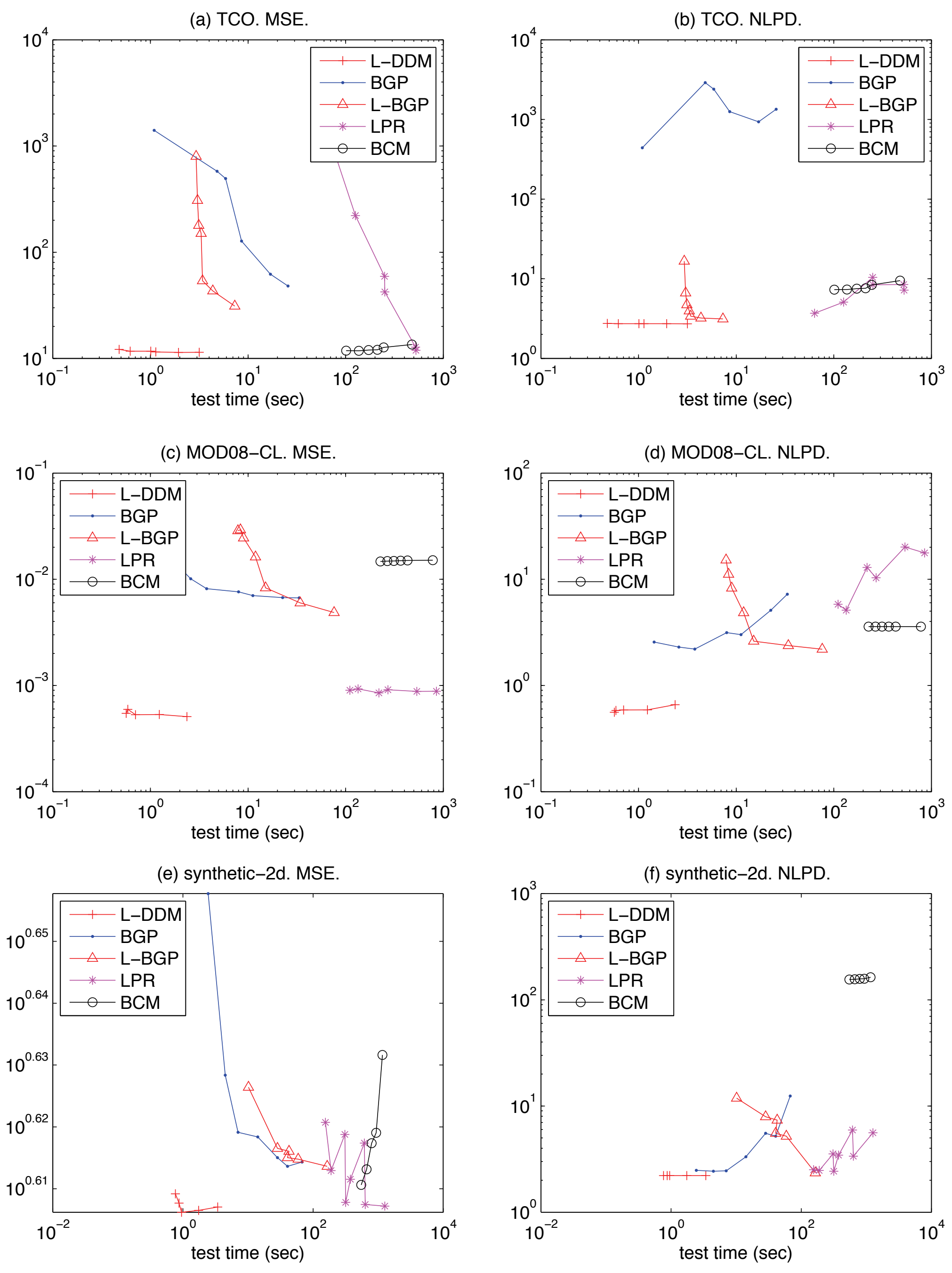


Figure 9: Prediction accuracy versus test time. Both of x-axis and y-axis are log-scaled due to big variations on values from the compared methods. For the three data sets, $m \in \{36, 56, 110, 156, 266, 638\}$ in L-DDM; $(K, M) \in \{(5, 700), (10, 700), (10, 900), (20, 900), (30, 900)\}$ in BGP; $K \in \{9, 16, 25, 36, 49, 64\}$ in L-BGP; $(K, R) \in \{5, 10, 20\} \otimes \{500, 1500\}$ in LPR; $M \in \{100, 150, 200, 250, 300, 600\}$ in BCM.

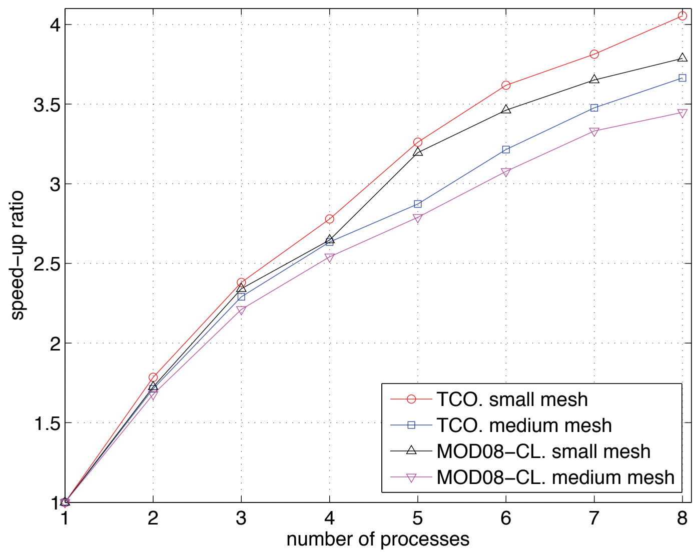


Figure 10: Speed-up ratios for different settings of parallel processing. Each line traces the training time of P-DDM when applied to a data set with a given size of meshes and number of control points.

clear benefit in time efficiency. The proposed method is specially designed to handle spatial data sets. Given the ubiquitous of large spatial data sets, our method should have wide applications.

In the meanwhile, we also acknowledge that more work is needed to fine tune the performance of the new method, including addressing the issues on meshing, hyperparameter learning, and parallelization. While extending the new method by addressing these issues is left for future research, we do want to present our thoughts regarding a possible improvement on mesh generation, for the purpose of facilitating further discussions and development.

7.1 Mesh Generation

Since meshing is a classical problem in the finite element analysis, methods in the finite element analysis literature could be helpful, or even readily applicable. A uniform mesh, as we used in this paper, works surprisingly well in many applications. However, the uniform mesh applies the equal-sized subdomains to both the slowly changing regions and the fast changing regions. Doing so may not be able to effectively adapt to local abrupt changes in the data and may lead to a large prediction error in fast changing regions. As a remedy, one can consider using the *adaptive mesh generation* (Becker and Rannacher, 2001) which adjusts the size of subdomains so that they are adaptive to local changes.

The basic idea is to start with a relatively coarse uniform mesh and to split subdomains until the approximation error is smaller than a prescribed tolerance. In each iteration followed, a certain percentage of the subdomains having higher local error estimates, for example, the top 20% of those, are split. After several iterations, local error estimates will become balanced over all the subdomains. This strategy of splitting a subdomain is called *error-balancing strategy*.

In DDM, we have a natural choice for local error estimator, which is the predictive error variance given in (19). Thus, it is possible to apply the error-balancing strategy. We can define our local error estimate using the integrated error variance as follows: for Ω_j,

$$\eta_{\Omega_j} = \int_{\Omega_j} \hat{\sigma}_j(x_*; \boldsymbol{r}_j) dx_*.$$

Since the integral is intractable, we may use the Nyström method to approximate the integral. If S_j is a set of points uniformly distributed over Ω_j, the error estimate is

$$\hat{\eta}_{\Omega_j} = \sum_{x_* \in S_j} \hat{\sigma}_j(x_*; \boldsymbol{r}_j).$$

Given the local error estimate for each subdomain, we define the overall error estimate as the summation of the local error estimates over all the subdomains, namely that $\hat{\eta} = \sum_{\Omega_j} \hat{\eta}_{\Omega_j}$, where $\hat{\eta}$ denotes the overall estimate. Thus the adaptive mesh generation in DDM could be performed as follows: Start with a coarse mesh and continue splitting the subdomains corresponding to the top $100 \cdot \alpha\%$ of the $\hat{\eta}_{\Omega_j}$'s until $\hat{\eta}$ is less than a pre-specified tolerance.

Acknowledgments

Yu Ding and Chiwoo Park were supported by the grants from NSF (CMMI-0348150, CMMI-0926803, CMMI-1000088), Texas Norman Hackerman Advanced Research Program (010366-0024-2007), and an IAMCS Innovation Award at Texas A&M University; Jianhua Z. Huang was supported by NCI (CA57030), NSF (DMS-0606580, DMS-0907170, DMS-1007618), and King Abdullah University of Science and Technology (KUS-CI-016-04).

Appendix A. Derivation of (16) for Local Predictor

With $\boldsymbol{\lambda}_j(x_*) = \boldsymbol{\Lambda}_j(\boldsymbol{k}^t_{\mathbf{x}^b_j *}\boldsymbol{k}_{\mathbf{x}^b_j *})^{-1/2}\boldsymbol{k}_{\mathbf{x}^b_j *}$, (15) and (14) can be written as

$$\boldsymbol{A}_j \boldsymbol{k}_{\mathbf{x}_j *} = (\sigma_j^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1} \left(\boldsymbol{k}_{\mathbf{x}_j *} + \frac{1}{2} \boldsymbol{y}_j (\boldsymbol{k}^t_{\mathbf{x}^b_j *}\boldsymbol{k}_{\mathbf{x}^b_j *})^{-1/2} \boldsymbol{k}^t_{\mathbf{x}^b_j *} \boldsymbol{\Lambda}_j \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j} (\boldsymbol{k}^t_{\mathbf{x}_j *}\boldsymbol{k}_{\mathbf{x}_j *})^{-1} \boldsymbol{k}_{\mathbf{x}_j *} \right), \tag{24}$$

$$\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j} \boldsymbol{A}^t_j \boldsymbol{y}_j - \boldsymbol{T}^t_j \boldsymbol{r}_j = 0, \tag{25}$$

where $\boldsymbol{\Lambda}_j$ is a $q_j \times q_j$ diagonal matrix and $\boldsymbol{\lambda}_j$ is a column vector of its diagonal elements. The expression (24) can be rewritten as

$$\boldsymbol{A}_j \boldsymbol{k}_{\mathbf{x}_j *} = (\sigma_j^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1} \left(\boldsymbol{k}_{\mathbf{x}_j *} + \frac{1}{2} \boldsymbol{y}_j \boldsymbol{\lambda}^t_j [(\boldsymbol{k}^t_{\mathbf{x}^b_j *}\boldsymbol{k}_{\mathbf{x}^b_j *})^{-1/2} \boldsymbol{k}_{\mathbf{x}^b_j *}] \circ [\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j} \boldsymbol{k}_{\mathbf{x}_j *} (\boldsymbol{k}^t_{\mathbf{x}_j *}\boldsymbol{k}_{\mathbf{x}_j *})^{-1}] \right). \tag{26}$$

Evaluating (26) at q points uniformly distributed on Γ_{jk} for $k \in N(j)$ and binding the evaluated values columnwise, we have

$$\boldsymbol{A}_j \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}^b_j} = (\sigma_j^2 \boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1} \left(\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}^b_j} + \frac{1}{2} \boldsymbol{y}_j \boldsymbol{\lambda}^t_j \boldsymbol{G}^{-1}_j \right), \tag{27}$$

where $\boldsymbol{G}_j^{-1}$ is symmetric and given by

$$\boldsymbol{G}_j^{-1} = \{\mathrm{diag}_{1/2}[(\boldsymbol{K}^t_{\mathbf{x}^b_j\mathbf{x}^b_j}\boldsymbol{K}_{\mathbf{x}^b_j\mathbf{x}^b_j})^{-1}]\boldsymbol{K}_{\mathbf{x}^b_j\mathbf{x}^b_j}\} \circ \{\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j}\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}^b_j}\mathrm{diag}[(\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j}\boldsymbol{K}_{\mathbf{x}_j\mathbf{x}^b_j})^{-1}]\}.$$

Substitute the transpose of (27) into (25) to get

$$\left(\boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j} + \frac{1}{2}\boldsymbol{G}_j^{-1}\boldsymbol{\lambda}_j\boldsymbol{y}_j^t\right)(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{y}_j = \boldsymbol{T}_j^t\boldsymbol{r}_j.$$

After some simple algebra, we obtain the optimal $\boldsymbol{\lambda}_j$ value

$$\boldsymbol{\lambda}_j = 2\boldsymbol{G}_j\frac{\boldsymbol{T}_j^t\boldsymbol{r}_j - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_j}(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{y}_j}{\boldsymbol{y}_j^t(\sigma_j^2\boldsymbol{I} + \boldsymbol{K}_{\mathbf{x}_j\mathbf{x}_j})^{-1}\boldsymbol{y}_j}.$$

Appendix B. Derivation of (21) for Interface Equation

Note that $\boldsymbol{T}_j^t\boldsymbol{r}_j$ is a rowwise binding of $\boldsymbol{T}_{jk}^t\boldsymbol{r}_{jk}$. Ignoring a constant, the objective function to be minimized can be written as

$$\sum_{j=1}^m \frac{1}{\boldsymbol{h}_j^t\boldsymbol{y}_j}\sum_{k\in N(j)}(\boldsymbol{T}_{jk}^t\boldsymbol{r}_{jk} - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_{jk}}\boldsymbol{h}_j)^t(\boldsymbol{T}_{jk}^t\boldsymbol{r}_{jk} - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_{jk}}\boldsymbol{h}_j). \tag{28}$$

To find the optimal $\boldsymbol{r}_{jk}$, we only need pay attention to the relevant terms in (28). Since $\boldsymbol{r}_{jk} = \boldsymbol{r}_{kj}$ and $\boldsymbol{T}_{jk} = \boldsymbol{T}_{kj}$, the objective function for finding optimal $\boldsymbol{r}_{jk}$ reduces to

$$\frac{1}{\boldsymbol{h}_j^t\boldsymbol{y}_j}(\boldsymbol{T}_{jk}^t\boldsymbol{r}_{jk} - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_{jk}}\boldsymbol{h}_j)^t(\boldsymbol{T}_{jk}^t\boldsymbol{r}_{jk} - \boldsymbol{K}^t_{\mathbf{x}_j\mathbf{x}^b_{jk}}\boldsymbol{h}_j)$$
$$+ \frac{1}{\boldsymbol{h}_k^t\boldsymbol{y}_k}(\boldsymbol{T}_{kj}^t\boldsymbol{r}_{kj} - \boldsymbol{K}^t_{\mathbf{x}_k\mathbf{x}^b_{kj}}\boldsymbol{h}_k)^t(\boldsymbol{T}_{kj}^t\boldsymbol{r}_{kj} - \boldsymbol{K}^t_{\mathbf{x}_k\mathbf{x}^b_{kj}}\boldsymbol{h}_k),$$

the minimization of which gives (21).

References

Roland Becker and Rolf Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–102, 2001.

Tao Chen and Jianghong Ren. Bagging for Gaussian process regression. *Neurocomputing*, 72(7-9): 1605–1610, 2009.

Alexandre Ern and Jean-Luc Guermond. *Theory and Practice of Finite Elements*. Springer, 2004.

Reinhard Furrer, Marc G. Genton, and Douglas Nychka. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3):502–523, 2006.

Tilmann Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83(2):493–508, 2002.

Robert B. Gramacy and Herbert K. H. Lee. Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

Alfio Quarteroni and Alberto Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, 1999.

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

Carl E. Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems 14*, pages 881–888. MIT Press, 2002.

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

Anton Schwaighofer, Marian Grigoras, Volker Tresp, and Clemens Hoffmann. Transductive and inductive methods for approximate Gaussian process regression. In *Advances in Neural Information Processing Systems 16*, pages 977–984. MIT Press, 2003.

Matthias Seeger, Christopher K. I. Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics 9*. Society for Artificial Intelligence and Statistics, 2003.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2006.

Edward Snelson and Zoubin Ghahramani. Local and global sparse Gaussian process approximations. In *International Conference on Artifical Intelligence and Statistics 11*, pages 524–531. Society for Artificial Intelligence and Statistics, 2007.

Volker Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

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

Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 12*, pages 682–688. MIT Press, 2000.

Journal of Machine Learning Research 12 (2011) 1729-1770 Submitted 10/08; Revised 11/10; Published 5/11

A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes

Stéphane Ross STEPHANEROSS@CMU.EDU
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA 15213

Joelle Pineau JPINEAU@CS.MCGILL.CA
School of Computer Science
McGill University
Montréal, PQ, Canada H3A 2A7

Brahim Chaib-draa CHAIB@IFT.ULAVAL.CA
Computer Science & Software Engineering Dept
Laval University
Québec, PQ, Canada G1K 7P4

Pierre Kreitmann PIERRE.KREITMANN@GMAIL.COM
Department of Computer Science
Stanford University
Stanford, CA, USA 94305

Editor: Satinder Baveja

Abstract

Bayesian learning methods have recently been shown to provide an elegant solution to the exploration-exploitation trade-off in reinforcement learning. However most investigations of Bayesian reinforcement learning to date focus on the standard Markov Decision Processes (MDPs). The primary focus of this paper is to extend these ideas to the case of partially observable domains, by introducing the Bayes-Adaptive Partially Observable Markov Decision Processes. This new framework can be used to simultaneously (1) learn a model of the POMDP domain through interaction with the environment, (2) track the state of the system under partial observability, and (3) plan (near-)optimal sequences of actions. An important contribution of this paper is to provide theoretical results showing how the model can be finitely approximated while preserving good learning performance. We present approximate algorithms for belief tracking and planning in this model, as well as empirical results that illustrate how the model estimate and agent's return improve as a function of experience.

Keywords: reinforcement learning, Bayesian inference, partially observable Markov decision processes

1. Introduction

Robust decision-making is a core component of many autonomous agents. This generally requires that an agent evaluate a set of possible actions, and choose the best one for its current situation. In many problems, actions have long-term consequences that must be considered by the agent; it is not useful to simply choose the action that looks the best in the immediate situation. Instead, the agent must choose its actions by carefully trading off their short-term and long-term costs and benefits. To do so, the agent must be able to predict the consequences of its actions, in so far as it is useful to determine future actions. In applications where it is not possible to predict exactly the outcomes of an action, the agent must also consider the uncertainty over possible future events.

Probabilistic models of sequential decision-making take into account such uncertainty by specifying the chance (probability) that any future outcome will occur, given any current configuration (state) of the system, and action taken by the agent. However, if the model used does not perfectly fit the real problem, the agent risks making poor decisions. This is currently an important limitation in practical deployment of autonomous decision-making agents, since available models are often crude and incomplete approximations of reality. Clearly, learning methods can play an important role in improving the model as experience is acquired, such that the agent's future decisions are also improved.

In the past few decades, Reinforcement Learning (RL) has emerged as an elegant and popular technique to handle sequential decision problems when the model is unknown (Sutton and Barto, 1998). Reinforcement learning is a general technique that allows an agent to learn the best way to behave, that is, such as to maximize expected return, from repeated interactions in the environment. A fundamental problem in RL is that of exploration-exploitation: namely, how should the agent chooses actions during the learning phase, in order to both maximize its knowledge of the model as needed to better achieve the objective later (i.e., *explore*), and maximize current achievement of the objective based on what is already known about the domain (i.e., *exploit*). Under some (reasonably general) conditions on the exploratory behavior, it has been shown that RL eventually learns the optimal action-select behavior. However, these conditions do not specify how to choose actions such as to maximize utilities throughout the life of the agent, including during the learning phase, as well as beyond.

Model-based Bayesian RL is an extension of RL that has gained significant interest from the AI community recently as it provides a principled approach to tackle the problem of exploration-exploitation during learning and beyond, within the standard Bayesian inference paradigm. In this framework, prior information about the problem (including uncertainty) is represented in parametric form, and Bayesian inference is used to incorporate any new information about the model. Thus the exploration-exploitation problem can be handled as an explicit sequential decision problem, where the agent seeks to maximize future expected return with respect to its current uncertainty on the model. An important limitation of this approach is that the decision-making process is significantly more complex since it involves reasoning about all possible models *and* courses of action. In addition, most work to date on this framework has been limited to cases where full knowledge of the agent's state is available at every time step (Dearden et al., 1999; Strens, 2000; Duff, 2002; Wang et al., 2005; Poupart et al., 2006; Castro and Precup, 2007; Delage and Mannor, 2007).

The primary contribution of this paper is an extension of the model-based Bayesian reinforcement learning to partially observable domains with discrete representations.[1] In support of this, we introduce a new mathematical model, called the *Bayes-Adaptive POMDP* (BAPOMDP). This is a model-based Bayesian RL approach, meaning that the framework maintains a posterior over the pa-

1. A preliminary version of this model was described by Ross et al. (2008a). The current paper provides an in-depth development of this model, as well as novel theoretical analysis and new empirical results.

rameters of the underlying POMDP domain.[2] We derive optimal algorithms for belief tracking and finite-horizon planning in this model. However, because the size of the state space in a BAPOMD can be countably infinite, these are, for all practical purposes, intractable. We therefore dedicate substantial attention to the problem of approximating the BAPOMDP model. We provide theoretical results for bounding the state space while preserving the value function. These results are leveraged to derive a novel belief monitoring algorithm, which is used to maintain a posterior over both model parameters, and state of the system. Finally, we describe an online planning algorithm which provides the core sequential decision-making component of the model. Both the belief tracking and planning algorithms are parameterized so as to allow a trade-off between computational time and accuracy, such that the algorithms can be applied in real-time settings.

An in-depth empirical validation of the algorithms on challenging real-world scenarios is outside the scope of this paper, since our focus here is on the theoretical properties of the exact and approximative approaches. Nonetheless we elaborate a tractable approach and characterize its performance in three contrasting problem domains. Empirical results show that the BAPOMDP agent is able to learn good POMDP models and improve its return as it learns better model estimates. Experiments on the two smaller domains illustrate performance of the novel belief tracking algorithm, in comparison to the well-known Monte-Carlo approximation methods. Experiments on the third domain confirm good planning and learning performance on a larger domain; we also analyze the impact of the choice of prior on the results.

The paper is organized as follows. Section 2 presents the models and methods necessary for Bayesian reinforcement learning in the fully observable case. Section 3 extends these ideas to the case of partially observable domains, focusing on the definition of the BAPOMDP model and exact algorithms. Section 4 defines a finite approximation of the BAPOMDP model that could be used to be solved by finite offline POMDP solvers. Section 5 presents a more tractable approach to solving the BAPOMDP model based on online POMDP solvers. Section 6 illustrates the empirical performance of the latter approach on sample domains. Finally, Section 7 discusses related Bayesian approaches for simultaneous planning and learning in partially observable domains.

2. Background and Notation

In this section we discuss the problem of model-based Bayesian reinforcement learning in the fully observable case, in preparation for the extension of these ideas to the partially observable case which is presented in Section 3. We begin with a quick review of Markov Decision Processes. We then present the models and methods necessary for Bayesian RL in MDPs. This literature has been developing over the last decade, and we aim to provide a brief but comprehensive survey of the models and algorithms in this area. Readers interested in a more detailed presentation of the material should seek additional references (Sutton and Barto, 1998; Duff, 2002).

2.1 Markov Decision Processes

We consider finite MDPs as defined by the following n-tuple (S,A,T,R,γ):

States: S is a finite set of states, which represents all possible configurations of the system. A state is essentially a sufficient statistic of what occurred in the past, such that what will occur in

2. This is in contrast to model-free Bayesian RL approaches, which maintain a posterior over the value function, for example, Engel et al. (2003, 2005); Ghavamzadeh and Engel (2007b).

the future only depends on the current state. For example, in a navigation task, the state is usually the current position of the agent, since its next position usually only depends on the current position, and not on previous positions.

Actions: A is a finite set of actions the agent can make in the system. These actions may influence the next state of the system and have different costs/payoffs.

Transition Probabilities: $T : S \times A \times S \to [0,1]$ is called the transition function. It models the uncertainty on the future state of the system. Given the current state s, and an action a executed by the agent, $T^{sas'}$ specifies the probability $\Pr(s'|s,a)$ of moving to state s'. For a fixed current state s and action a, $T^{sa\cdot}$ defines a probability distribution over the next state s', such that $\sum_{s' \in S} T^{sas'} = 1$, for all (s,a). The definition of T is based on the *Markov assumption*, which states that the transition probabilities only depend on the current state and action, that is, $\Pr(s_{t+1} = s'|a_t, s_t, \ldots, a_0, s_0) = \Pr(s_{t+1} = s'|a_t, s_t)$, where a_t and s_t denote respectively the action and state at time t. It is also assumed that T is time-homogenous, that is, the transition probabilities do not depend on the current time: $\Pr(s_{t+1} = s'|a_t = a, s_t = s) = \Pr(s_t = s'|a_{t-1} = a, s_{t-1} = s)$ for all t.

Rewards: $R : S \times A \to \mathbb{R}$ is the function which specifies the reward $R(s,a)$ obtained by the agent for doing a particular action a in current state s. This models the immediate costs (negative rewards) and payoffs (positive rewards) incurred by performing different actions in the system.

Discount Factor: $\gamma \in [0,1)$ is a discount rate which allows a trade-off between short-term and long-term rewards. A reward obtained t-steps in the future is discounted by the factor γ^t. Intuitively, this indicates that it is better to obtain a given reward now, rather than later in the future.

Initially, the agent starts in some initial state, $s_0 \in S$. Then at any time t, the agent chooses an action $a_t \in A$, performs it in the current state s_t, receives the reward $R(s_t, a_t)$ and moves to the next state s_{t+1} with probability $T^{s_t a_t s_{t+1}}$. This process is iterated until termination; the task horizon can be specified *a priori*, or determined by the discount factor.

We define a **policy**, $\pi : S \to A$, to be a mapping from states to actions. The optimal policy, denoted π^*, corresponds to the mapping which maximizes the expected sum of discounted rewards over a trajectory. The **value** of the optimal policy is defined by Bellman's equation:

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T^{sas'} V^*(s') \right].$$

The optimal policy at a given state, $\pi^*(s)$, is defined to be the action that maximizes the value at that state, $V^*(s)$. Thus the main objective of the MDP framework is to accurately estimate this value function, so as to then obtain the optimal policy. There is a large literature on the computational techniques that can be leveraged to solve this problem. A good starting point is the recent text by Szepesvari (2010).

A key aspect of reinforcement learning is the issue of *exploration*. This corresponds to the question of determining how the agent should choose actions while learning about the task. This is in contrast to the phase called *exploitation*, through which actions are selected so as to maximize expected reward with respect to the current value function estimate. The issues of value function estimation and exploration are assumed to be orthogonal in much of the MDP literature. However in many applications, where data is expensive or difficult to acquire, it is important to consider the rewards accumulated during the learning phase, and to try to take this cost-of-learning into account in the optimization of the policy.

In RL, most practical work uses a variety of heuristics to balance the exploration and exploitation, including for example the well-known ε-greedy and Boltzmann strategies. The main problem with such heuristic methods is that the exploration occurs randomly and is not focused on what needs to be learned.

More recently, it has been shown that it is possible for an agent to reach near-optimal performance with high probability using only a polynomial number of steps (Kearns and Singh, 1998; Brafman and Tennenholtz, 2003; Strehl and Littman, 2005), or alternately to have small regret with respect to the optimal policy (Auer and Ortner, 2006; Tewari and Bartlett, 2008; Auer et al., 2009). Such theoretical results are highly encouraging, and in some cases lead to algorithms which exhibit reasonably good empirical performance.

2.2 Bayesian Learning

Bayesian Learning (or Bayesian Inference) is a general technique for learning the unknown parameters of a probability model from observations generated by this model. In Bayesian learning, a probability distribution is maintained over all possible values of the unknown parameters. As observations are made, this probability distribution is updated via Bayes' rule, and probability density increases around the most likely parameter values.

Formally, consider a random variable X with probability density $f_{X|\Theta}$ over its domain $\mathcal{X}$ parameterized by the unknown vector of parameters Θ in some parameter space $\mathcal{P}$. Let $X_1, X_2, \cdots, X_n$ be a random i.i.d. sample from $f_{X|\Theta}$. Then by Bayes' rule, the posterior probability density $g_{\Theta|X_1,X_2,\ldots,X_n}(\theta|x_1,x_2,\ldots,x_n)$ of the parameters $\Theta = \theta$, after the observations of $X_1 = x_1, X_2 = x_2, \cdots,$ $X_n = x_n$, is:

$$g_{\Theta|X_1,X_2,\ldots,X_n}(\theta|x_1,x_2,\ldots,x_n) = \frac{g_\Theta(\theta)\prod_{i=1}^n f_{X|\Theta}(x_i|\theta)}{\int_{\mathcal{P}} g_\Theta(\theta')\prod_{i=1}^n f_{X|\Theta}(x_i|\theta')d\theta'},$$

where $g_\Theta(\theta)$ is the prior probability density of $\Theta = \theta$, that is, g_Θ over the parameter space $\mathcal{P}$ is a distribution that represents the initial belief (or uncertainty) on the values of Θ. Note that the posterior can be defined recursively as follows:

$$g_{\Theta|X_1,X_2,\ldots,X_n}(\theta|x_1,x_2,\ldots,x_n) = \frac{g_{\Theta|X_1,X_2,\ldots,X_{n-1}}(\theta|x_1,x_2,\ldots,x_{n-1})f_{X|\Theta}(x_n|\theta)}{\int_{\mathcal{P}} g_{\Theta|X_1,X_2,\ldots,X_{n-1}}(\theta'|x_1,x_2,\ldots,x_{n-1})f_{X|\Theta}(x_n|\theta')d\theta'},$$

so that whenever we get the n^{th} observation of X, denoted x_n, we can compute the new posterior distribution $g_{\Theta|X_1,X_2,\ldots,X_n}$ from the previous posterior $g_{\Theta|X_1,X_2,\ldots,X_{n-1}}$.

In general, updating the posterior distribution $g_{\Theta|X_1,X_2,\ldots,X_n}$ is difficult due to the need to compute the normalization constant $\int_{\mathcal{P}} g_\Theta(\theta)\prod_{i=1}^n f_{X|\Theta}(x_i|\theta)d\theta$. However, for conjugate family distributions, updating the posterior can be achieved very efficiently with a simple update of the parameters defining the posterior distribution (Casella and Berger, 2001).

Formally, consider a particular class $\mathcal{G}$ of prior distributions over the parameter space $\mathcal{P}$, and a class $\mathcal{F}$ of likelihood functions $f_{X|\Theta}$ over $\mathcal{X}$ parameterized by parameters $\Theta \in \mathcal{P}$, then $\mathcal{F}$ and $\mathcal{G}$ are said to be conjugate if for any choice of prior $g_\Theta \in \mathcal{G}$, likelihood $f_{X|\Theta} \in \mathcal{F}$ and observation $X = x$, the posterior distribution $g_{\Theta|X}$ after observation of $X = x$ is also in $\mathcal{G}$.

For example, the Beta distribution[3] is conjugate to the Binomial distribution.[4] Consider $X \sim Binomial(n, p)$ with unknown probability parameter p, and consider a prior $Beta(\alpha, \beta)$ over the unknown value of p. Then following an observation $X = x$, the posterior over p is also Beta distributed and is defined by $Beta(\alpha + x, \beta + n - x)$.

Another important issue with Bayesian methods is the need to specify a prior. While the influence of the prior tends to be negligible when provided with a large amount of data, its choice is particularly important for any inference and decision-making performed when only a small amount of data has been observed. In many practical problems, informative priors can be obtained from domain knowledge. For example many sensors and actuators used in engineering applications have specified confidence intervals on their accuracy provided by the manufacturer. In other applications, such as medical treatment design or portfolio management, data about the problem may have been collected for other tasks, which can guide the construction of the prior.

In the absence of any knowledge, uninformative priors can be specified. Under such priors, any inference done *a posteriori* is dominated by the data, that is, the influence of the prior is minimal. A common uninformative prior consists of using a distribution that is constant over the whole parameter space, such that every possible parameter has equal probability density. From an information theoretic point of view, such priors have maximum entropy and thus contain the least amount of information about the true parameter (Jaynes, 1968). However, one problem with such uniform priors is that typically, under different re-parameterization, one has different amounts of information about the unknown parameters. A preferred uninformative prior, which is invariant under reparameterization, is Jeffreys' prior (Jeffreys, 1961).

The third issue of concern with Bayesian methods concerns the convergence of the posterior towards the true parameter of the system. In general, the posterior density concentrates around the parameters that have highest likelihood of generating the observed data in the limit. For finite parameter spaces, and for smooth families with continuous finite dimensional parameter spaces, the posterior converges towards the true parameter as long as the prior assigns non-zero probability to every neighborhood of the true parameter. Hence in practice, it is often desirable to assign non-zero prior density over the full parameter space.

It should also be noted that if multiple parameters within the parameter space can generate the observed data with equal likelihood, then the posterior distribution will usually be multimodal, with one mode surrounding each equally likely parameter. In such cases, it may be impossible to identify the true underlying parameter. However for practical purposes, such as making predictions about future observations, it is sufficient to identify any of the equally likely parameters.

Lastly, another concern is how fast the posterior converges towards the true parameters. This is mostly influenced by how far the prior is from the true parameter. If the prior is poor, that is, it assigns most probability density to parameters far from the true parameters, then it will take much more data to learn the correct parameter than if the prior assigns most probability density around the

3. $Beta(\alpha, \beta)$ is defined by the density function $f(p|\alpha, \beta) \propto p^{\alpha-1}(1-p)^{\beta-1}$ for $p \in [0, 1]$ and parameters $\alpha, \beta \geq 0$.
4. $Binomial(n, p)$ is defined by the density function $f(k|n, p) \propto p^k(1-p)^{n-k}$ for $k \in \{0, 1, \ldots, n\}$ and parameters $p \in [0, 1], n \in \mathbb{N}$.

true parameter. For such reasons, a safe choice is to use an uninformative prior, unless some data is already available for the problem at hand.

2.3 Bayesian Reinforcement Learning in Markov Decision Processes

Work on model-based Bayesian reinforcement learning dates back to the days of Bellman, who studied this problem under the name of Adaptive control processes (Bellman, 1961). An excellent review of the literature on model-based Bayesian RL is provided in Duff (2002). This paper outlines, where appropriate, more recent contributions in this area.

As a side note, model-free BRL methods also exist (Engel et al., 2003, 2005; Ghavamzadeh and Engel, 2007a,b). Instead of representing the uncertainty on the model, these methods explicitly model the uncertainty on the value function or optimal policy. These methods must often rely on heuristics to handle the exploration-exploitation trade-off, but may be useful in cases where it is easier to express prior knowledge about initial uncertainty on the value function or policy, rather than on the model.

The main idea behind model-based BRL is to use Bayesian learning methods to learn the unknown model parameters of the system, based on what is observed by the agent in the environment. Starting from a prior distribution over the unknown model parameters, the agent updates a posterior distribution over these parameters as it performs actions and gets observations from the environment. Under such a Bayesian approach, the agent can compute the best action-selection strategy by finding the one that maximizes its future expected return under the current posterior distribution, but also considering how this distribution will evolve in the future under different possible sequences of actions and observations.

To formalize these ideas, consider an MDP (S,A,T,R,γ), where S, A and R are known, and T is unknown. Furthermore, assume that S and A are finite. The unknown parameters in this case are the transition probabilities, $T^{sas'}$, for all $s,s' \in S$, $a \in A$. The model-based BRL approach to this problem is to start off with a prior, g, over the space of transition functions, T. Now let $s_t = (s_0,s_1,\dots,s_t)$ and $a_{t-1} = (a_0,a_1,\dots,a_{t-1})$ denote the agent's history of visited states and actions up to time t. Then the posterior over transition functions after this sequence is defined by:

$$\begin{array}{rcl} g(T|s_t,a_{t-1}) & \propto & g(T)\prod_{i=0}^{t-1} T^{s_i a_i s_{i+1}} \\ & \propto & g(T)\prod_{s\in S,a\in A}\prod_{s'\in S}(T^{sas'})^{N^a_{s,s'}(s_t,a_{t-1})}, \end{array}$$

where $N^a_{s,s'}(s_t,a_{t-1}) = \sum_{i=0}^{t-1} I_{\{(s,a,s')\}}(s_i,a_i,s_{i+1})$ is the number of times[5] the transition (s,a,s') occurred in the history (s_t,a_{t-1}). As we can see from this equation, the likelihood $\prod_{s\in S,a\in A}\prod_{s'\in S}(T^{sas'})^{N^a_{s,s'}(s_t,a_{t-1})}$ is a product of $|S||A|$ independent Multinomial[6] distributions over S. Hence, if we define the prior g as a product of $|S||A|$ independent priors over each distribution over next states $T^{sa\cdot}$, that is, $g(T) = \prod_{s\in S,a\in A} g_{s,a}(T^{sa\cdot})$, then the posterior is also defined as a product of $|S||A|$ independent posterior distributions: $g(T|s_t,a_{t-1}) = \prod_{s\in S,a\in A} g_{s,a}(T^{sa\cdot}|s_t,a_{t-1})$, where $g_{s,a}(T^{sa\cdot}|s_t,a_{t-1})$ is defined as:

$$g_{s,a}(T^{sa\cdot}|s_t,a_{t-1}) \propto g_{s,a}(T^{sa\cdot})\prod_{s'\in S}(T^{sas'})^{N^a_{s,s'}(s_t,a_{t-1})}.$$

5. We use $I()$ to denote the indicator function.

6. $Multinomial_k(p,N)$ is defined by the density function $f(\mathbf{n}|p,N) \propto \prod_{i=1}^{k} p_i^{n_i}$ for $n_i \in \{0,1,\dots,N\}$ such that $\sum_{i=1}^{k} n_i = N$, parameters $N \in \mathbb{N}$, and p is a discrete distribution over k outcomes.

Furthermore, since the Dirichlet distribution is the conjugate of the Multinomial, it follows that if the priors $g_{s,a}(T^{sa\cdot})$ are Dirichlet distributions for all s,a, then the posteriors $g_{s,a}(T^{sa\cdot}|s_t,a_{t-1})$ will also be Dirichlet distributions for all s,a. The Dirichlet distribution is the multivariate extension of the Beta distribution and defines a probability distribution over discrete distributions. It is parameterized by a count vector, $\phi = (\phi_1,\dots,\phi_k)$, where $\phi_i \geq 0$, such that the density of probability distribution $p = (p_1,\dots,p_k)$ is defined as $f(p|\phi) \propto \prod_{i=1}^{k} p_i^{\phi_i - 1}$. If $X \sim Multinomial_k(p,N)$ is a random variable with unknown probability distribution $p = (p_1,\dots,p_k)$, and $Dirichlet(\phi_1,\dots,\phi_k)$ is a prior over p, then after the observation of $X = \mathbf{n}$, the posterior over p is $Dirichlet(\phi_1 + n_1,\dots,\phi_k + n_k)$. Hence, if the prior $g_{s,a}(T^{sa\cdot})$ is $Dirichlet(\phi^a_{s,s_1},\dots,\phi^a_{s,s_{|S|}})$, then after the observation of history (s_t,a_{t-1}), the posterior $g_{s,a}(T^{sa\cdot}|s_t,a_{t-1})$ is $Dirichlet(\phi^a_{s,s_1} + N^a_{s,s_1}(s_t,a_{t-1}),\dots,\phi^a_{s,s_{|S|}} + N^a_{s,s_{|S|}}(s_t,a_{t-1}))$. It follows that if $\phi = \{\phi^a_{s,s'}|a \in A, s,s' \in S\}$ represents the set of all Dirichlet parameters defining the current prior/posterior over T, then if the agent performs a transition (s,a,s'), the posterior Dirichlet parameters ϕ' after this transition are simply defined as:

$$\begin{aligned}
\phi'^a_{s,s'} &= \phi^a_{s,s'} + 1,\\
\phi'^{a'}_{s'',s'''} &= \phi^{a'}_{s'',s'''}, \forall (s'',a',s''') \neq (s,a,s').
\end{aligned}$$

We denote this update by the function $\mathcal{U}$, where $\mathcal{U}(\phi,s,a,s')$ returns the set ϕ' as updated in the previous equation.

Because of this convenience, most authors assume that the prior over the transition function T follows the previous independence and Dirichlet assumptions (Duff, 2002; Dearden et al., 1999; Wang et al., 2005; Castro and Precup, 2007). We also make such assumptions throughout this paper.

2.3.1 BAYES-ADAPTIVE MDP MODEL

The core sequential decision-making problem of model-based Bayesian RL can be cast as the problem of finding a policy that maps extended states of the form (s,ϕ) to actions $a \in A$, such as to maximize the long-term rewards of the agent. If this decision problem can be modeled as an MDP over extended states (s,ϕ), then by solving this new MDP, we would find such an optimal policy. We now explain how to construct this MDP.

Consider a new MDP defined by the tuple (S',A,T',R',γ). We define the new set of states $S' = S \times \mathcal{T}$, where $\mathcal{T} = \{\phi \in \mathbb{N}^{|S|^2|A|} | \forall (s,a) \in S \times A, \sum_{s' \in S} \phi^a_{ss'} > 0\}$, and A is the original action space. Here, the constraints on the set $\mathcal{T}$ of possible count parameters ϕ are only needed to ensure that the transition probabilities are well defined. To avoid confusion, we refer to the extended states $(s,\phi) \in S'$ as hyperstates. Also note that the next information state ϕ' only depends on the previous information state ϕ and the transition (s,a,s') that occurred in the physical system, so that transitions between hyperstates also exhibit the Markov property. Since we want the agent to maximize the rewards it obtains in the physical system, the reward function R' should return the same reward as in the physical system, as defined in R. Thus we define $R'(s,\phi,a) = R(s,a)$. The only remaining issue is to define the transition probabilities between hyperstates. The new transition function T' must specify the transition probabilities $T'(s,\phi,a,s',\phi') = \Pr(s',\phi'|s,a,\phi)$. By the chain rule, $\Pr(s',\phi'|s,a,\phi) = \Pr(s'|s,a,\phi)\Pr(\phi'|s,a,s',\phi)$. Since the update of the information state ϕ to ϕ' is deterministic, then $\Pr(\phi'|s,a,s',\phi)$ is either 0 or 1, depending on whether $\phi' = \mathcal{U}(\phi,s,a,s')$ or not. Hence $\Pr(\phi'|s,a,s',\phi) = I_{\{\phi'\}}(\mathcal{U}(\phi,s,a,s'))$. By the law of total probability, $\Pr(s'|s,a,\phi) = \int \Pr(s'|s,a,T,\phi) f(T|\phi) dT = \int T^{sas'} f(T|\phi) dT$, where the integral is carried over transition function T, and $f(T|\phi)$ is the probability density of transition function T under the posterior defined by

ϕ. The term $\int T^{sas'} f(T|\phi)dT$ is the expectation of $T^{sas'}$ for the Dirichlet posterior defined by the parameters $\phi^a_{s,s_1}, \ldots, \phi^a_{s,s_{|S|}}$, which corresponds to $\frac{\phi^a_{s,s'}}{\sum_{s'' \in S} \phi^a_{s,s''}}$. Thus it follows that:

$$T'(s, \phi, a, s', \phi') = \frac{\phi^a_{s,s'}}{\sum_{s'' \in S} \phi^a_{s,s''}} I_{\{\phi'\}}(\mathcal{U}(\phi, s, a, s')).$$

We now have a new MDP with a known model. By solving this MDP, we can find the optimal action-selection strategy, given *a posteriori* knowledge of the environment. This new MDP has been called the Bayes-Adaptive MDP (Duff, 2002) or the HyperMDP (Castro and Precup, 2007).

Notice that while we have assumed that the reward function R is known, this BRL framework can easily be extended to the case where R is unknown. In such a case, one can proceed similarly by using a Bayesian learning method to learn the reward function R and add the posterior parameters for R in the hyperstate. The new reward function R' then becomes the expected reward under the current posterior over R, and the transition function T' would also model how to update the posterior over R, upon observation of any reward r. For brevity of presentation, it is assumed that the reward function is known throughout this paper. However, the frameworks we present in the following sections can also be extended to handle cases where the rewards are unknown, by following a similar reasoning.

2.3.2 OPTIMALITY AND VALUE FUNCTION

The Bayes-Adaptive MDP (BAMDP) is just a conventional MDP with a countably infinite number of states. Fortunately, many theoretical results derived for standard MDPs carry over to the Bayes-Adaptive MDP model (Duff, 2002). Hence, we know there exists an optimal deterministic policy $\pi^* : S' \to A$, and that its value function is defined by:

$$\begin{array}{rcl} V^*(s,\phi) & = & \max_{a \in A} \left[R'(s,\phi,a) + \gamma \sum_{(s',\phi') \in S'} T'(s,\phi,a,s',\phi') V^*(s',\phi') \right] \\ & = & \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} \frac{\phi^a_{s,s'}}{\sum_{s'' \in S} \phi^a_{s,s''}} V^*(s', \mathcal{U}(\phi,s,a,s')) \right]. \end{array} \quad (1)$$

This value function is defined over an infinite number of hyperstates, therefore, in practice, computing V^* exactly for all hyperstates is unfeasible. However, since the summation over S is finite, we observe that from one given hyperstate, the agent can transit only to a finite number of hyperstates in one step. It follows that for any finite planning horizon t, one can compute exactly the optimal value function for a particular starting hyperstate. However the number of reachable hyperstates grows exponentially with the planning horizon.

2.3.3 PLANNING ALGORITHMS

We now review existing approximate algorithms for estimating the value function in the BAMDP. Dearden et al. (1999) proposed one of the first Bayesian model-based exploration methods for RL. Instead of solving the BAMDP directly via Equation 1, the Dirichlet distributions are used to compute a distribution over the state-action values $Q^*(s,a)$, in order to select the action that has the highest expected return and value of information (Dearden et al., 1998). The distribution over Q-values is estimated by sampling MDPs from the posterior Dirichlet distribution, and then solving each sampled MDP to obtain different sampled Q-values. Re-sampling and importance sampling techniques are proposed to update the estimated Q-value distribution as the Dirichlet posteriors are updated.

Rather than using a maximum likelihood estimate for the underlying process, Strens (2000) proposes to fully represent the posterior distribution over process parameters. He then uses a greedy behavior with respect to a sample from this posterior. By doing so, he retains each hypothesis over a period of time, ensuring goal-directed exploratory behavior without the need to use approximate measures or heuristic exploration as other approaches did. The number of steps for which each hypothesis is retained limits the length of exploration sequences. The results of this method is then an automatic way of obtaining behavior which moves gradually from exploration to exploitation, without using heuristics.

Duff (2001) suggests using Finite-State Controllers (FSC) to represent compactly the optimal policy π^* of the BAMDP and then finding the best FSC in the space of FSCs of some bounded size. A gradient descent algorithm is presented to optimize the FSC and a Monte-Carlo gradient estimation is proposed to make it more tractable. This approach presupposes the existence of a good FSC representation for the policy.

For their part, Wang et al. (2005) present an online planning algorithm that estimates the optimal value function of the BAMDP (Equation 1) using Monte-Carlo sampling. This algorithm is essentially an adaptation of the Sparse Sampling algorithm (Kearns et al., 1999) to BAMDPs. However instead of growing the tree evenly by looking at all actions at each level of the tree, the tree is grown stochastically. Actions are sampled according to their likelihood of being optimal, according to their Q-value distributions (as defined by the Dirichlet posteriors); next states are sampled according to the Dirichlet posterior on the model. This approach requires multiple sampling and solving of MDPs from the Dirichlet distributions to find which action has highest Q-value at each state node in the tree. This can be very time consuming, and so far the approach has only been applied to small MDPs.

Castro and Precup (2007) present a similar approach to Wang et al. However their approach differs on two main points. First, instead of maintaining only the posterior over models, they also maintain Q-value estimates using a standard Q-Learning method. Planning is done by growing a stochastic tree as in Wang et al. (but sampling actions uniformly instead) and solving for the value estimates in that tree using Linear Programming (LP), instead of dynamic programming. In this case, the stochastic tree represents sampled constraints, which the value estimates in the tree must satisfy. The Q-value estimates maintained by Q-Learning are used as value estimates for the fringe nodes (thus as value constraints on the fringe nodes in the LP).

Finally, Poupart et al. (2006) proposed an approximate offline algorithm to solve the BAMDP. Their algorithm, called Beetle, is an extension of the Perseus algorithm (Spaan and Vlassis, 2005) to the BAMDP model. Essentially, at the beginning, hyperstates (s,ϕ) are sampled from random interactions with the BAMDP model. An equivalent continuous POMDP (over the space of states and transition functions) is solved instead of the BAMDP (assuming (s,ϕ) is a belief state in that POMDP). The value function is represented by a set of α-*functions* over the continuous space of transition functions. Each α-function is constructed as a linear combination of basis functions; the sampled hyperstates can serve as the set of basis functions. Dynamic programming is used to incrementally construct the set of α-functions. At each iteration, updates are only performed at the sampled hyperstates, similarly to Perseus (Spaan and Vlassis, 2005) and other Point-Based POMDP algorithms (Pineau et al., 2003).

3. Bayes-Adaptive POMDPs

Despite the sustained interest in model-based BRL, the deployment to real-world applications is limited both by scalability and representation issues. In terms of representation, an important challenge for many practical problems is in handling cases where the state of the system is only partially observable. Our goal here is to show that the model-based BRL framework can be extended to handle partially observable domains. Section 3.1 provides a brief overview of the Partially Observable Markov Decision Process framework. In order to apply Bayesian RL methods in this context, we draw inspiration from the Bayes-Adaptive MDP framework presented in Section 2.3, and propose an extension of this model, called the Bayes-Adaptive POMDP (BAPOMDP). One of the main challenges that arises when considering such an extension is how to update the Dirichlet count parameters when the state is a hidden variable. As will be explained in Section 3.2, this can be achieved by including the Dirichlet parameters in the state space, and maintaining a belief state over these parameters. The BAPOMDP model thus allows an agent to improve its knowledge of an unknown POMDP domain through interaction with the environment, but also allows the decision-making aspect to be contingent on uncertainty over the model parameters. As a result, it is possible to define an action-selection strategy which can directly trade-off between (1) learning the model of the POMDP, (2) identifying the unknown state, and (3) gathering rewards, such as to maximize its future expected return. This model offers an alternative framework for reinforcement learning in POMDPs, compared to previous history-based approaches (McCallum, 1996; Littman et al., 2002).

3.1 Background on POMDPs

While an MDP is able to capture uncertainty on future outcomes, and the BAMDP is able to capture uncertainty over the model parameters, both fail to capture uncertainty that can exist on the current state of the system at a given time step. For example, consider a medical diagnosis problem where the doctor must prescribe the best treatment to an ill patient. In this problem the state (illness) of the patient is unknown, and only its symptoms can be observed. Given the observed symptoms the doctor may believe that some illnesses are more likely, however he may still have some uncertainty about the exact illness of the patient. The doctor must take this uncertainty into account when deciding which treatment is best for the patient. When the uncertainty is high, the best action may be to order additional medical tests in order to get a better diagnosis of the patient's illness.

To address such problems, the Partially Observable Markov Decision Process (POMDP) was proposed as a generalization of the standard MDP model. POMDPs are able to model and reason about the uncertainty on the current state of the system in sequential decision problems (Sondik, 1971).

A POMDP is defined by a finite set of states S, a finite set of actions A, as well as a finite set of observations Z. These observations capture the aspects of the state which can be perceived by the agent. The POMDP is also defined by transition probabilities $\{T^{sas'}\}_{s,s'\in S, a\in A}$, where $T^{sas'} = \Pr(s_{t+1} = s'|s_t = s, a_t = a)$, as well as observation probabilities $\{O^{saz}\}_{s\in S,a\in A,z\in Z}$ where $O^{saz} = \Pr(z_t = z|s_t = s, a_{t-1} = a)$. The reward function, $R : S \times A \to \mathbb{R}$, and discount factor, γ, are as in the MDP model.

Since the state is not directly observed, the agent must rely on the observation and action at each time step to maintain a belief state $b \in \Delta S$, where ΔS is the space of probability distributions over S. The belief state specifies the probability of being in each state given the history of action and observation experienced so far, starting from an initial belief b_0. It can be updated at each time step using the following Bayes rule:

$$b_{t+1}(s') = \frac{O^{s'a_tz_{t+1}} \sum_{s\in S} T^{sa_ts'} b_t(s)}{\sum_{s''\in s} O^{s''a_tz_{t+1}} \sum_{s\in S} T^{sa_ts''} b_t(s)}.$$

A policy $\pi : \Delta S \to A$ indicates how the agent should select actions as a function of the current belief. Solving a POMDP involves finding the optimal policy π^* that maximizes the expected discounted return over the infinite horizon. The return obtained by following π^* from a belief b is defined by Bellman's equation:

$$V^*(b) = \max_{a\in A} \left[\sum_{s\in S} b(s)R(s,a) + \gamma \sum_{z\in Z} \Pr(z|b,a) V^*(\tau(b,a,z)) \right],$$

where $\tau(b,a,z)$ is the new belief after performing action a and observation z,and $\gamma \in [0,1)$ is the discount factor.

A key result by Smallwood and Sondik (1973) shows that the optimal value function for a finite-horizon POMDP is piecewise-linear and convex. It means that the value function V_t at any finite horizon t can be represented by a finite set of $|S|$-dimensional hyperplanes: $\Gamma_t = \{\alpha_0, \alpha_1, \ldots, \alpha_m\}$. These hyperplanes are often called α-vectors. Each defines a linear value function over the belief state space, associated with some action, $a \in A$. The value of a belief state is the maximum value returned by one of the α-vectors for this belief state:

$$V_t(b) = \max_{\alpha\in\Gamma_t} \sum_{s\in S} \alpha(s) b(s).$$

The best action is the one associated with the α-vector that returns the best value.

The Enumeration algorithm by Sondik (1971) shows how the finite set of α-vectors, Γ_t, can be built incrementally via dynamic programming. The idea is that any t-step contingency plan can be expressed by an immediate action and a mapping associating a (t-1)-step contingency plan to every observation the agent could get after this immediate action. The value of the 1-step plans corresponds directly to the immediate rewards:

$$\begin{aligned} \Gamma_1^a &= \{\alpha^a | \alpha^a(s) = R(s,a)\}, \\ \Gamma_1 &= \textstyle\bigcup_{a\in A} \Gamma_1^a. \end{aligned}$$

Then to build the α-vectors at time t, we consider all possible immediate actions the agent could take, every observation that could follow, and every combination of (t-1)-step plans to pursue subsequently:

$$\begin{aligned} \Gamma_t^{a,z} &= \{\alpha^{a,z} | \alpha^{a,z}(s) = \gamma \textstyle\sum_{s'\in S} T^{sas'} O^{s'az} \alpha'(s'), \alpha' \in \Gamma_{t-1}\}, \\ \Gamma_t^a &= \Gamma_1^a \oplus \Gamma_t^{a,z_1} \oplus \Gamma_t^{a,z_2} \oplus \ldots \oplus \Gamma_t^{a,z_{|Z|}}, \\ \Gamma_t &= \textstyle\bigcup_{a\in A} \Gamma_t^a, \end{aligned}$$

where $\oplus$ is the cross-sum operator.[7]

Exactly solving the POMDP is usually intractable, except on small domains with only a few states, actions and observations (Kaelbling et al., 1998). Various approximate algorithms, both offline (Pineau et al., 2003; Spaan and Vlassis, 2005; Smith and Simmons, 2004) and online (Paquet

7. Let A and B be sets of vectors, then $A \oplus B = \{a + b | a \in A, b \in B\}$.

et al., 2005; Ross et al., 2008c), have been proposed to tackle increasingly large domains. However, all these methods require full knowledge of the POMDP model, which is a strong assumption in practice. Some approaches do not require knowledge of the model, as in Baxter and Bartlett (2001), but these approaches generally require some knowledge of a good (and preferably compact) policy class, as well as needing substantial amounts of data.

3.2 Bayesian Learning of a POMDP model

Before we introduce the full BAPOMDP model for sequential decision-making under model uncertainty in a POMDP, we first show how a POMDP model can be learned via a Bayesian approach.

Consider an agent in a POMDP (S,A,Z,T,O,R,γ), where the transition function T and observation function O are the only unknown components of the POMDP model. Let $z_t = (z_1, z_2, \ldots, z_t)$ be the history of observations of the agent up to time t. Recall also that we denote $s_t = (s_0, s_1, \ldots, s_t)$ and $a_{t-1} = (a_0, a_1, \ldots, a_{t-1})$ the history of visited states and actions respectively. The Bayesian approach to learning T and O involves starting with a prior distribution over T and O, and maintaining the posterior distribution over T and O after observing the history (a_{t-1}, z_t). Since the current state s_t of the agent at time t is unknown in the POMDP, we consider a joint posterior $g(s_t, T, O|a_{t-1}, z_t)$ over s_t, T, and O. By the laws of probability and Markovian assumption of the POMDP, we have:

$$\begin{array}{rcl} g(s_t,T,O|a_{t-1},z_t) & \propto & \Pr(z_t,s_t|T,O,a_{t-1})g(T,O,a_{t-1}) \\ & \propto & \sum_{s_{t-1}\in S^t} \Pr(z_t,s_t|T,O,a_{t-1})g(T,O) \\ & \propto & \sum_{s_{t-1}\in S^t} g(s_0,T,O)\prod_{i=1}^{t} T^{s_{i-1}a_{i-1}s_i} O^{s_i a_{i-1} z_i} \\ & \propto & \sum_{s_{t-1}\in S^t} g(s_0,T,O)\left[\prod_{s,a,s'} (T^{sas'})^{N^a_{ss'}(s_t,a_{t-1})}\right] \times \\ & & \left[\prod_{s,a,z} (O^{saz})^{N^a_{sz}(s_t,a_{t-1},z_t)}\right], \end{array}$$

where $g(s_0,T,O)$ is the joint prior over the initial state s_0, transition function T, and observation function O; $N^a_{ss'}(s_t,a_{t-1}) = \sum_{i=0}^{t-1} I_{\{(s,a,s')\}}(s_i,a_i,s_{i+1})$ is the number of times the transition (s,a,s') appears in the history of state-action (s_t,a_{t-1}); and $N^a_{sz}(s_t,a_{t-1},z_t) = \sum_{i=1}^{t} I_{\{(s,a,z)\}}(s_i,a_{i-1},z_i)$ is the number of times the observation (s,a,z) appears in the history of state-action-observations (s_t,a_{t-1},z_t). We use proportionality rather than equality in the expressions above because we have not included the normalization constant.

Under the assumption that the prior $g(s_0,T,O)$ is defined by a product of independent priors of the form:

$$g(s_0,T,O) = g(s_0)\prod_{s,a} g_{sa}(T^{sa\cdot})g_{sa}(O^{sa\cdot}),$$

and that $g_{sa}(T^{sa\cdot})$ and $g_{sa}(O^{sa\cdot})$ are Dirichlet priors defined $\forall s,a$, then we observe that the posterior is a mixture of joint Dirichlets, where each joint Dirichlet component is parameterized by the counts corresponding to one specific possible state sequence:

$$\begin{array}{rcl} g(s_t,T,O|a_{t-1},z_t) & \propto & \sum_{s_{t-1}\in S^t} g(s_0)c(s_t,a_{t-1},z_t)\times \\ & & \left[\prod_{s,a,s'} (T^{sas'})^{N^a_{ss'}(s_t,a_{t-1})+\phi^a_{ss'}-1}\right] \times \\ & & \left[\prod_{s,a,z} (O^{saz})^{N^a_{sz}(s_t,a_{t-1},z_t)+\psi^a_{sz}-1}\right]. \end{array} \qquad (2)$$

Here, $\phi^a_{s\cdot}$ are the prior Dirichlet count parameters for $g_{sa}(T^{sa\cdot})$, $\psi^a_{s\cdot}$ are the prior Dirichlet count parameters for $g_{sa}(O^{sa\cdot})$, and $c(s_t,a_{t-1},z_t)$ is a constant which corresponds to the normalization constant of the joint Dirichlet component for the state-action-observation history (s_t, a_{t-1}, z_t). Intuitively, Bayes' rule tells us that given a particular state sequence, it is possible to compute the proper posterior counts of the Dirichlets, but since the state sequence that actually occurred is unknown, all state sequences (and their corresponding Dirichlet posteriors) must be considered, with some weight proportional to the likelihood of each state sequence.

In order to update the posterior online, each time the agent performs an action and gets an observation, it is more useful to express the posterior in recursive form:

$$g(s_t, T, O|a_{t-1}, z_t) \propto \sum_{s_{t-1} \in S} T^{s_{t-1}a_{t-1}s_t} O^{s_t a_{t-1} z_t} g(s_{t-1}, T, O|a_{t-2}, z_{t-1}).$$

Hence if $g(s_{t-1}, T, O|a_{t-2}, z_{t-1}) = \sum_{(\phi,\psi)\in \mathcal{C}(s_{t-1})} w(s_{t-1}, \phi, \psi) f(T, O|\phi, \psi)$ is a mixture of $|\mathcal{C}(s_{t-1})|$ joint Dirichlet components, where each component (ϕ, ψ) is parameterized by the set of transition counts $\phi = \{\phi^a_{ss'} \in \mathbb{N} | s, s' \in S, a \in A\}$ and the set observation counts $\psi = \{\psi^a_{sz} \in \mathbb{N} | s \in S, a \in A, z \in Z\}$, then $g(s_t, T, O|a_{t-1}, z_t)$ is a mixture of $\prod_{s\in S} |\mathcal{C}(s)|$ joint Dirichlet components, given by:

$$\begin{array}{rcl} g(s_t, T, O|a_{t-1}, z_t) & \propto & \sum_{s_{t-1}\in S} \sum_{(\phi,\psi)\in \mathcal{C}(s_{t-1})} w(s_{t-1}, \phi, \psi) c(s_{t-1}, a_{t-1}, s_t, z_{t-1}, \phi, \psi) \\ & & f(T, O| \mathcal{U}(\phi, s_{t-1}, a_{t-1}, s_t), \mathcal{U}(\psi, s_t, a_{t-1}, z_t)), \end{array}$$

where $\mathcal{U}(\phi, s, a, s')$ increments the count $\phi^a_{ss'}$ by one in the set of counts ϕ, $\mathcal{U}(\psi, s, a, z)$ increments the count ψ^a_{sz} by one in the set of counts ψ, and $c(s_{t-1}, a_{t-1}, s_t, z_{t-1}, \phi, \psi)$ is a constant corresponding to the ratio of the normalization constants of the joint Dirichlet component (ϕ, ψ) before and after the update with $(s_{t-1}, a_{t-1}, s_t, z_{t-1})$. This last equation gives us an online algorithm to maintain the posterior over (s, T, O), and thus allows the agent to learn about the unknown T and O via Bayesian inference.

Now that we have a simple method of maintaining the uncertainty over both the state and model parameters, we would like to address the more interesting question of how to optimally behave in the environment under such uncertainty, in order to maximize future expected return. Here we proceed similarly to the Bayes-Adaptive MDP framework defined in Section 2.3.

First, notice that the posterior $g(s_t, T, O|a_{t-1}, z_t)$ can be seen as a probability distribution (belief) b over tuples (s, ϕ, ψ), where each tuple represents a particular joint Dirichlet component parameterized by the counts (ϕ, ψ) for a state sequence ending in state s (i.e., the current state is s), and the probabilities in the belief b correspond to the mixture weights. Now we would like to find a policy π for the agent which maps such beliefs over (s, ϕ, ψ) to actions $a \in A$. This suggests that the sequential decision problem of optimally behaving under state and model uncertainty can be modeled as a POMDP over hyperstates of the form (s, ϕ, ψ).

Consider a new POMDP $(S', A, Z, P', R', \gamma)$, where the set of states (hyperstates) is formally defined as $S' = S \times \mathcal{T} \times \mathcal{O}$, with $\mathcal{T} = \{\phi \in \mathbb{N}^{|S|^2|A|} | \forall (s, a) \in S \times A,\ \sum_{s'\in S} \phi^a_{ss'} > 0\}$ and $\mathcal{O} = \{\psi \in \mathbb{N}^{|S||A||Z|} | \forall (s, a) \in S \times A,\ \sum_{z\in Z} \psi^a_{sz} > 0\}$. As in the definition of the BAMDP, the constraints on the count parameters ϕ and ψ are only to ensure that the transition-observation probabilities, as defined below, are well defined. The action and observation sets are the same as in the original POMDP. The rewards depend only on the state $s \in S$ and action $a \in A$ (but not the counts ϕ and ψ), thus we have $R'(s, \phi, \psi, a) = R(s, a)$. The transition and observations probabilities in the BAPOMDP are defined by a joint transition-observation function $P' : S' \times A \times S' \times Z \to [0, 1]$, such that $P'(s,\phi,\psi,a,s',\phi',\psi',z) = \Pr(s',\phi',\psi',z|s,\phi,\psi,a)$. This joint probability can be factorized by using the laws of probability and standard independence assumptions:

$$\begin{aligned}
&\Pr(s',\phi',\psi',z|s,\phi,\psi,a)\\
= \ &\Pr(s'|s,\phi,\psi,a)\Pr(z|s,\phi,\psi,a,s')\Pr(\phi'|s,\phi,\psi,a,s',z)\Pr(\psi'|s,\phi,\psi,a,s',\phi',z)\\
= \ &\Pr(s'|s,a,\phi)\Pr(z|a,s',\psi)\Pr(\phi'|\phi,s,a,s')\Pr(\psi'|\psi,a,s',z).
\end{aligned}$$

As in the Bayes-Adaptive MDP case, $\Pr(s'|s,a,\phi)$ corresponds to the expectation of $\Pr(s'|s,a)$ under the joint Dirichlet posterior defined by ϕ, and $\Pr(\phi'|\phi,s,a,s')$ is either 0 or 1, depending on whether ϕ' corresponds to the posterior after observing transition (s,a,s') from prior ϕ. Hence $\Pr(s'|s,a,\phi) = \frac{\phi^a_{ss'}}{\sum_{s''\in S}\phi^a_{ss''}}$, and $\Pr(\phi'|\phi,s,a,s') = I_{\{\phi'\}}(\mathcal{U}(\phi,s,a,s'))$. Similarly, $\Pr(z|a,s',\psi) = \int O^{s'az}f(O|\psi)dO$, which is the expectation of the Dirichlet posterior for $\Pr(z|s',a)$, and $\Pr(\psi'|\psi,a,s',z)$, is either 0 or 1, depending on whether ψ' corresponds to the posterior after observing observation (s',a,z) from prior ψ. Thus $\Pr(z|a,s',\psi) = \frac{\psi^a_{s'z}}{\sum_{z'\in Z}\psi^a_{s'z'}}$, and $\Pr(\psi'|\psi,a,s',z) = I_{\{\psi'\}}(\mathcal{U}(\psi,s',a,z))$. To simplify notation, we denote $T^{sas'}_{\phi} = \frac{\phi^a_{ss'}}{\sum_{s''\in S}\phi^a_{ss''}}$ and $O^{s'az}_{\psi} = \frac{\psi^a_{s'z}}{\sum_{z'\in Z}\psi^a_{s'z'}}$. It follows that the joint transition-observation probabilities in the BAPOMDP are defined by:

$$\Pr(s',\phi',\psi',z|s,\phi,\psi,a) = T^{sas'}_{\phi}O^{s'az}_{\psi}I_{\{\phi'\}}(\mathcal{U}(\phi,s,a,s'))I_{\{\psi'\}}(\mathcal{U}(\psi,s',a,z)).$$

Hence, the BAPOMDP defined by the POMDP (S',A,Z,P',R',γ) has a known model and characterizes the problem of optimal sequential decision-making in the original POMDP (S,A,Z,T,O,R,γ) with uncertainty on the transition T and observation functions O described by Dirichlet distributions.

An alternative interpretation of the BAPOMDP is as follows: given the unknown state sequence that occurred since the beginning, one can compute exactly the posterior counts ϕ and ψ. Thus there exists a unique (ϕ,ψ) reflecting the correct posterior counts according to the state sequence that occurred, but these correct posterior counts are only partially observable through the observations $z\in Z$ obtained by the agent. Thus (ϕ,ψ) can simply be thought of as other hidden state variables that the agent tracks via the belief state, based on its observations. The BAPOMDP formulates the decision problem of optimal sequential decision-making under partial observability of both the state $s\in S$, and posterior counts (ϕ,ψ).

The belief state in the BAPOMDP corresponds exactly to the posterior defined in the previous section (Equation 2). By maintaining this belief, the agent maintains its uncertainty on the POMDP model and learns about the unknown transition and observations functions. Initially, if ϕ_0 and ψ_0 represent the prior Dirichlet count parameters (i.e., the agent's prior knowledge of T and O), and b_0 the initial state distribution of the unknown POMDP, then the initial belief b'_0 of the BAPOMDP is defined as $b'_0(s,\phi,\psi) = b_0(s)I_{\{\phi_0\}}(\phi)I_{\{\psi_0\}}(\psi)$. Since the BAPOMDP is just a POMDP with an infinite number of states, the belief update and value function equations presented in Section 3.1 can be applied directly to the BAPOMDP model. However, since there is an infinite number of hyperstates, these calculations can be performed exactly in practice only if the number of possible hyperstates in the belief is finite. The following theorem shows that this is the case at any finite time t:

Theorem 1 *Let (S',A,Z,P',R',γ) be a BAPOMDP constructed from the POMDP (S,A,Z,T,O,R,γ). If S is finite, then at any time t, the set $S'_{b'_t} = \{\sigma\in S'|b'_t(\sigma)>0\}$ has size $|S'_{b'_t}| \leq |S|^{t+1}$.*

```
function $\tau(b,a,z)$
Initialize $b'$ as a 0 vector.
for all $(s,\phi,\psi) \in S'_b$ do
   for all $s' \in S$ do
      $\phi' \leftarrow \mathcal{U}(\phi,s,a,s')$
      $\psi' \leftarrow \mathcal{U}(\psi,s',a,z)$
      $b'(s',\phi',\psi') \leftarrow b'(s',\phi',\psi') + b(s,\phi,\psi)T_\phi^{sas'}O_\psi^{s'az}$
   end for
end for
return normalized $b'$
```

Algorithm 1: Exact Belief Update in BAPOMDP.

Proof Proof available in Appendix A. ■

The proof of Theorem 1 suggests that it is sufficient to iterate over S and $S'_{b'_{t-1}}$ in order to compute the belief state b'_t when an action and observation are taken in the environment. Hence, we can update the belief state in closed-form, as outlined in Algorithm 1. Of course this algorithm is not tractable for large domains with long action-observation sequences. Section 5 provides a number of approximate tracking algorithms which tackle this problem.

3.3 Exact Solution for the BAPOMDP in Finite Horizons

The value function of a BAPOMDP for finite horizons can be represented by a finite set Γ of functions $\alpha : S' \to \mathbb{R}$, as in standard POMDPs. This is shown formally in the following theorem:

Theorem 2 *For any horizon t, there exists a finite set Γ_t of functions $S' \to \mathbb{R}$, such that $V_t^*(b) = \max_{\alpha\in\Gamma_t} \sum_{\sigma\in S'} \alpha(\sigma)b(\sigma)$.*

Proof Proof available in the appendix. ■

The proof of this theorem shows that as in any POMDP, an exact solution of the BAPOMDP can be computed using dynamic programming, by incrementally constructing the set of α-functions that defines the value function as follows:

$$\begin{array}{rcl}
\Gamma_1^a & = & \{\alpha^a | \alpha^a(s,\phi,\psi) = R(s,a)\}, \\
\Gamma_t^{a,z} & = & \{\alpha^{a,z} | \alpha^{a,z}(s,\phi,\psi) = \gamma \sum_{s'\in S} T_\phi^{sas'} O_\psi^{s'az} \alpha'(s', \mathcal{U}(\phi,s,a,s'), \mathcal{U}(\psi,s',a,z)), \\
 & & \hfill \alpha' \in \Gamma_{t-1}\}, \\
\Gamma_t^a & = & \Gamma_1^a \oplus \Gamma_t^{a,z_1} \oplus \Gamma_t^{a,z_2} \oplus \cdots \oplus \Gamma_t^{a,z_{|Z|}}, \quad \text{(where } \oplus \text{ is the cross sum operator)}, \\
\Gamma_t & = & \bigcup_{a\in A} \Gamma_t^a.
\end{array}$$

However in practice, it will be impossible to compute $\alpha_i^{a,z}(s,\phi,\psi)$ for all $(s,\phi,\psi) \in S'$, unless a particular finite parametric form for the α-functions is used. Poupart and Vlassis (2008) showed that these α-functions can be represented as a linear combination of product of Dirichlets and can thus be represented by a finite number of parameters. Further discussion of their work is included in Section 7. We present an alternate approach in Section 5.

4. Approximating the BAPOMDP by a Finite POMDP

Solving the BAPOMDP exactly for all belief states is often impossible due to the dimensionality of the state space, in particular because the count vectors can grow unbounded. The first proposed method to address this problem is to reduce this infinite state space to a finite state space, while preserving the value function of the BAPOMDP to arbitrary precision. This allows us to compute an ε-optimal value function over the resulting finite-dimensional belief space using standard finite POMDP solvers. This can then be used to obtain an ε-optimal policy to the BAPOMDP.

The main intuition behind the compression of the state space presented here is that, as the Dirichlet counts grow larger and larger, the transition and observation probabilities defined by these counts do not change much when the counts are incremented by one. Hence, there should exist a point where if we simply stop incrementing the counts, the value function of that approximate BAPOMDP (where the counts are bounded) approximates the value function of the BAPOMDP within some $\varepsilon > 0$. If we can bound above the counts in such a way, this ensures that the state space will be finite.

In order to find such a bound on the counts, we begin by deriving an upper bound on the value difference between two hyperstates that differ only by their model estimates ϕ and ψ. This bound uses the following definitions: given $\phi, \phi' \in \mathcal{T}$, and $\psi, \psi' \in \mathcal{O}$, define $D_S^{sa}(\phi, \phi') = \sum_{s' \in S} \left| T_{\phi}^{sas'} - T_{\phi'}^{sas'} \right|$, $D_Z^{sa}(\psi, \psi') = \sum_{z \in Z} \left| O_{\psi}^{saz} - O_{\psi'}^{saz} \right|$, $\mathcal{N}_{\phi}^{sa} = \sum_{s' \in S} \phi_{ss'}^{a}$, and $\mathcal{N}_{\psi}^{sa} = \sum_{z \in Z} \psi_{sz}^{a}$.

Theorem 3 *Given any $\phi, \phi' \in \mathcal{T}$, $\psi, \psi' \in \mathcal{O}$, and $\gamma \in (0,1)$, then for all t:*

$$\sup_{\alpha_t \in \Gamma_t, s \in S} |\alpha_t(s,\phi,\psi) - \alpha_t(s,\phi',\psi')| \leq \frac{2\gamma||R||_\infty}{(1-\gamma)^2} \sup_{s,s' \in S, a \in A} \Bigg[D_S^{sa}(\phi,\phi') + D_Z^{s'a}(\psi,\psi') + \frac{4}{\ln(\gamma^{-e})} \left(\frac{\sum_{s'' \in S} \left|\phi_{ss''}^{a} - \phi'^{a}_{ss''}\right|}{(\mathcal{N}_{\phi}^{sa}+1)(\mathcal{N}_{\phi'}^{sa}+1)} + \frac{\sum_{z \in Z} \left|\psi_{s'z}^{a} - \psi'^{a}_{s'z}\right|}{(\mathcal{N}_{\psi}^{s'a}+1)(\mathcal{N}_{\psi'}^{s'a}+1)} \right) \Bigg]$$

Proof Proof available in the appendix. ■

We now use this bound on the α-vector values to approximate the space of Dirichlet parameters within a finite subspace. We use the following definitions: given any $\varepsilon > 0$, define $\varepsilon' = \frac{\varepsilon(1-\gamma)^2}{8\gamma||R||_\infty}$, $\varepsilon'' = \frac{\varepsilon(1-\gamma)^2 \ln(\gamma^{-e})}{32\gamma||R||_\infty}$, $N_S^{\varepsilon} = \max\left(\frac{|S|(1+\varepsilon')}{\varepsilon'}, \frac{1}{\varepsilon''} - 1\right)$ and $N_Z^{\varepsilon} = \max\left(\frac{|Z|(1+\varepsilon')}{\varepsilon'}, \frac{1}{\varepsilon''} - 1\right)$.

Theorem 4 *Given any $\varepsilon > 0$ and $(s,\phi,\psi) \in S'$ such that $\exists a \in A, \exists s' \in S$, $\mathcal{N}_{\phi}^{s'a} > N_S^{\varepsilon}$ or $\mathcal{N}_{\psi}^{s'a} > N_Z^{\varepsilon}$, then $\exists (s,\phi',\psi') \in S'$ such that $\forall a \in A, \forall s' \in S$, $\mathcal{N}_{\phi'}^{s'a} \leq N_S^{\varepsilon}$, $\mathcal{N}_{\psi'}^{s'a} \leq N_Z^{\varepsilon}$ and $|\alpha_t(s,\phi,\psi) - \alpha_t(s,\phi',\psi')| < \varepsilon$ holds for all t and $\alpha_t \in \Gamma_t$.*

Proof Proof available in the appendix. ■

Theorem 4 suggests that if we want a precision of ε on the value function, we just need to restrict the space of Dirichlet parameters to count vectors $\phi \in \tilde{\mathcal{T}}_{\varepsilon} = \{\phi \in \mathbb{N}^{|S|^2|A|} | \forall a \in A, s \in S, 0 < \mathcal{N}_{\phi}^{sa} \leq N_S^{\varepsilon}\}$, and $\psi \in \tilde{\mathcal{O}}_{\varepsilon} = \{\psi \in \mathbb{N}^{|S||A||Z|} | \forall a \in A, s \in S, 0 < \mathcal{N}_{\psi}^{sa} \leq N_Z^{\varepsilon}\}$. Since $\tilde{\mathcal{T}}_{\varepsilon}$ and $\tilde{\mathcal{O}}_{\varepsilon}$ are finite, we can define a finite approximate BAPOMDP as the tuple $(\tilde{S}_{\varepsilon}, A, Z, \tilde{P}_{\varepsilon}, \tilde{R}_{\varepsilon}, \gamma)$, where $\tilde{S}_{\varepsilon} = S \times \tilde{\mathcal{T}}_{\varepsilon} \times \tilde{\mathcal{O}}_{\varepsilon}$ is the finite state space, and $\tilde{P}_{\varepsilon}$ is the joint transition-observation function over this finite state space.

To define this function, we need to ensure that whenever the count vectors are incremented, they stay within the finite space. To achieve this, we define a projection operator $\mathcal{P}_\varepsilon : S' \to \tilde{S}_\varepsilon$ that simply projects every state in S' to their closest state in $\tilde{S}_\varepsilon$.

Definition 1 *Let $d : S' \times S' \to \mathbb{R}$ be defined such that:*

$$d(s,\phi,\psi,s',\phi',\psi') = \begin{cases} \frac{2\gamma||R||_\infty}{(1-\gamma)^2} \sup\limits_{s,s' \in S, a \in A} \Big[D_S^{sa}(\phi,\phi') + D_Z^{s'a}(\psi,\psi') & \\ \quad + \frac{4}{\ln(\gamma^{-e})} \left(\frac{\sum_{s'' \in S} |\phi^a_{ss''} - \phi'^a_{ss''}|}{(\mathcal{N}^{as}_\phi + 1)(\mathcal{N}^{as}_{\phi'}+1)} + \frac{\sum_{z \in Z} |\psi^a_{s'z} - \psi'^a_{s'z}|}{(\mathcal{N}^{as'}_\psi + 1)(\mathcal{N}^{as'}_{\psi'}+1)} \right) \Big], & \textit{if } s = s' \\ \frac{8\gamma||R||_\infty}{(1-\gamma)^2} \left(1 + \frac{4}{\ln(\gamma^{-e})}\right) + \frac{2||R||_\infty}{(1-\gamma)}, & \textit{otherwise}. \end{cases}$$

Definition 2 *Let $\mathcal{P}_\varepsilon : S' \to \tilde{S}_\varepsilon$ be defined as $\mathcal{P}_\varepsilon(s) = \operatorname*{arg\,min}\limits_{s' \in \tilde{S}_\varepsilon} d(s,s')$.*

The function d uses the bound defined in Theorem 3 as a distance between states that only differ in their ϕ and ψ vectors, and uses an upper bound on that value when the states differ. Thus $\mathcal{P}_\varepsilon$ always maps states $(s,\phi,\psi) \in S'$ to some state $(s,\phi',\psi') \in \tilde{S}_\varepsilon$. Note that if $\sigma \in \tilde{S}_\varepsilon$, then $\mathcal{P}_\varepsilon(\sigma) = \sigma$. Using $\mathcal{P}_\varepsilon$, the joint transition-observation function can then be defined as follows:

$$\tilde{P}_\varepsilon(s,\phi,\psi,a,s',\phi',\psi',z) = T_\phi^{sas'} O_\psi^{s'az} I_{\{(s',\phi',\psi')\}}(\mathcal{P}_\varepsilon(s', \mathcal{U}(\phi,s,a,s'), \mathcal{U}(\psi,s',a,z))).$$

This definition is the same as the one in the BAPOMDP, except that now an extra projection is added to make sure that the incremented count vectors stay in $\tilde{S}_\varepsilon$. Finally, the reward function $\tilde{R}_\varepsilon : \tilde{S}_\varepsilon \times A \to \mathbb{R}$ is defined as $\tilde{R}_\varepsilon((s,\phi,\psi),a) = R(s,a)$. This defines a proper finite POMDP $(\tilde{S}_\varepsilon, A, Z, \tilde{P}_\varepsilon, \tilde{R}_\varepsilon, \gamma)$, which can be used to approximate the original BAPOMDP model.

Next, we are interested in characterizing the quality of solutions that can be obtained with this finite model. Theorem 5 bounds the value difference between α-vectors computed with this finite model and the α-vector computed with the original model.

Theorem 5 *Given any $\varepsilon > 0$, $(s,\phi,\psi) \in S'$ and $\alpha_t \in \Gamma_t$ computed from the infinite BAPOMDP. Let $\tilde{\alpha}_t$ be the α-vector representing the same conditional plan as α_t but computed with the finite POMDP $(\tilde{S}_\varepsilon, A, Z, \tilde{T}_\varepsilon, \tilde{O}_\varepsilon, \tilde{R}_\varepsilon, \gamma)$, then $|\tilde{\alpha}_t(\mathcal{P}_\varepsilon(s,\phi,\psi)) - \alpha_t(s,\phi,\psi)| < \frac{\varepsilon}{1-\gamma}$.*

Proof Proof available in the appendix. To summarize, it solves a recurrence over the 1-step approximation in Theorem 4. ■

Such bounded approximation over the α-functions of the BAPOMDP implies that the optimal policy obtained from the finite POMDP approximation has an expected value close to the value of the optimal policy of the full (non-projected) BAPOMDP:

Theorem 6 *Given any $\varepsilon > 0$, and any horizon t, let $\tilde{\pi}_t$ be the optimal t-step policy computed from the finite POMDP $(\tilde{S}_\varepsilon, A, Z, \tilde{T}_\varepsilon, \tilde{O}_\varepsilon, \tilde{R}_\varepsilon, \gamma)$, then for any initial belief b the value of executing policy $\tilde{\pi}_t$ in the BAPOMDP $V_{\tilde{\pi}_t}(b) \geq V^*(b) - 2\frac{\varepsilon}{1-\gamma}$.*

Proof Proof available in the appendix, and follows from Theorem 5. ■

We note that the last two theorems hold even if we construct the finite POMDP with the following approximate state projection $\tilde{\mathcal{P}}_\varepsilon$, which is more easy to use in practice:

Definition 3 *Let $\tilde{\mathcal{P}}_\varepsilon : S' \to \tilde{S}_\varepsilon$ be defined as $\tilde{\mathcal{P}}_\varepsilon(s,\phi,\psi) = (s,\hat{\phi},\hat{\psi})$ where:*

$$\hat{\phi}^a_{s',s''} = \begin{cases} \phi^a_{s',s''} & \text{if } \mathcal{N}^{s'a}_\phi \leq N^\varepsilon_S \\ \lfloor N^\varepsilon_S T^{s'as''}_\phi \rfloor & \text{if } \mathcal{N}^{s'a}_\phi > N^\varepsilon_S \end{cases}$$

$$\hat{\psi}^a_{s',z} = \begin{cases} \psi^a_{s',z} & \text{if } \mathcal{N}^{s'a}_\psi \leq N^\varepsilon_Z \\ \lfloor N^\varepsilon_Z O^{s'az}_\psi \rfloor & \text{if } \mathcal{N}^{s'a}_\psi > N^\varepsilon_Z \end{cases}$$

This follows from the proof of Theorem 5, which only relies on such a projection, and not on the projection that minimizes d (as done by $\mathcal{P}_\varepsilon$).

Given that the state space is now finite, offline solution methods from the literature on finite POMDPs could potentially be applied to obtain an ε-optimal policy to the BAPOMDP. Note however that even though the state space is finite, it will generally be very large for small ε, such that the resulting finite POMDP may still be intractable to solve offline, even for small domains.

An alternative approach is to solve the BAPOMDP online, by focusing on finding the best immediate action to perform in the current belief of the agent, as in online POMDP solution methods (Ross et al., 2008c). In fact, provided we have an efficient way of updating the belief, online POMDP solvers can be applied directly in the infinite BAPOMDP without requiring a finite approximation of the state space. In practice, maintaining the exact belief in the BAPOMDP quickly becomes intractable (exponential in the history length, as shown in Theorem 1). The next section proposes several practical efficient approximations for both belief updating and online planning in the BAPOMDP.

5. Towards a Tractable Approach to BAPOMDPs

Having fully specified the BAPOMDP framework and its finite approximation, we now turn our attention to the problem of scalable belief tracking and planning in this framework. This section is intentionally briefer, as many of the results in the probabilistic reasoning literature can be applied to the BAPOMDP framework. We outline those methods which have proven effective in our empirical evaluations.

5.1 Approximate Belief Monitoring

As shown in Theorem 1, the number of states with non-zero probability grows exponentially in the planning horizon, thus exact belief monitoring can quickly become intractable. This problem is not unique to the Bayes-optimal POMDP framework, and was observed in the context of Bayes nets with missing data (Heckerman et al., 1995). We now discuss different particle-based approximations that allow polynomial-time belief tracking.

Monte-Carlo Filtering: Monte-Carlo filtering algorithms have been widely used for sequential state estimation (Doucet et al., 2001). Given a prior belief b, followed by action a and observation z, the new belief b' is obtained by first sampling K states from the distribution b, then for each sampled s a new state s' is sampled from $T^{sa\cdot}$. Finally, the probability $O^{s'az}$ is added to $b'(s')$ and the belief b' is re-normalized. This will capture at most K states with non-zero probabilities. In the context of BAPOMDPs, we use a slight variation of this method, where (s,ϕ,ψ) are first sampled from b, and then a next state $s' \in S$ is sampled from the normalized distribution $T^{sa\cdot}_\phi O^{\cdot az}_\psi$. The probability $1/K$ is added directly to $b'(s', \mathcal{U}(\phi,s,a,s'), \mathcal{U}(\psi,s,a,s'))$.

```
**function** $WD(b,a,z,K)$
$b' \leftarrow \tau(b,a,z)$
Initialize $b''$ as a 0 vector.
$(s,\phi,\psi) \leftarrow \text{argmax}_{(s',\phi',\psi')\in S'_{b'}} b'(s',\phi',\psi')$
$b''(s,\phi,\psi) \leftarrow b'(s,\phi,\psi)$
**for** $i=2$ to $K$ **do**
    $(s,\phi,\psi) \leftarrow \text{argmax}_{(s',\phi',\psi')\in S'_{b'}} b'(s',\phi',\psi') \min_{(s'',\phi'',\psi'')\in S'_{b''}} d(s',\phi',\psi',s'',\phi'',\psi'')$
    $b''(s,\phi,\psi) \leftarrow b'(s,\phi,\psi)$
**end for**
**return** normalized $b''$
```

Algorithm 2: Weighted Distance Belief Update in BAPOMDP.

Most Probable: Another possibility is to perform the exact belief update at a given time step, but then only keep the K most probable states in the new belief b', and re-normalize b'. This minimizes the L_1 distance between the exact belief b' and the approximate belief maintained with K particles.[8] While keeping only the K most probable particles biases the belief of the agent, this can still be a good approach in practice, as minimizing the L_1 distance bounds the difference between the values of the exact and approximate belief: that is, $|V^*(b)-V^*(b')| \leq \frac{||R||_\infty}{1-\gamma}||b-b'||_1$.

Weighted Distance Minimization: Finally, we consider an belief approximation technique which aims to directly minimize the difference in value function between the approximate and exact belief state by exploiting the upper bound on the value difference defined in Section 4. Hence, in order to keep the K particles which best approximate the exact belief's value, an exact belief update is performed and then the K particles which minimize the weighted sum of distance measures, where distance is defined as in Definition 1, are kept to approximate the exact belief. This procedure is described in Algorithm 2.

5.2 Online Planning

As discussed above, standard offline or online POMDP solvers can be used to optimize the choice of action in the BAPOMDP model. Online POMDP solvers (Paquet et al., 2005; Ross et al., 2008c) have a clear advantage over offline finite POMDP solvers (Pineau et al., 2003; Spaan and Vlassis, 2005; Smith and Simmons, 2004) in the context of the BAPOMDP as they can be applied directly in infinite POMDPs, provided we have an efficient way to compute beliefs. Hence online POMDP solvers can be applied directly to solve the BAPOMDP without using the finite POMDP representation presented in Section 4. Another advantage of the online approach is that by planning from the current belief, for any finite planning horizon t, one can compute exactly the optimal value function, as only a finite number of beliefs can be reached over that finite planning horizon. While the number of reachable beliefs is exponential in the horizon, often only a small subset is most relevant for obtaining a good estimate of the value function. Recent online algorithms (Ross et al., 2008c) have leveraged this by developing several heuristics for focusing computations on only the most important reachable beliefs to obtain a good estimate quickly.

Since our focus is not on developing new online planning algorithms, we hereby simply present a simple online lookahead search algorithm that performs dynamic programming over all the beliefs

8. The L_1 distance between two beliefs b and b', denoted $||b-b'||_1$, is defined as $\sum_{\sigma\in S'} |b(\sigma)-b'(\sigma)|$.

reachable within some fixed finite planning horizon from the current belief. The action with highest return over that finite horizon is executed and then planning is conducted again on the next belief.

To further limit the complexity of the online planning algorithm, we used the approximate belief monitoring methods detailed above. The detailed procedure is provided in Algorithm 3. This algorithm takes as input: b is the current belief of the agent, D the desired depth of the search, and K the number of particles to use to compute the next belief states. At the end of this procedure, the agent executes action *bestA* in the environment and restarts this procedure with its next belief. Note here that an approximate value function $\hat{V}$ can be used to approximate the long term return obtained by the optimal policy from the fringe beliefs. For efficiency reasons, we simply defined $\hat{V}(b)$ to be the maximum immediate reward in belief b throughout our experiments. The overall complexity of this planning approach is $O((|A||Z|)^D C_b)$, where C_b is the complexity of updating the belief.

```
1: function $V(b,d,K)$
2: if $d = 0$ then
3:     return $\hat{V}(b)$
4: end if
5: $maxQ \leftarrow -\infty$
6: for all $a \in A$ do
7:     $q \leftarrow \sum_{(s,\phi,\psi)\in S'_b} b(s,\phi,\psi)R(s,a)$
8:     for all $z \in Z$ do
9:         $b' \leftarrow \hat{\tau}(b,a,z,K)$
10:        $q \leftarrow q + \gamma \Pr(z|b,a) V(b',d-1,K)$
11:    end for
12:    if $q > maxQ$ then
13:        $maxQ \leftarrow q$
14:        $maxA \leftarrow a$
15:    end if
16: end for
17: if $d = D$ then
18:     $bestA \leftarrow maxA$
19: end if
20: return $maxQ$
```

Algorithm 3: Online Planning in the BAPOMDP.

In general, planning via forward search can be improved by using an accurate simulator, a good exploration policy, and a good heuristic function. For example, any offline POMDP solution can be used at the leaves of the lookahead search to improve search quality (Ross et al., 2008c). Additionally, more efficient online planning algorithms presented in Ross et al. (2008c) could be used provided one can compute an informative upper bound and lower bound on the value function of the BAPOMDP.

6. Empirical Evaluation

The main focus of this paper is on the definition of the Bayes-Adaptive POMDP model, and examination of its theoretical properties. Nonetheless it is useful to consider experiments on a few sample domains to verify that the algorithms outlined in Section 5 produce reasonable results. We begin by comparing the three different belief approximations introduced above. To do so, we use a simple online d-step lookahead search, and compare the overall expected return and model accuracy in three different problems: the well-known Tiger domain (Kaelbling et al., 1998), a new domain called Follow which simulates simple human-robot interactions and finally a standard robot planning domain known as RockSample (Smith and Simmons, 2004).

Given $T^{sas'}$ and $O^{s'az}$ the exact probabilities of the (unknown) POMDP, the model accuracy is measured in terms of the weighted sum of L1-distance, denoted $WL1$, between the exact model and the probable models in a belief state b:

$$\begin{array}{rcl} WL1(b) & = & \sum_{(s,\phi,\psi)\in S'_b} b(s,\phi,\psi) L1(\phi,\psi) \\ L1(\phi,\psi) & = & \sum_{a\in A}\sum_{s'\in S}\left[\sum_{s\in S}|T_{\phi}^{sas'} - T^{sas'}| + \sum_{z\in Z}|O_{\psi}^{s'az} - O^{s'az}|\right] \end{array}$$

6.1 Tiger

The Tiger problem (Kaelbling et al., 1998) is a 2-state POMDP, $S = \{tiger\_left, tiger\_right\}$, describing the position of the tiger. The tiger is assumed to be behind a door; its location is inferred through a noisy observation, $Z = \{hear\_right, hear\_left\}$. The agent has to select whether to open a door (preferably such as to avoid the tiger), or listen for further information, $A = \{open\_left, open\_right, listen\}$. We consider the case where the transition and reward parameters are known, but the observation probabilities are not. Hence, there are four unknown parameters: $O_{Ll}, O_{Lr}, O_{Rl}, O_{Rr}$ (O_{Lr} stands for $\Pr(z = hear\_right|s = tiger\_left, a = listen)$). We define the observation count vector, $\psi = (\psi_{Ll}, \psi_{Lr}, \psi_{Rl}, \psi_{Rr})$, and consider a prior of $\psi_0 = (5,3,3,5)$, which specifies an expected sensor accuracy of 62.5% (instead of the correct 85%) in both states. Each simulation consists of 100 episodes. Episodes terminate when the agent opens a door, at which point the POMDP state (i.e., tiger's position) is reset, but the distribution over count vectors is carried over to the next episode.

Figure 1 shows how the average return and model accuracy evolve over the 100 episodes (results are averaged over 1000 simulations), using an online 3-step lookahead search with varying belief approximations and parameters. Returns obtained by planning directly with the prior and exact model (without learning) are shown for comparison. Model accuracy is measured on the initial belief of each episode. Figure 1 also compares the average planning time per action taken by each approach. We observe from these figures that the results for the Most Probable and Weighted Distance approximations are similar and perform well even with few particles. On the other hand, the performance of the Monte-Carlo belief tracking is much weaker, even using many more particles (64). The Most Probable approach yields slightly more efficient planning times than the Weighted Distance approximation.

6.2 Follow

We also consider a new POMDP domain, called Follow, inspired by an interactive human-robot task. It is often the case that such domains are particularly subject to parameter uncertainty (due to the difficulty in modeling human behavior), thus this environment motivates the utility of Bayes-Adaptive POMDP in a very practical way. The goal of the Follow task is for a robot to continuously follow one of two individuals in a 2D open area. The two subjects have different motion behavior, requiring the robot to use a different policy for each. At every episode, the target person is selected randomly with $Pr = 0.5$ (and the other is not present). The person's identity is not observable (except through their motion). The state space has two features: a binary variable indicating which person is being followed, and a position variable indicating the person's position relative to the robot (5×5 square

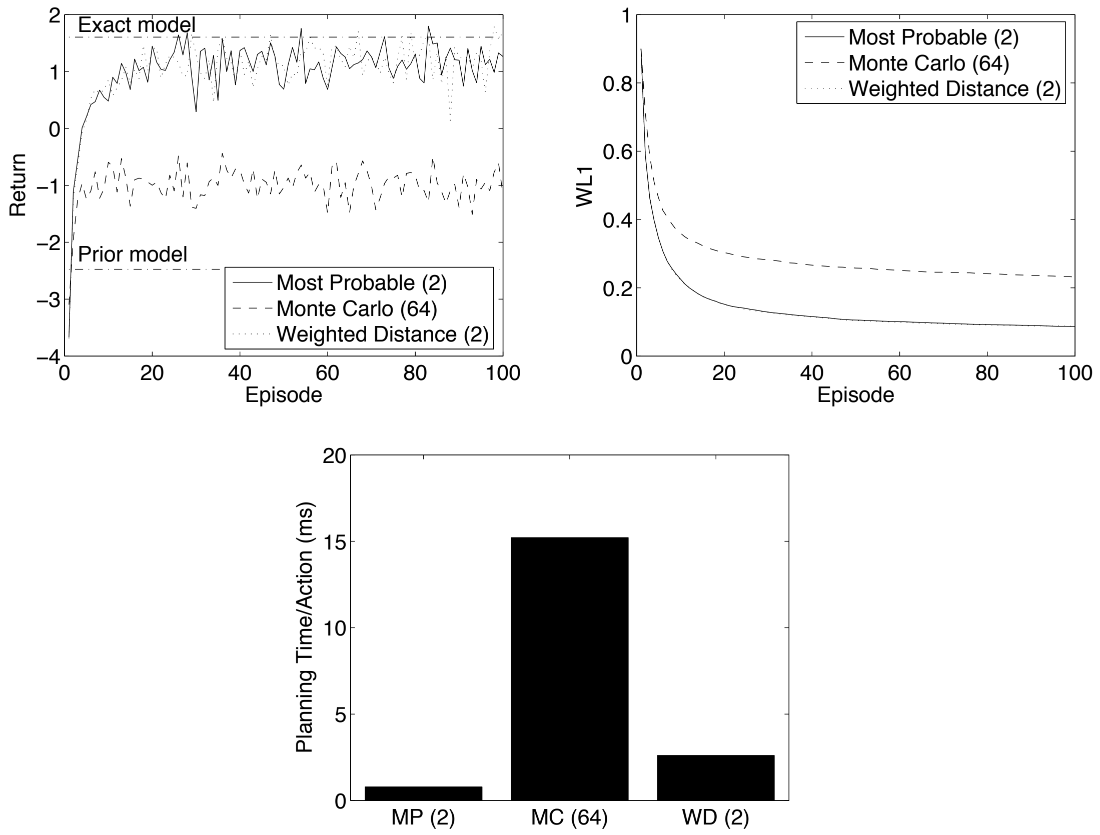


Figure 1: Tiger: Empirical return (top left), belief estimation error (top right), and planning time (bottom), for different belief tracking approximation.

grid with the robot always at the center). Initially, the robot and person are at the same position. Both the robot and the person can perform five motion actions $\{NoAction, North, East, South, West\}$. The person follows a fixed stochastic policy (stationary over space and time), but the parameters of this behavior are unknown. The robot perceives observations indicating the person's position relative to the robot: $\{Same, North, East, South, West, Unseen\}$. The robot perceives the correct observation $Pr = 0.8$ and $Unseen$ with $Pr = 0.2$. The reward $R = +1$ if the robot and person are at the same position (central grid cell), $R = 0$ if the person is one cell away from the robot, and $R = -1$ if the person is two cells away. The task terminates if the person reaches a distance of 3 cells away from the robot, also causing a reward of -20. We use a discount factor of 0.9.

When formulating the BAPOMDP, the robot's motion model (deterministic), the observation probabilities, and the rewards are all assumed to be known. However we consider the case where each person's motion model is unknown. We maintain a separate count vector for each person, representing the number of times they move in each direction, that is, $\phi^1 = (\phi^1_{NA}, \phi^1_N, \phi^1_E, \phi^1_S, \phi^1_W)$, $\phi^2 = (\phi^2_{NA}, \phi^2_N, \phi^2_E, \phi^2_S, \phi^2_W)$. We assume a prior $\phi^1_0 = (2,3,1,2,2)$ for person 1 and $\phi^2_0 = (2,1,3,2,2)$ for person 2, while in reality person 1 moves with probabilities $Pr = (0.3, 0.4, 0.2, 0.05, 0.05)$ and person 2 with $Pr = (0.1, 0.05, 0.8, 0.03, 0.02)$. We run 200 simulations, each consisting of 100 episodes (of at most 10 time steps). The count vectors' distributions are reset after every simulation, and the target person is reset after every episode. We use a 2-step lookahead search for planning in the BAPOMDP.

Figure 2 shows how the average return and model accuracy evolve over the 100 episodes (averaged over the 200 simulations) with different belief approximations. Figure 2 also compares the planning time taken by each approach. We observe from these figures that the results for the Weighted Distance approximations are much better both in terms of return and model accuracy, even with fewer particles (16). Monte-Carlo fails at providing any improvement over the prior model, which indicates it would require much more particles. Running Weighted Distance with 16 particles require less time than both Monte-Carlo and Most Probable with 64 particles, showing that it can be more time efficient for the performance it provides in complex environment.

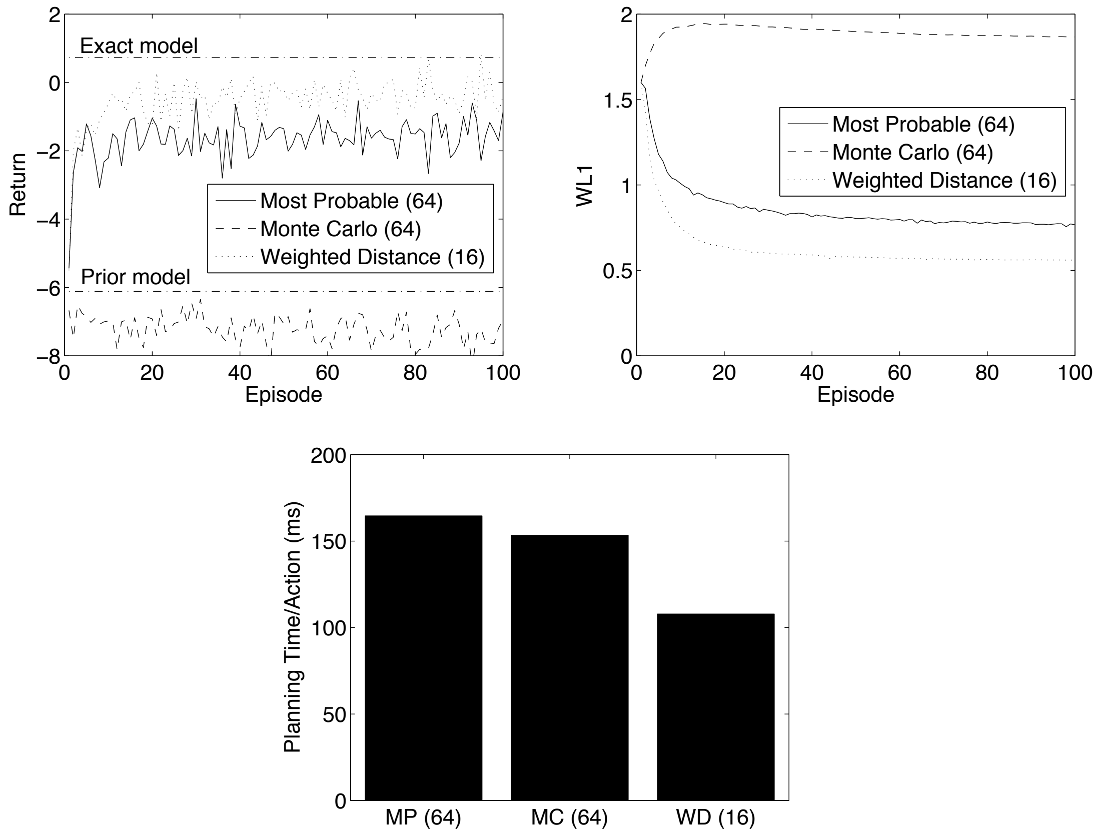


Figure 2: Follow: Empirical return (top left), belief estimation error (top right), and planning time (bottom), for different belief tracking approximation.

6.3 RockSample

To test our algorithm against problems with a larger number of states, we consider the RockSample problem (Smith and Simmons, 2004). In this domain, a robot is on an $n \times n$ square board, with rocks on some of the cells. Each rock has an unknown binary quality (good or bad). The goal of the robot is to gather samples of the good rocks. Sampling a good rock yields high reward (+10), in contrast to sampling a bad rock (-10). However a sample can only be acquired when the robot is in the same cell as the rock. The number of rocks and their respective positions are fixed and known, while their qualities are fixed but unknown. A state is defined by the position of the robot on the board and the quality of all the rocks. With an $n \times n$ board and k rocks, the number of states is then $n^2 2^k$. Most results below assume $n = 3$ and $k = 2$, which makes 36 states. The robot can choose between 4 (deterministic) motion actions to move to neighboring cells, the Sample action, and a Sensor action for each rock, so there are $k + 5$ actions in general. The robot is able to acquire information on the quality of each rock by using the corresponding sensor action. The sensor returns either GOOD or BAD, according to the quality of the rock. The sensor can be used when the robot is away from the rock, but the accuracy depends on the distance d between the robot and the rock. As in the original problem, the accuracy η of the sensor is given by $\eta = 2^{-d/d_0}$.

6.3.1 INFLUENCE OF LARGE NUMBER OF STATES

We consider the case where transition probabilities are known, and the agent must learn its observation function. The prior knowledge over the structure of the observation function is as follows:

- the probability distribution over observations after performing action CHECK$_i$ in state s depends only on the distance between the robot and the rock i;
- at a given distance d, the probability of observing GOOD when the rock is a good one is equal to the probability of observing BAD when the rock is a bad one. This means that for each distance d, the robot's sensor has a probability to give incorrect observations, which doesn't depend of the quality of the rock.

These two assumptions seem reasonable in practice, and allow the robot to learn a model efficiently without having to try all CHECK actions in all states.

We begin by comparing performance of the BAPOMDP framework with different belief approximations. For the belief tracking, we focus on the Most Probable and Weighted Distance Minimization approximations, knowing that the Monte Carlo has given poor results in the two smaller domains. Each simulation consists of 100 episodes, and the results are averaged over 100 simulations.

As we can see in Figure 3(left), the Most Probable approximation outperforms Weighted Distance Minimization; in fact, after only 50 iterations, it reaches the same level of performance as a robot that knows the true model. Figure 3(right) sheds further light on this issue, by showing, for each episode, the maximum L_1 distance between the estimated belief $\hat{b}(s) = \sum_{\psi,\phi} b(s, \phi, \psi)$, and the correct belief $b(s)$ (assuming the model is known *a priori*). We see that this distance decreases for both approximations, and that it reaches values close to 0 after 50 episodes for the Most Probable approximation. This suggests that the robot has reached a point where it knows its model well enough to have the same belief over the physical states as a robot who would know the true model. Note that the error in belief estimate is calculated over the trajectories; it is possible that the estimated model is wrong in parts of the beliefs which are not visited under the current (learned) policy.

To further verify the scalability of our approach, we consider larger versions of the RockSample domain in Figure 4. Recall that for k rocks and an $n \times n$ board, the domain has state space $|S| = n^2 2^k$ and action space $|A| = 5 + k$. For this experiment, and all subsequent ones, belief tracking

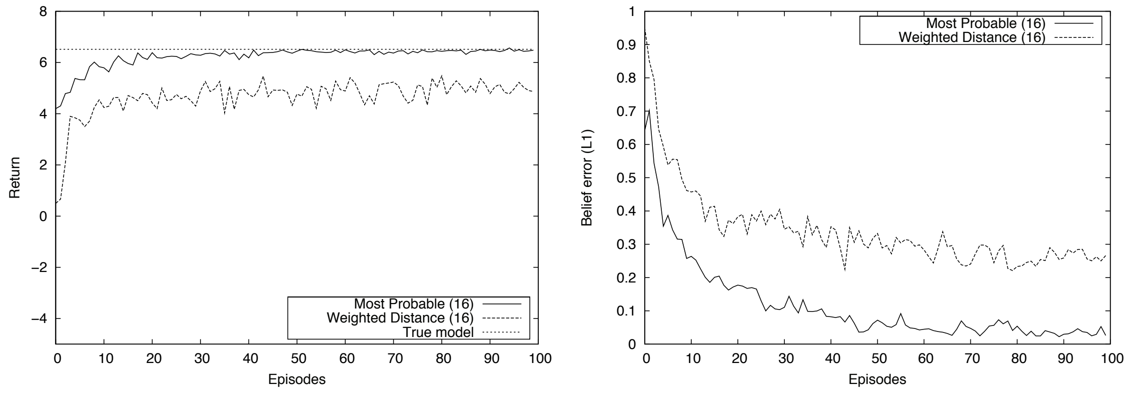


Figure 3: RockSample: Empirical return (left) and belief estimation error (right) for different belief tracking approximation.

in the BAPOMDP is done with the Most Probable approximation (with $K = 16$). As expected, the computational time for planning grows quickly with n and k. Better solutions could likely be obtained with appropriate use of heuristics in the forward search planner (Ross et al., 2008c).

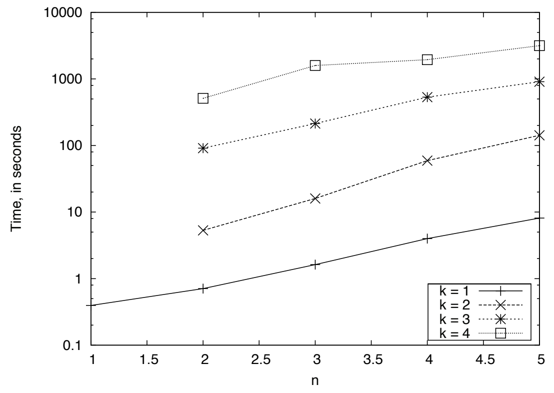


Figure 4: RockSample: Computational time for different values of k and n. All results are computed with $K = 16$ and a depth=3 planning horizon.

6.3.2 INFLUENCE OF THE PRIORS

The choice of prior plays an important role in Bayesian Learning. As explained above, in the Rock-Sample domain we have constrained the structure of the observation probability model structural assumptions in the prior. For all results presented above, we used a prior made of 4 ϕ-vectors with probability $\frac{1}{4}$ each. Each of those vectors ϕ_i is made of coefficients (ϕ_{ij}), where ϕ_{ij} is the probability that the sensor will give a correct observation at distance j. For each of the 4 vectors ϕ_i, we sample the coefficients ϕ_{ij} from an uniform distribution between 0.45 and 0.95. We adopt this approach for a number of reasons. First, this prior is very general, in assuming that the sensor's probability to make a mistake is uniformly distributed between 0.05 and 0.55, at every distance d. Second, by sampling a new prior for every simulation, we ensure that the results do not depend closely on inadvertent similarities between our prior and the correct model.

We now consider two other forms of prior. First, we consider the case where the coefficients ϕ_{ij} are not sampled uniformly from $\mathcal{U}_{[0.45,0.95]}$, but rather from $\mathcal{U}_{[\phi_j^* \pm \varepsilon]}$, where ϕ_j^* is the value of the true model (that is, the probability that the true sensor gives a true observation at distance j). We consider performance for various levels of noise, $0 \leq \varepsilon \leq 0.25$. This experiment allows us to measure the influence of prior uncertainty on the performances of our algorithm. The results in Figure 5 show that the BAPOMDP agent performs well for various levels of initial uncertainty over the model. As expected, the fact that all the priors have ϕ_{ij} coefficients centered around the true value ϕ_j^* carries in itself substantial information, in many cases enough for the robot to perform very well from the first episode (note that the y-axis in Fig. 5 is different than that in Fig. 3). Furthermore, we observe that the noise has very little influence on the performances of the robot: for all values of ε, the empirical return is above 6.3 after only 30 episodes.

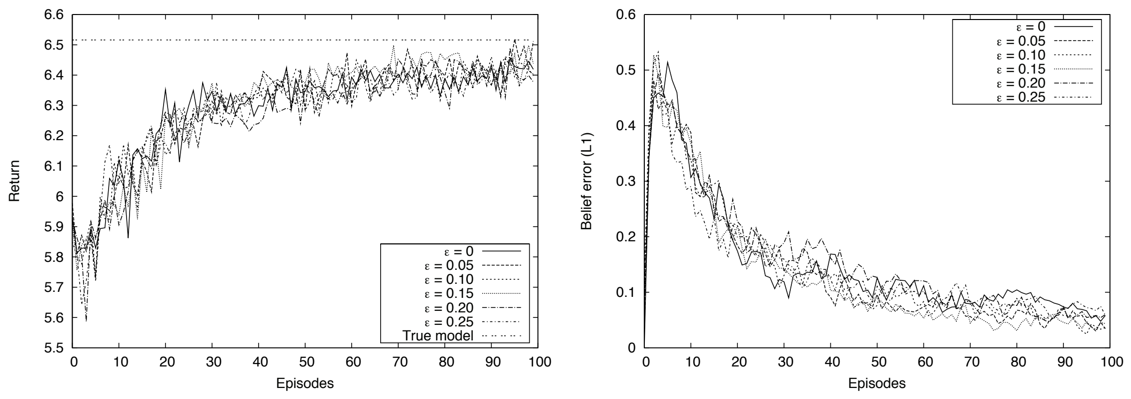


Figure 5: Performance of BAPOMDP with centered uniform priors in RockSample domain, using the Most Probably (K=16) belief tracking approximation. Empirical return (left). Belief state tracking error (right).

Second, we consider the case where there is only one ϕ-vector, which has probability one. This vector has coefficients ϕ_j, such that for all j, $\phi_j = \frac{k-1}{k}$, for different values of k. This represents a beta distribution of parameters $(1,k)$, where 1 is the count of wrong observations, and k the count of correct observations. The results presented in Figure 6 show that for all values of k, the rewards converge towards the optimal value within 100 episodes. We see that for high values of k, the robot needs more time to converge towards optimal rewards. Indeed, those priors have a large total count (k+1), which means their variance is small. Thus, they need more time to correct themselves towards the true model. In particular, the $(1,16)$ is very optimistic (it considers that the sensor only makes an error with probability $\frac{1}{17}$), which causes the robot to make mistakes during the first experiments, thus earning poor rewards at the beginning, and needing about 80 episodes to learn a sufficiently good model to achieve near-optimal performance. The right-side graph clearly shows how the magnitude of the initial k impacts the error in the belief over physical states (indicating that the robot doesn't know the quality of the rocks as well as if it knew the correct model). The error in

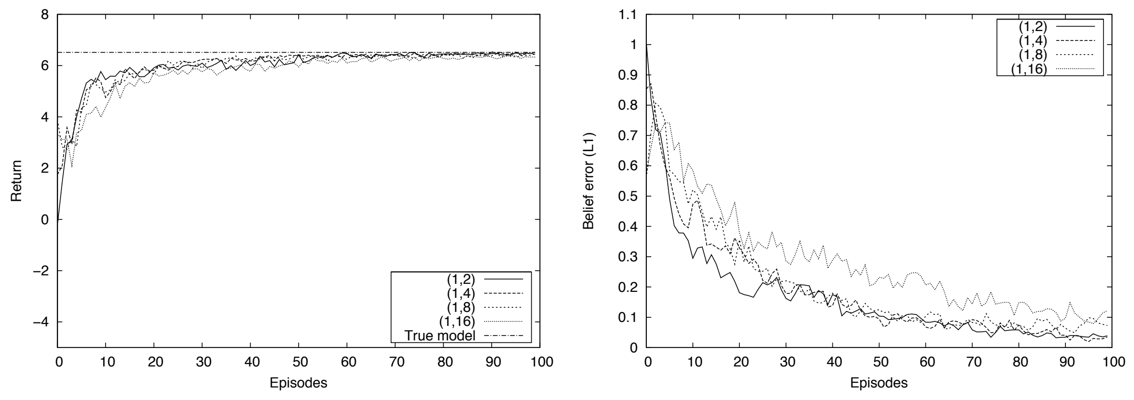


Figure 6: Performance of BAPOMDP with Beta priors in RockSample domain, using the Most Probable (K=16) belief tracking approximation. Empirical return (left). Belief state tracking error (right).

belief state tracking is significantly reduced after about 80 iterations, confirming that our algorithm is able to overcome poor priors, even those with high initial confidence.

Finally, we consider the case where the true underlying POMDP model is changed such that the sensor has a constant probability ε of making mistakes for all distances; the prior is sampled as for the results of Figure 3. This makes the situation harder for the robot, because it increases its sensor's overall probability of making mistakes, including at distance zero (i.e., when the robot is on the same cell as the rock). The empirical results presented in Figure 7 show a decrease in the empirical return as ε increases. Similarly, as shown in the right graph, the learning performance suffers with higher values of ε. This is not surprising since a higher ε indicates that the robot's CHECKs are more prone to error, which makes it more difficult for the robot to improve its knowledge about its physical states, and thus about its model. In fact, it is easy to verify that the optimal return (assuming a fully known model) is lower for the noisier model. In general, in domains where the observations are noisy or aliased, it is difficult for the agent to learn a good model, as well as perform well (unless the observations are not necessary for good performance).

7. Related Work

A few recent approaches have tackled the problem of joint planning and learning under partial (state and model) observability using a Bayesian framework. The work of Poupart and Vlassis (2008) is probably closest to the BAPOMDP outlined here. Using a similar Bayesian representation of model uncertainty, they proposed an extension of the Beetle algorithm (Poupart et al., 2006) (original designed for fully observable domains) to compute an approximate solution for BAPOMDP-type problems. Their work is presented in the context of factored representations, but the model learning is done using similar Bayesian mechanisms, namely by updating a posterior represented by a mixture of Dirichlet distributions. They outline approximation methods to maintain a compact belief set that are similar to the Most Probably and Monte-Carlo methods outlined in Section 5.1 above. Presumably our Weighted Distance minimization technique could be extended to their factored representation, assuming one can compute the distance metric. Finally, they propose an offline planning

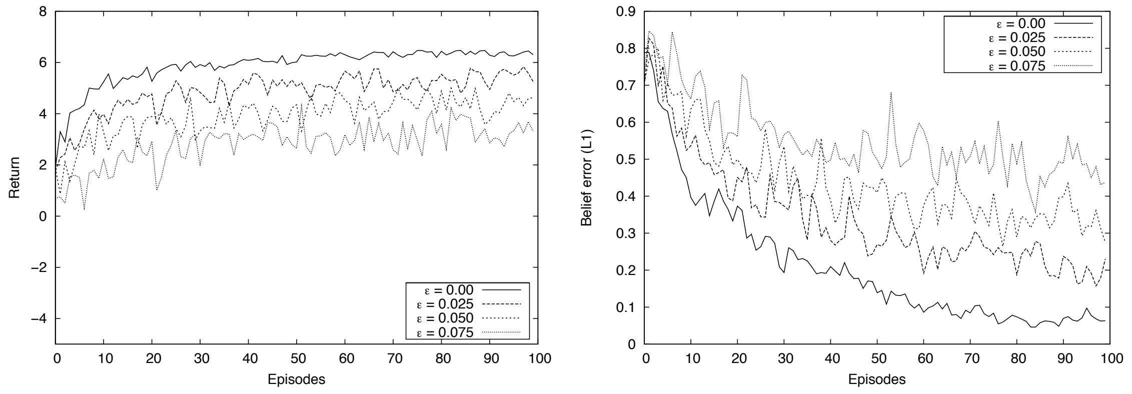


Figure 7: Performance of BAPOMDP with varying observation models in RockSample domain. Empirical return (left). Belief error (right).

algorithm, similar to the literature on point-based POMDP solvers, to find a policy. However we are not aware of any empirical validation with this approach, thus scalability and expressivity in experimental domains remains to be determined.

Jaulmes et al. (2005) have for their part considered active learning in partially observable domains where information gathering actions are provided by oracles that reveal the underlying state. The key assumption of this approach, which is not used in other model-free approaches, concerns the existence of this oracle (or human) which is able to correctly identify the state following each transition. This makes it much easier to know how to update the prior. In the same vein than Jaulmes and colleagues, Doshi et al. (2008) developed an approach for active learning in POMDPs that can robustly determine a near-optimal policy. To achieve that, they introduced meta-queries (questions about action) and a risk-averse action selection criterion that allows agents to behave robustly even with uncertain knowledge of the POMDP model. Finally, Doshi-Velez (2010) proposed a Bayesian learning framework for the case of POMDPs where the number of states is not known *a priori*, thus allowing the number of states to grow gradually as the agent explores the world, while simultaneously updating a posterior over the parameters.

The work on Universal Artificial Intelligence (Hutter, 2005) presents an interesting alternative to the framework of BAPOMDPs. It tackles a similar problem, namely sequential decision-making under (general) uncertainty. But Hutter's AIXI framework is more general, in that it allows the model to be sampled from any computable distribution. The learning problem is constrained by placing an Occam's razor prior (measured by Kolmogorov complexity) over the space of models. The main drawback is that inference in this framework is incomputable, though an algorithm is presented for computing time/space-bounded solutions. Further development of a general purpose AIXI learning/planning algorithm would be necessary to allow a direct comparison between AIXI and BAPOMDPs on practical problems. Recent results in Monte-Carlo Planning provide a good basis for this (Silver and Veness, 2010; Veness et al., 2011).

A number of useful theoretical results have also been published recently. For the specific case of exploration in reinforcement learning, Asmuth et al. (2009) presented a fully Bayesian analysis of the performance of a sampling approach. Subsequently, Kolter and Ng (2009) clarified the relation between Bayesian and PAC-MDP approaches and presented a simple algorithm for efficiently achieving near-Bayesian exploration.

Finally, it is worth emphasizing that Bayesian approaches have also been investigated in the control literature. The problem of optimal control under uncertain model parameters was originally introduced by Feldbaum (1961), as the theory of dual control, also sometimes referred to as adaptive dual control. Extensions of this theory have been developed for time-varying systems (Filatov and Unbehauen, 2000). Several authors have studied this problem for different kinds of dynamical systems : linear time invariant systems under partial observability (Rusnak, 1995), linear time varying Gaussian models under partial observability (Ravikanth et al., 1992), nonlinear systems with full observability (Zane, 1992), and more recently a non linear systems under partial observability (Greenfield and Brockwell, 2003). All this work is targeted towards specific classes of continuous systems, and we are not aware of similar work in the control literature for discrete (or hybrid) systems.

8. Conclusion

The problem of sequential decision-making under model uncertainty arises in many practical applications of AI and decision systems. Developing effective models and algorithms to handle these problems under realistic conditions—including stochasticity, partial state observability, and model inaccuracy—is imperative if we hope to deploy robots and other autonomous agents in real-world situations.

This paper focuses in particular on the problem of simultaneous learning and decision-making in dynamic environments under partial model and state uncertainty. We adopt a model-based Bayesian reinforcement learning framework, which allows us to explicitly target the exploration-exploitation problem in a coherent mathematical framework. Our work is a direct extension of previous results on model-based Bayesian reinforcement learning in fully observable domains.

The main contributions of the paper pertains to the development of the Bayes-Adaptive POMDP model, and analysis of its theoretical properties. This work addresses a number of interesting questions, including:

1. defining an appropriate model for POMDP parameter uncertainty,
2. approximating this model while maintaining performance guarantees,
3. performing tractable belief updating, and
4. optimizing action sequences given a posterior over state and model uncertainty.

From the theoretical analysis, we are able to derive simple algorithms for belief tracking and (near-)optimal decision-making in this model. We illustrate performance of these algorithms in a collection of synthetic POMDP domains. Results in the Follow problem showed that our approach is able to learn the motion patterns of two (simulated) individuals. This suggests interesting applications in human-robot interaction, where we often lack good models of human behavior and where it is imperative that an agent be able to learn quickly, lest the human user lose interest (this is in contrast to robot navigation tasks, for which we often have access to more precise dynamical models and/or high-fidelity simulators). For their part, results of RockSample problem shows how one should take into account prior knowledge on agent's sensors when this knowledge is available.

While the BAPOMDP model provides a rich model for sequential decision-making under uncertainty, it has a number of important limitations. First, the model and theoretical analysis are limited to discrete domains. It is worth noting however that the approximate algorithms extend quite easily to the continuous case (Ross et al., 2008b), at least for some families of dynamical systems. Other related references for the continuous case are available in the control literature, as described in Section 7.

Another limitation is the fact that the model requires specification of a prior. This is standard in the Bayesian RL framework. The main concern is to ensure that the prior assigns some weight to the correct model. Our empirical evaluation shows good performance for a range of priors; though the issue of choosing good priors in large domains remains a challenge in general. Our empirical results also confirm standard Bayesian intuition, whereby the influence of the prior is particularly important for any inference and decision-making performed when only a small amount of data has been observed, but the influence becomes negligible as large amounts of data are acquired.

As a word of caution, problems may arise in cases where Bayesian RL is used to infer both transition and observation probabilities simultaneously, while the rewards are not explicitly perceived through the observations (even if the rewards are known *a priori*). In this challenging setting, the Bayes-Adaptive POMDP framework as outlined above might converge to an incorrect model if the initial priors on the transition and observation model are non-informative. This is mainly due to the fact that many possible parameters may correctly explain the observed action-observation sequences. While the agent is able to predict observations correctly, this leads to poor prediction of rewards and thus possibly sub-optimal long-term rewards. However if the rewards are observable, and their probabilities taken into account in the belief update, such problems do not arise, in the sense that the agent learns an equivalent model that correctly explains the observed action-observation-reward sequence and recovers a good policy for the unknown POMDP model. In the latter case, where rewards are observable, the framework presented in this paper can be used with only minor modifications to also learn the reward function.

Finally, it is worth pointing out that Bayesian RL methods in general have not been deployed in real-world domains yet. We hope that the work presented here will motivate further investigation of practical issues pertaining to the application and deployment of this class of learning approaches.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT), and the National Institutes of Health (grant R21DA019800). We would also like to thank Michael Bowling, Patrick Dallaire, Mohammad Ghavamzadeh, Doina Precup, Prakash Panangaden, as well as the anonymous reviewers, for offering thoughtful comments on earlier versions of this work.

Appendix A. Theorems and Proofs

This appendix presents the proofs of the theorems presented throughout this paper. Theorems 1 and 2 are presented first, then some useful lemmas, followed by the proofs of the remaining Theorems.

Theorem 1 *Let (S',A,Z,T',O',R',γ) be a BAPOMDP constructed from the POMDP (S,A,Z,T,O,R,γ). If S is finite, then at any time t, the set $S'_{b'_t} = \{\sigma \in S' | b'_t(\sigma) > 0\}$ has size $|S'_{b'_t}| \leq |S|^{t+1}$.*

Proof Proof by induction. When $t = 0$, $b'_0(s,\phi,\psi) > 0$ only if $\phi = \phi_0$ and $\psi = \psi_0$. Hence $|S'_{b'_0}| \leq |S|$. For the general case, assume that $|S'_{b'_{t-1}}| \leq |S|^t$. From the definitions of the belief update function, $b'_t(s',\phi',\psi') > 0$ iff $\exists(s,\phi,\psi)$ such that $b'_{t-1}(s,\phi,\psi) > 0$, $\phi' = \phi + \delta^a_{ss'}$ and $\psi' = \psi + \delta^a_{s'z}$. Hence, a particular (s,ϕ,ψ) such that $b'_{t-1}(s,\phi,\psi) > 0$ yields non-zero probabilities to at most $|S|$ different states in b'_t. Since $|S'_{b'_{t-1}}| \leq |S|^t$ by assumption, then if we generate $|S|$ different probable states in b'_t, for each probable state in $S'_{b_{t-1}}$ it follows that $|S'_{b'_t}| \leq |S|^{t+1}$. ■

Theorem 2 *For any horizon t, there exists a finite set Γ_t of functions $S' \to \mathbb{R}$, such that $V^*_t(b) = \max_{\alpha\in\Gamma_t} \sum_{\sigma\in S'} \alpha(\sigma)b(\sigma)$.*

Proof Proof by induction. This holds true for horizon $t = 1$, since $V^*_1(b) = \max_{a\in A} \sum_{(s,\phi,\psi)} b(s,\phi,\psi)R(s,a)$. Hence by defining $\Gamma_1 = \{\alpha_a | \alpha_a(s,\phi,\psi) = R(s,a), a \in A\}$, $V^*_1(b) = \max_{\alpha\in\Gamma_1} \sum_{\sigma\in S'} b(\sigma)\alpha(\sigma)$. By induction, we assume that there exists a set Γ_t such that $V^*_t(b) = \max_{\alpha\in\Gamma_t} \sum_{\sigma\in S'} b(\sigma)\alpha(\sigma)$.
Now $V^*_{t+1}(b) = \max_{a\in A} \left[\sum_{(s,\phi,\psi)} b(s,\phi,\psi)R(s,a) + \sum_{z\in Z} \Pr(z|b,a)V^*_t(b^{az})\right]$. Hence:

$$\begin{aligned}
V^*_{t+1}(b) &= \max_{a\in A} \left[\textstyle\sum_{(s,\phi,\psi)} b(s,\phi,\psi)R(s,a) + \sum_{z\in Z} \Pr(z|b,a) \max_{\alpha\in\Gamma_t} \sum_{\sigma\in S'} b^{az}(\sigma)\alpha(\sigma)\right] \\
&= \max_{a\in A} \left[\textstyle\sum_{(s,\phi,\psi)} b(s,\phi,\psi)R(s,a) + \sum_{z\in Z} \max_{\alpha\in\Gamma_t} \sum_{\sigma\in S'} \Pr(z|b,a)b^{az}(\sigma)\alpha(\sigma)\right] \\
&= \max_{a\in A} \left[\textstyle\sum_{(s,\phi,\psi)} b(s,\phi,\psi)R(s,a) + \right. \\
&\qquad \left.\textstyle\sum_{z\in Z} \max_{\alpha\in\Gamma_t} \sum_{(s,\phi,\psi)\in S'} \sum_{s'\in S} b(s,\phi,\psi)T^{sas'}_{\phi} O^{s'az}_{\psi} \alpha(s', \mathcal{U}(\phi,s,a,s'), \mathcal{U}(\psi,s',a,z))\right].
\end{aligned}$$

Thus if we define:

$$\begin{aligned}
\Gamma_{t+1} &= \{\alpha_{a,f} | \alpha_{a,f}(s,\phi,\psi) = R(s,a) + \\
&\qquad \textstyle\sum_{z\in Z} \sum_{s'\in S} T^{sas'}_{\phi} O^{s'az}_{\psi} f(z)(s', \mathcal{U}(\phi,s,a,s'), \mathcal{U}(\psi,s',a,z)), a \in A, f \in [Z \to \Gamma_t]\},
\end{aligned}$$

then $V^*_{t+1}(b) = \max_{\alpha\in\Gamma_{t+1}} \sum_{\sigma\in S'} b(\sigma)\alpha(\sigma)$ and Γ_{t+1} is finite since $|\Gamma_{t+1}| = |A||\Gamma_t|^{|Z|}$, which is finite by assumptions that A, Z and Γ_t are all finite. ■

For some of the following theorems, lemmas and proofs, we will sometime denote the Dirichlet count update operator $\mathcal{U}$, as defined for the BAPOMDP, as a vector addition: $\phi' = \phi + \delta^a_{ss'} = \mathcal{U}(\phi,s,a,s')$, that is, $\delta^a_{ss'}$ is a vector full of zeros, with a 1 for the element $\phi^a_{ss'}$.

Lemma 1 *For any $t \geq 2$, any α-vector $\alpha_t \in \Gamma_t$ can be expressed as $\alpha^{a,\alpha'}_t(s,\phi,\psi) = R(s,a) + \gamma \sum_{z\in Z} \sum_{s\in S'} T^{sas'}_{\phi} O^{s'az}_{\psi} \alpha'(z)(s', \phi + \delta^a_{ss'}, \psi + \delta^a_{s'z})$ for some $a \in A$, and α' defining a mapping $Z \to \Gamma_{t-1}$.*

Proof Follows from proof of theorem 2. ■

Lemma 2 *Given any $a,b,c,d \in \mathbb{R}$, $ab - cd = \frac{(a-c)(b+d)+(a+c)(b-d)}{2}$.*

Proof Follows from direct computation. ■

Lemma 3 *Given any $\phi, \phi' \in \mathcal{T}$, $\psi, \psi' \in \mathcal{O}$, then for all $s \in S$, $a \in A$, we have that* $\sum_{s' \in S} \sum_{z \in Z} \left| \frac{\phi'^a_{ss'} \psi'^a_{s'z}}{\mathcal{N}^{sa}_{\phi'} \mathcal{N}^{s'a}_{\psi'}} - \frac{\phi^a_{ss'} \psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi} \mathcal{N}^{s'a}_{\psi}} \right| \le D^{sa}_S(\phi', \phi) + \sup_{s' \in S} D^{s'a}_Z(\psi', \psi)$.

Proof Using lemma 2, we have that:

$$\begin{array}{cl}
& \sum_{s' \in S} \sum_{z \in Z} \left| \frac{\phi'^a_{ss'} \psi'^a_{s'z}}{\mathcal{N}^{sa}_{\phi'} \mathcal{N}^{s'a}_{\psi'}} - \frac{\phi^a_{ss'} \psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi} \mathcal{N}^{s'a}_{\psi}} \right| \\
= & \frac{1}{2} \sum_{s' \in S} \sum_{z \in Z} \left| \left(\frac{\phi'^a_{ss'}}{\mathcal{N}^{sa}_{\phi'}} - \frac{\phi^a_{ss'}}{\mathcal{N}^{sa}_{\phi}} \right) \left(\frac{\psi'^a_{s'z}}{\mathcal{N}^{s'a}_{\psi'}} + \frac{\psi^a_{s'z}}{\mathcal{N}^{s'a}_{\psi}} \right) + \left(\frac{\phi'^a_{ss'}}{\mathcal{N}^{sa}_{\phi'}} + \frac{\phi^a_{ss'}}{\mathcal{N}^{sa}_{\phi}} \right) \left(\frac{\psi'^a_{s'z}}{\mathcal{N}^{s'a}_{\psi'}} - \frac{\psi^a_{s'z}}{\mathcal{N}^{s'a}_{\psi}} \right) \right| \\
\le & \frac{1}{2} \sum_{s' \in S} \left| \frac{\phi'^a_{ss'}}{\mathcal{N}^{sa}_{\phi'}} - \frac{\phi^a_{ss'}}{\mathcal{N}^{sa}_{\phi}} \right| \sum_{z \in Z} \left| \frac{\psi'^a_{s'z}}{\mathcal{N}^{s'a}_{\psi'}} + \frac{\psi^a_{s'z}}{\mathcal{N}^{s'a}_{\psi}} \right| + \frac{1}{2} \sum_{s' \in S} \left| \frac{\phi'^a_{ss'}}{\mathcal{N}^{sa}_{\phi'}} + \frac{\phi^a_{ss'}}{\mathcal{N}^{sa}_{\phi}} \right| \sum_{z \in Z} \left| \frac{\psi'^a_{s'z}}{\mathcal{N}^{s'a}_{\psi'}} - \frac{\psi^a_{s'z}}{\mathcal{N}^{s'a}_{\psi}} \right| \\
\le & \sum_{s' \in S} \left| \frac{\phi'^a_{ss'}}{\mathcal{N}^{sa}_{\phi'}} - \frac{\phi^a_{ss'}}{\mathcal{N}^{sa}_{\phi}} \right| + \frac{1}{2} \left[\sup_{s' \in S} \sum_{z \in Z} \left| \frac{\psi'^a_{s'z}}{\mathcal{N}^{s'a}_{\psi'}} - \frac{\psi^a_{s'z}}{\mathcal{N}^{s'a}_{\psi}} \right| \right] \left[\sum_{s' \in S} \left| \frac{\phi'^a_{ss'}}{\mathcal{N}^{sa}_{\phi'}} + \frac{\phi^a_{ss'}}{\mathcal{N}^{sa}_{\phi}} \right| \right] \\
= & \sum_{s' \in S} \left| \frac{\phi'^a_{ss'}}{\mathcal{N}^{sa}_{\phi'}} - \frac{\phi^a_{ss'}}{\mathcal{N}^{sa}_{\phi}} \right| + \sup_{s' \in S} \sum_{z \in Z} \left| \frac{\psi'^a_{s'z}}{\mathcal{N}^{s'a}_{\psi'}} - \frac{\psi^a_{s'z}}{\mathcal{N}^{s'a}_{\psi}} \right| \\
= & D^{sa}_S(\phi', \phi) + \sup_{s' \in S} D^{s'a}_Z(\psi', \psi).
\end{array}$$

■

Lemma 4 *Given any $\phi, \phi', \Delta \in \mathcal{T}$, then for all $s \in S$, $a \in A$,* $D^{sa}_S(\phi + \Delta, \phi' + \Delta) \le D^{sa}_S(\phi, \phi') + \frac{2\mathcal{N}^{sa}_{\Delta} \sum_{s' \in S} |\phi^a_{ss'} - \phi'^a_{ss'}|}{(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta})}$.

Proof We have that:

$$\begin{array}{cl}
& D^{sa}_S(\phi + \Delta, \phi' + \Delta) \\
= & \sum_{s' \in S} \left| \frac{\phi^a_{ss'} + \Delta^a_{ss'}}{\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta}} - \frac{\phi'^a_{ss'} + \Delta^a_{ss'}}{\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta}} \right| \\
= & \sum_{s' \in S} \left| \frac{(\phi^a_{ss'} + \Delta^a_{ss'})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta}) - (\phi'^a_{ss'} + \Delta^a_{ss'})(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})}{(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta})} \right| \\
= & \sum_{s' \in S} \left| \frac{\phi^a_{ss'} \mathcal{N}^{sa}_{\phi'} + \phi^a_{ss'} \mathcal{N}^{sa}_{\Delta} + \Delta^a_{ss'} \mathcal{N}^{sa}_{\phi'} - \phi'^a_{ss'} \mathcal{N}^{sa}_{\phi} - \phi'^a_{ss'} \mathcal{N}^{sa}_{\Delta} - \Delta^a_{ss'} \mathcal{N}^{sa}_{\phi}}{(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta})} \right| \\
\le & \sum_{s' \in S} \left| \frac{\phi^a_{ss'} \mathcal{N}^{sa}_{\phi'} - \phi'^a_{ss'} \mathcal{N}^{sa}_{\phi}}{(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta})} \right| + \sum_{s' \in S} \left| \frac{\mathcal{N}^{sa}_{\Delta}(\phi^a_{ss'} - \phi'^a_{ss'}) + \Delta^a_{ss'}(\mathcal{N}^{sa}_{\phi'} - \mathcal{N}^{sa}_{\phi})}{(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta})} \right| \\
\le & \sum_{s' \in S} \left| \frac{\phi^a_{ss'} \mathcal{N}^{sa}_{\phi'} - \phi'^a_{ss'} \mathcal{N}^{sa}_{\phi}}{\mathcal{N}^{sa}_{\phi} \mathcal{N}^{sa}_{\phi'}} \right| + \frac{\mathcal{N}^{sa}_{\Delta} \left[\sum_{s' \in S} \left| \phi^a_{ss'} - \phi'^a_{ss'} \right| \right] + \left| \mathcal{N}^{sa}_{\phi'} - \mathcal{N}^{sa}_{\phi} \right| \sum_{s' \in S} \Delta^a_{ss'}}{(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta})} \\
= & D^{sa}_S(\phi, \phi') + \frac{\mathcal{N}^{sa}_{\Delta} \left[\sum_{s' \in S} \left| \phi^a_{ss'} - \phi'^a_{ss'} \right| \right] + \mathcal{N}^{sa}_{\Delta} \left| \mathcal{N}^{sa}_{\phi'} - \mathcal{N}^{sa}_{\phi} \right|}{(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta})} \\
\le & D^{sa}_S(\phi, \phi') + \frac{2\mathcal{N}^{sa}_{\Delta} \sum_{s' \in S} \left| \phi^a_{ss'} - \phi'^a_{ss'} \right|}{(\mathcal{N}^{sa}_{\phi} + \mathcal{N}^{sa}_{\Delta})(\mathcal{N}^{sa}_{\phi'} + \mathcal{N}^{sa}_{\Delta})}.
\end{array}$$

■

Lemma 5 *Given any* $\psi, \psi', \Delta \in \mathcal{O}$*, then for all* $s \in S$*,* $a \in A$*,*
$D_Z^{sa}(\psi+\Delta, \psi'+\Delta) \leq D_Z^{sa}(\psi, \psi') + \frac{2\mathcal{N}_\Delta^{sa} \sum_{z\in Z} |\psi_{sz}^a - \psi_{sz}'^a|}{(\mathcal{N}_\psi^{sa}+\mathcal{N}_\Delta^{sa})(\mathcal{N}_{\psi'}^{sa}+\mathcal{N}_\Delta^{sa})}$.

Proof Same proof as for lemma 4, except that we sum over $z \in Z$ in this case. ■

Lemma 6 *Given any* $\gamma \in (0,1)$*, then* $\sup_x \gamma^{x/2} x = \frac{2}{\ln(\gamma^{-e})}$.

Proof We observe that when $x = 0$, $\gamma^{x/2}x = 0$ and $\lim_{x\to\infty} \gamma^{x/2} x = 0$. Furthermore, $\gamma^{x/2}$ is monotonically decreasing exponentially as x increases, while x is monotonically increasing linearly as x increases. Thus it is clear that $\gamma^{x/2}x$ will have a unique global maximum in $(0,\infty)$. We can find this maximum by taking the derivative:

$$\begin{array}{ll} & \frac{\partial}{\partial x}(\gamma^{x/2}x) \\ = & \frac{(\ln\gamma)\gamma^{x/2}x}{2} + \gamma^{x/2} \\ = & \gamma^{x/2}(\frac{(\ln\gamma)x}{2}+1). \end{array}$$

Hence by solving when this is equal 0, we have:

$$\begin{array}{ll} & \gamma^{x/2}(\frac{(\ln\gamma)x}{2}+1) = 0 \\ \Leftrightarrow & \frac{(\ln\gamma)x}{2}+1 = 0 \\ \Leftrightarrow & x = \frac{-2}{\ln\gamma} = -2\log_\gamma(e). \end{array}$$

Hence we have that:

$$\begin{array}{ll} & \gamma^{x/2}x \\ \leq & -2\gamma^{-\log_\gamma(e)}\log_\gamma(e) \\ = & -2e^{-1}\log_\gamma(e) \\ = & \frac{2}{\ln(\gamma^{-e})}. \end{array}$$

■

Lemma 7 $\sup_{\alpha_1\in\Gamma_1, s\in S} |\alpha_1(s,\phi,\psi) - \alpha_1(s,\phi',\psi')| = 0$ *for any* ϕ*,* ϕ'*,* ψ*,* ψ'.

Proof For any $a \in A, s \in S$, $|\alpha_1^a(s,\phi,\psi) - \alpha_1^a(s,\phi',\psi')| = |R(s,a) - R(s,a)| = 0$. ■

Theorem 3 *Given any* $\phi, \phi' \in \mathcal{T}$*,* $\psi, \psi' \in \mathcal{O}$ *and* $\gamma \in (0,1)$*, then* $\forall t$*:*

$$\sup_{\alpha_t\in\Gamma_t, s\in S} |\alpha_t(s,\phi,\psi) - \alpha_t(s,\phi',\psi')| \leq \frac{2\gamma||R||_\infty}{(1-\gamma)^2} \sup_{s,s'\in S, a\in A} \Bigg[D_S^{sa}(\phi,\phi') + D_Z^{s'a}(\psi,\psi') + \\ \frac{4}{\ln(\gamma^{-e})}\left(\frac{\sum_{s''\in S}|\phi_{ss''}^a - \phi_{ss''}'^a|}{(\mathcal{N}_\phi^{sa}+1)(\mathcal{N}_{\phi'}^{sa}+1)} + \frac{\sum_{z\in Z}|\psi_{s'z}^a - \psi_{s'z}'^a|}{(\mathcal{N}_\psi^{s'a}+1)(\mathcal{N}_{\psi'}^{s'a}+1)} \right) \Bigg].$$

Proof Using lemma 1, we have that:

$$\begin{aligned}
&|\alpha_t^{a,\alpha'}(s,\phi,\psi)-\alpha_t^{a,\alpha'}(s,\phi',\psi')| \\
= & \left|R(s,a)+\gamma\sum_{s'\in S}\sum_{z\in Z}\frac{\phi^a_{ss'}\psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi}\mathcal{N}^{s'a}_{\psi}}\alpha'(z)(s',\phi+\delta^a_{ss'},\psi+\delta^a_{s'z})\right. \\
&\left.-R(s,a)-\gamma\sum_{s'\in S}\sum_{z\in Z}\frac{\phi'^a_{ss'}\psi'^a_{s'z}}{\mathcal{N}^{sa}_{\phi'}\mathcal{N}^{s'a}_{\psi'}}\alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right| \\
= & \gamma\left|\sum_{s'\in S}\sum_{z\in Z}\left[\frac{\phi^a_{ss'}\psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi}\mathcal{N}^{s'a}_{\psi}}\alpha'(z)(s',\phi+\delta^a_{ss'},\psi+\delta^a_{s'z})-\frac{\phi'^a_{ss'}\psi'^a_{s'z}}{\mathcal{N}^{sa}_{\phi'}\mathcal{N}^{s'a}_{\psi'}}\alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right]\right| \\
= & \gamma\left|\sum_{s'\in S}\sum_{z\in Z}\left[\frac{\phi^a_{ss'}\psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi}\mathcal{N}^{s'a}_{\psi}}(\alpha'(z)(s',\phi+\delta^a_{ss'},\psi+\delta^a_{s'z})-\alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z}))\right.\right. \\
&\left.\left.-\left(\frac{\phi'^a_{ss'}\psi'^a_{s'z}}{\mathcal{N}^{sa}_{\phi'}\mathcal{N}^{s'a}_{\psi'}}-\frac{\phi^a_{ss'}\psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi}\mathcal{N}^{s'a}_{\psi}}\right)\alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right]\right| \\
\leq & \gamma\sum_{s'\in S}\sum_{z\in Z}\frac{\phi^a_{ss'}\psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi}\mathcal{N}^{s'a}_{\psi}}\left|\alpha'(z)(s',\phi+\delta^a_{ss'},\psi+\delta^a_{s'z})-\alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right| \\
&+\gamma\sum_{s'\in S}\sum_{z\in Z}\left|\frac{\phi'^a_{ss'}\psi'^a_{s'z}}{\mathcal{N}^{sa}_{\phi'}\mathcal{N}^{s'a}_{\psi'}}-\frac{\phi^a_{ss'}\psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi}\mathcal{N}^{s'a}_{\psi}}\right|\left|\alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right| \\
\leq & \gamma\sup_{s'\in S,z\in Z}\left|\alpha'(z)(s',\phi+\delta^a_{ss'},\psi+\delta^a_{s'z})-\alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right| \\
&+\frac{\gamma||R||_\infty}{1-\gamma}\sum_{s'\in S}\sum_{z\in Z}\left|\frac{\phi'^a_{ss'}\psi'^a_{s'z}}{\mathcal{N}^{sa}_{\phi'}\mathcal{N}^{s'a}_{\psi'}}-\frac{\phi^a_{ss'}\psi^a_{s'z}}{\mathcal{N}^{sa}_{\phi}\mathcal{N}^{s'a}_{\psi}}\right| \\
\leq & \gamma\sup_{s'\in S,z\in Z}\left|\alpha'(z)(s',\phi+\delta^a_{ss'},\psi+\delta^a_{s'z})-\alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right| \\
&+\frac{\gamma||R||_\infty}{1-\gamma}\left(D^{sa}_S(\phi',\phi)+\sup_{s'\in S}D^{s'a}_Z(\psi',\psi)\right).
\end{aligned}$$

The last inequality follows from lemma 3. Hence by taking the sup we get:

$$\begin{aligned}
&\sup_{\alpha_t\in\Gamma_t,s\in S}|\alpha_t(s,\phi,\psi)-\alpha_t(s,\phi',\psi')| \\
\leq & \gamma\sup_{s,s'\in S,a\in A,z\in Z,\alpha_{t-1}\in\Gamma_{t-1}}\left|\alpha_{t-1}(s',\phi+\delta^a_{ss'},\psi+\delta^a_{s'z})-\alpha_{t-1}(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right| \\
&+\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\left(D^{sa}_S(\phi',\phi)+D^{s'a}_Z(\psi',\psi)\right).
\end{aligned}$$

We notice that this inequality defines a recurrence. By unfolding it up to $t=1$ we get that:

$$\begin{aligned}
&\sup_{\alpha_t\in\Gamma_t,s\in S}|\alpha_t(s,\phi,\psi)-\alpha_t(s,\phi',\psi')| \\
\leq & \gamma^{t-1}\sup_{\alpha_1\in\Gamma_1,s'\in S,\Delta\in\mathcal{T},\Delta'\in\mathcal{O}|\ ||\Delta||_1=||\Delta'||_1=(t-1)}|\alpha_1(s',\phi+\Delta,\psi+\Delta')-\alpha_1(s',\phi'+\Delta,\psi'+\Delta')| \\
&+\frac{\gamma||R||_\infty}{1-\gamma}\sum_{i=1}^{t-2}\gamma^i\sup_{s,s'\in S,a\in A,\Delta\in\mathcal{T},\Delta'\in\mathcal{O}|\ ||\Delta||_1=||\Delta'||_1=i}\left(D^{sa}_S(\phi'+\Delta,\phi+\Delta)+D^{s'a}_Z(\psi'+\Delta',\psi+\Delta')\right) \\
&+\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\left(D^{sa}_S(\phi',\phi)+D^{s'a}_Z(\psi',\psi)\right).
\end{aligned}$$

Applying lemmas 7, 4 and 5 to the last term, we get that:

$$\begin{aligned}
&\sup_{\alpha_t\in\Gamma_t, s\in S}|\alpha_t(s,\phi,\psi)-\alpha_t(s,\phi',\psi')|\\
\leq\ & \frac{\gamma||R||_\infty}{1-\gamma}\sum_{i=1}^{t-2}\gamma^i \sup_{s,s'\in S,a\in A,\Delta\in\mathcal{T},\Delta'\in\mathcal{O}|\ ||\Delta||_1=||\Delta'||_1=i}\Bigg(D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)\\
&+\frac{2\mathcal{N}_\Delta^{sa}\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{(\mathcal{N}_\phi^{sa}+\mathcal{N}_\Delta^{sa})(\mathcal{N}_{\phi'}^{sa}+\mathcal{N}_\Delta^{sa})}+\frac{2\mathcal{N}_{\Delta'}^{s'a}\sum_{z\in Z}|\psi^a_{s'z}-\psi'^a_{s'z}|}{(\mathcal{N}_\psi^{s'a}+\mathcal{N}_{\Delta'}^{s'a})(\mathcal{N}_{\psi'}^{s'a}+\mathcal{N}_{\Delta'}^{s'a})}\Bigg)\\
&+\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\Big(D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)\Big)\\
=\ & \frac{\gamma||R||_\infty}{1-\gamma}\sum_{i=1}^{t-2}\gamma^{i/2} \sup_{s,s'\in S,a\in A,\Delta\in\mathcal{T},\Delta'\in\mathcal{O}|\ ||\Delta||_1=||\Delta'||_1=i}\Bigg(\gamma^{i/2}D_S^{sa}(\phi',\phi)+\gamma^{i/2}D_Z^{s'a}(\psi',\psi)\\
&+\frac{2\gamma^{i/2}\mathcal{N}_\Delta^{sa}\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{(\mathcal{N}_\phi^{sa}+\mathcal{N}_\Delta^{sa})(\mathcal{N}_{\phi'}^{sa}+\mathcal{N}_\Delta^{sa})}+\frac{2\gamma^{i/2}\mathcal{N}_{\Delta'}^{s'a}\sum_{z\in Z}|\psi^a_{s'z}-\psi'^a_{s'z}|}{(\mathcal{N}_\psi^{s'a}+\mathcal{N}_{\Delta'}^{s'a})(\mathcal{N}_{\psi'}^{s'a}+\mathcal{N}_{\Delta'}^{s'a})}\Bigg)\\
&+\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\Big(D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)\Big).
\end{aligned}$$

Now we notice that $\gamma^{i/2}\leq\gamma^{\mathcal{N}_\Delta^{sa}/2}$ since $||\Delta||_1=i$, and similarly $\gamma^{i/2}\leq\gamma^{\mathcal{N}_{\Delta'}^{sa}/2}$. Hence by applying lemma 6, we get that:

$$\begin{aligned}
&\sup_{\alpha_t\in\Gamma_t, s\in S}|\alpha_t(s,\phi,\psi)-\alpha_t(s,\phi',\psi')|\\
\leq\ & \frac{\gamma||R||_\infty}{1-\gamma}\sum_{i=1}^{t-2}\gamma^{i/2} \sup_{s,s'\in S,a\in A,\Delta\in\mathcal{T},\Delta'\in\mathcal{O}|\ ||\Delta||_1=||\Delta'||_1=i}\Bigg(D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)\\
&+\frac{4\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{\ln(\gamma^{-e})(\mathcal{N}_\phi^{sa}+\mathcal{N}_\Delta^{sa})(\mathcal{N}_{\phi'}^{sa}+\mathcal{N}_\Delta^{sa})}+\frac{4\sum_{z\in Z}|\psi^a_{s'z}-\psi'^a_{s'z}|}{\ln(\gamma^{-e})(\mathcal{N}_\psi^{s'a}+\mathcal{N}_{\Delta'}^{s'a})(\mathcal{N}_{\psi'}^{s'a}+\mathcal{N}_{\Delta'}^{s'a})}\Bigg)\\
&+\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\Big(D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)\Big)\\
\leq\ & \frac{\gamma||R||_\infty}{1-\gamma}\sum_{i=1}^{t-2}\gamma^{i/2}\sup_{s,s'\in S,a\in A}\Bigg(D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)+\frac{4\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{\ln(\gamma^{-e})(\mathcal{N}_\phi^{sa}+1)(\mathcal{N}_{\phi'}^{sa}+1)}\\
&+\frac{4\sum_{z\in Z}|\psi^a_{s'z}-\psi'^a_{s'z}|}{\ln(\gamma^{-e})(\mathcal{N}_\psi^{s'a}+1)(\mathcal{N}_{\psi'}^{s'a}+1)}\Bigg)+\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\Big(D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)\Big)\\
\leq\ & \Big(\sum_{i=0}^{t-2}\gamma^{i/2}\Big)\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\Bigg[D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)\\
&+\frac{4}{\ln(\gamma^{-e})}\Bigg(\frac{\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{(\mathcal{N}_\phi^{sa}+1)(\mathcal{N}_{\phi'}^{sa}+1)}+\frac{\sum_{z\in Z}|\psi^a_{s'z}-\psi'^a_{s'z}|}{(\mathcal{N}_\psi^{s'a}+1)(\mathcal{N}_{\psi'}^{s'a}+1)}\Bigg)\Bigg]\\
\leq\ & \Big(\sum_{i=0}^{\infty}\gamma^{i/2}\Big)\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\Bigg[D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)\\
&+\frac{4}{\ln(\gamma^{-e})}\Bigg(\frac{\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{(\mathcal{N}_\phi^{sa}+1)(\mathcal{N}_{\phi'}^{sa}+1)}+\frac{\sum_{z\in Z}|\psi^a_{s'z}-\psi'^a_{s'z}|}{(\mathcal{N}_\psi^{s'a}+1)(\mathcal{N}_{\psi'}^{s'a}+1)}\Bigg)\Bigg]\\
=\ & \frac{1+\sqrt{\gamma}}{1-\gamma}\frac{\gamma||R||_\infty}{1-\gamma}\sup_{s,s'\in S,a\in A}\Bigg[D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)+\frac{4}{\ln(\gamma^{-e})}\Bigg(\frac{\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{(\mathcal{N}_\phi^{sa}+1)(\mathcal{N}_{\phi'}^{sa}+1)}+\frac{\sum_{z\in Z}|\psi^a_{s'z}-\psi'^a_{s'z}|}{(\mathcal{N}_\psi^{s'a}+1)(\mathcal{N}_{\psi'}^{s'a}+1)}\Bigg)\Bigg]\\
\leq\ & \frac{2\gamma||R||_\infty}{(1-\gamma)^2}\sup_{s,s'\in S,a\in A}\Bigg[D_S^{sa}(\phi',\phi)+D_Z^{s'a}(\psi',\psi)+\frac{4}{\ln(\gamma^{-e})}\Bigg(\frac{\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{(\mathcal{N}_\phi^{sa}+1)(\mathcal{N}_{\phi'}^{sa}+1)}+\frac{\sum_{z\in Z}|\psi^a_{s'z}-\psi'^a_{s'z}|}{(\mathcal{N}_\psi^{s'a}+1)(\mathcal{N}_{\psi'}^{s'a}+1)}\Bigg)\Bigg].
\end{aligned}$$

■

Lemma 8 *Given $\phi\in\mathcal{T}$, $s\in S$, $a\in A$, then for all $\Delta\in\mathcal{T}$, $\frac{\sum_{s'\in S}|\phi^a_{ss'}-(\phi^a_{ss'}+\Delta^a_{ss'})|}{(\mathcal{N}_\phi^{sa}+1)(\mathcal{N}_\phi^{sa}+\mathcal{N}_\Delta^{sa}+1)}\leq\frac{1}{\mathcal{N}_\phi^{sa}+1}$.*

Proof

$$\begin{aligned} &\frac{\sum_{s'\in S}|\phi^a_{ss'}-(\phi^a_{ss'}+\Delta^a_{ss'})|}{(\mathcal{N}^{sa}_\phi+1)(\mathcal{N}^{sa}_\phi+\mathcal{N}^{sa}_\Delta+1)} \\ =\; & \frac{\sum_{s'\in S}\Delta^a_{ss'}}{(\mathcal{N}^{sa}_\phi+1)(\mathcal{N}^{sa}_\phi+\mathcal{N}^{sa}_\Delta+1)} \\ =\; & \frac{1}{\mathcal{N}^{sa}_\phi+1}\left(\frac{\mathcal{N}^{sa}_\Delta}{\mathcal{N}^{sa}_\Delta+\mathcal{N}^{sa}_\phi+1}\right). \end{aligned}$$

The term $\frac{\mathcal{N}^{sa}_\Delta}{\mathcal{N}^{sa}_\Delta+\mathcal{N}^{sa}_\phi+1}$ is monotonically increasing and converge to 1 as $\mathcal{N}^{sa}_\Delta \to \infty$. Thus the lemma follows. ■

Corollary 1 *Given $\varepsilon > 0$, $\phi \in \mathcal{T}$, $s \in S$, $a \in A$, if $\mathcal{N}^{sa}_\phi > \frac{1}{\varepsilon} - 1$ then for all $\Delta \in \mathcal{T}$ we have that* $\frac{\sum_{s'\in S}|\phi^a_{ss'}-(\phi^a_{ss'}+\Delta^a_{ss'})|}{(\mathcal{N}^{sa}_\phi+1)(\mathcal{N}^{sa}_\phi+\mathcal{N}^{sa}_\Delta+1)} < \varepsilon$.

Proof According to lemma 8, we know that for all $\Delta \in \mathcal{T}$, we have that $\frac{\sum_{s'\in S}|\phi^a_{ss'}-(\phi^a_{ss'}+\Delta^a_{ss'})|}{(\mathcal{N}^{sa}_\phi+1)(\mathcal{N}^{sa}_\phi+\mathcal{N}^{sa}_\Delta+1)} \leq \frac{1}{\mathcal{N}^{sa}_\phi+1}$. Hence if $\mathcal{N}^{sa}_\phi > \frac{1}{\varepsilon} - 1$, then $\frac{1}{\mathcal{N}^{sa}_\phi+1} < \varepsilon$. ■

Lemma 9 *Given $\psi \in \mathcal{O}$, $s \in S$, $a \in A$, then for all $\Delta \in \mathcal{O}$,* $\frac{\sum_{z\in Z}|\psi^a_{sz}-(\psi^a_{sz}+\Delta^a_{sz})|}{(\mathcal{N}^{sa}_\psi+1)(\mathcal{N}^{sa}_\psi+\mathcal{N}^{sa}_\Delta+1)} \leq \frac{1}{\mathcal{N}^{sa}_\psi+1}$.

Proof Same proof as lemma 8. ■

Corollary 2 *Given $\varepsilon > 0$, $\psi \in \mathcal{O}$, $s \in S$, $a \in A$, if $\mathcal{N}^{sa}_\psi > \frac{1}{\varepsilon} - 1$ then for all $\Delta \in \mathcal{O}$ we have that* $\frac{\sum_{z\in Z}|\psi^a_{sz}-(\psi^a_{sz}+\Delta^a_{sz})|}{(\mathcal{N}^{sa}_\psi+1)(\mathcal{N}^{sa}_\psi+\mathcal{N}^{sa}_\Delta+1)} < \varepsilon$

Proof Same proof as corollary 1, but using lemma 9 instead. ■

Theorem 4 *Given any $\varepsilon > 0$ and $(s,\phi,\psi) \in S'$ such that $\exists a \in A, \exists s' \in S$, $\mathcal{N}^{s'a}_\phi > N^\varepsilon_S$ or $\mathcal{N}^{s'a}_\psi > N^\varepsilon_Z$, then $\exists(s,\phi',\psi') \in S'$ such that $\forall a \in A, \forall s' \in S$, $\mathcal{N}^{s'a}_{\phi'} \leq N^\varepsilon_S$, $\mathcal{N}^{s'a}_{\psi'} \leq N^\varepsilon_Z$ and $|\alpha_t(s,\phi,\psi) - \alpha_t(s,\phi',\psi')| < \varepsilon$ holds for all t and $\alpha_t \in \Gamma_t$.*

Proof Consider an arbitrary $\varepsilon > 0$. We first find a bound on $\mathcal{N}^{sa}_\phi$ and $\mathcal{N}^{sa}_\psi$ such that any vector with higher counts is within ε distance of another vector with lower counts. Let's define $\varepsilon' = \frac{\varepsilon(1-\gamma)^2}{8\gamma||R||_\infty}$ and $\varepsilon'' = \frac{\varepsilon(1-\gamma)^2\ln(\gamma^{-e})}{32\gamma||R||_\infty}$. According to corollary 1, we have that for any $\phi \in \mathcal{T}$ such that $\mathcal{N}^{sa}_\phi > \frac{1}{\varepsilon''} - 1$, then for all $\phi' \in \mathcal{T}$ such that there exists a $\Delta \in \mathcal{T}$ where $\phi' = \phi + \Delta$, then $\frac{\sum_{s''\in S}|\phi^a_{ss''}-\phi'^a_{ss''}|}{(\mathcal{N}^{sa}_\phi+1)(\mathcal{N}^{sa}_{\phi'}+1)} < \varepsilon''$. Hence we want to find an N such that given $\phi \in \mathcal{T}$ with $\mathcal{N}^{sa}_\phi > N$, there exists a $\phi' \in \mathcal{T}$ such that $\mathcal{N}^{sa}_{\phi'} \leq N$, $D^{sa}_S(\phi,\phi') < \varepsilon'$ and exists a $\Delta \in \mathcal{T}$ such that $\phi = \phi' + \Delta$. Let's consider an arbitrary ϕ such that $\mathcal{N}^{sa}_\phi > N$. We can construct a new vector ϕ' as follows, for all s' define $\phi'^a_{ss'} = \left\lfloor \frac{N\phi^a_{ss'}}{\mathcal{N}^{sa}_\phi} \right\rfloor$ and for all other $a' \neq a, s'' \neq s$, define $\phi'^{a'}_{s''s'} = \phi^{a'}_{s''s'}$ for all s'. Clearly, $\phi' \in \mathcal{T}$, such that $N - |S| \leq \mathcal{N}^{sa}_{\phi'} \leq N$. Moreover, we have that $\phi'^{a'}_{s's''} \leq \phi^{a'}_{s's''}$ for all s', a', s'', and thus there exists a $\Delta \in \mathcal{T}$ such that $\phi = \phi' + \Delta$.

Furthermore, from its construction, we know that $\forall s'$, $\left|\frac{\phi'^a_{ss'}}{\mathcal{N}^{sa}_{\phi'}} - \frac{\phi^a_{ss'}}{\mathcal{N}^{sa}_{\phi}}\right| \leq \frac{1}{\mathcal{N}^{sa}_{\phi'}}$. Hence it is clear from this that $D^{sa}_S(\phi,\phi') \leq \frac{|S|}{N-|S|}$. Thus, if we want $D^{sa}_S(\phi,\phi') < \varepsilon'$, we just need to take $N > \frac{|S|(1+\varepsilon')}{\varepsilon'}$. Since we also want $N > \frac{1}{\varepsilon''} - 1$, let's just define $N_S = \max\left(\frac{|S|(1+\varepsilon')}{\varepsilon'}, \frac{1}{\varepsilon''} - 1\right)$. $N_S = N^{\varepsilon}_S$, as defined in Section 4, will be our bound on $\mathcal{N}^{sa}_{\phi}$ such that, as we have just showed, for any $\phi \in \mathcal{T}$ such that $\mathcal{N}^{sa}_{\phi} > N_S$, we can find a $\phi' \in \mathcal{T}$ such that $\mathcal{N}^{sa}_{\phi'} \leq N_S$, $D^{sa}_S(\phi,\phi') < \varepsilon'$ and $\frac{\sum_{s''\in S}|\phi^a_{ss''} - \phi'^a_{ss''}|}{(\mathcal{N}^{sa}_{\phi}+1)(\mathcal{N}^{sa}_{\phi'}+1)} < \varepsilon''$. Similarly, since we have a similar corollary (corollary 1) for the observation counts ψ, we can proceed in the same way and define $N_Z = \max\left(\frac{|Z|(1+\varepsilon')}{\varepsilon'}, \frac{1}{\varepsilon''} - 1\right)$, such that for any $\psi \in O$ such that $\mathcal{N}^{sa}_{\psi} > N_Z$, we can find a $\psi' \in O$ such that $\mathcal{N}^{sa}_{\psi'} \leq N_Z$, $D^{sa}_Z(\psi,\psi') < \varepsilon'$ and $\frac{\sum_{z\in Z}|\psi^a_{sz} - \psi'^a_{sz}|}{(\mathcal{N}^{sa}_{\psi}+1)(\mathcal{N}^{sa}_{\psi'}+1)} < \varepsilon''$. $N_Z = N^{\varepsilon}_Z$ as we have defined in Section 4.

Now let $\tilde{S} = \{(s,\phi,\psi) \in S' | \forall s' \in S, a \in A, N^{s'a}_{\phi} \leq N_S \ \& \ N^{s'a}_{\psi} \leq N_Z\}$ and consider an arbitrary $(s,\phi,\psi) \in S'$. For any $s' \in S$, $a \in A$, such that $\mathcal{N}^{s'a}_{\phi} > N_S$, there exists a $\phi' \in \mathcal{T}$ such that $\mathcal{N}^{s'a}_{\phi'} \leq N_S$, $D^{s'a}_S(\phi,\phi') < \varepsilon'$ and $\frac{\sum_{s''\in S}|\phi^a_{s's''} - \phi'^a_{s's''}|}{(\mathcal{N}^{s'a}_{\phi}+1)(\mathcal{N}^{s'a}_{\phi'}+1)} < \varepsilon''$ (as we have just showed above). Thus let's define $\tilde{\phi}^a_{s's''} = \phi'^a_{s's''}$ for all $s'' \in S$. For any $s' \in S$, $a \in A$, such that $\mathcal{N}^{s'a}_{\phi} \leq N_S$, just set $\tilde{\phi}^a_{s's''} = \phi^a_{s's''}$, $\forall s'' \in S$. Similarly, for any $s' \in S$, $a \in A$, such that $\mathcal{N}^{s'a}_{\psi} > N_Z$, there exists a $\psi' \in O$ such that $\mathcal{N}^{s'a}_{\psi'} \leq N_Z$, $D^{s'a}_Z(\psi,\psi') < \varepsilon'$ and $\frac{\sum_{z\in Z}|\psi^a_{s'z} - \psi'^a_{s'z}|}{(\mathcal{N}^{s'a}_{\psi}+1)(\mathcal{N}^{s'a}_{\psi'}+1)} < \varepsilon''$ (as we have just showed above). Thus let's define $\tilde{\psi}^a_{s's''} = \psi'^a_{s's''}$ for all $s'' \in S$. For any $s' \in S$, $a \in A$, such that $\mathcal{N}^{s'a}_{\psi} \leq N_Z$, just set $\tilde{\psi}^a_{s's''} = \psi^a_{s's''}$ $\forall s'' \in S$. Now it is clear from this construction that $(s,\tilde{\phi},\tilde{\psi}) \in \tilde{S}$. By Theorem 3, for any t, $\sup_{\alpha_t\in\Gamma_t, s\in S}|\alpha_t(s,\phi,\psi) - \alpha_t(s,\tilde{\phi},\tilde{\psi})| \leq \frac{2\gamma||R||_\infty}{(1-\gamma)^2} \sup_{s,s'\in S, a\in A}\left[D^{s,a}_S(\phi,\tilde{\phi}) + D^{s',a}_Z(\psi,\tilde{\psi}) + \frac{4}{\ln(\gamma^{-e})}\left(\frac{\sum_{s''\in S}|\phi^a_{ss''} - \tilde{\phi}^a_{ss''}|}{(\mathcal{N}^{sa}_{\phi}+1)(\mathcal{N}^{sa}_{\tilde{\phi}}+1)} + \frac{\sum_{z\in Z}|\psi^a_{s'z} - \tilde{\psi}^a_{s'z}|}{(\mathcal{N}^{s'a}_{\psi}+1)(\mathcal{N}^{s'a}_{\tilde{\psi}}+1)}\right)\right] < \frac{2\gamma||R||_\infty}{(1-\gamma)^2}\left[\varepsilon' + \varepsilon' + \frac{4}{\ln(\gamma^{-e})}(\varepsilon'' + \varepsilon'')\right] = \varepsilon$. ■

Theorem 5 *Given any $\varepsilon > 0$, $(s,\phi,\psi) \in S'$ and $\alpha_t \in \Gamma_t$ computed from the infinite BAPOMDP. Let $\tilde{\alpha}_t$ be the α-vector representing the same conditional plan as α_t but computed with the finite BAPOMDP $(\tilde{S}_\varepsilon, A, Z, \tilde{T}_\varepsilon, \tilde{O}_\varepsilon, \tilde{R}_\varepsilon, \gamma)$, then $|\tilde{\alpha}_t(\mathcal{P}_\varepsilon(s,\phi,\psi)) - \alpha_t(s,\phi,\psi)| < \frac{\varepsilon}{1-\gamma}$.*

Proof Let $(s,\phi',\psi') = \mathcal{P}_\varepsilon(s,\phi,\psi)$.

$$\begin{array}{ll} & |\tilde{\alpha}_t(\mathcal{P}_\varepsilon(s,\phi,\psi)) - \alpha_t(s,\phi,\psi)| \\ \leq & |\tilde{\alpha}_t(s,\phi',\psi') - \alpha_t(s,\phi',\psi')| + |\alpha_t(s,\phi',\psi') - \alpha_t(s,\phi,\psi)| \\ < & |\tilde{\alpha}_t(s,\phi',\psi') - \alpha_t(s,\phi',\psi')| + \varepsilon \quad \text{(by Theorem 4)} \\ = & |\gamma \sum_{z\in Z}\sum_{s'\in S} T^{sas'}_{\phi'} O^{s'az}_{\psi'}\left[\tilde{\alpha}'(z)(\mathcal{P}_\varepsilon(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})) - \alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right]| + \varepsilon \\ \leq & \gamma \sum_{z\in Z}\sum_{s'\in S} T^{sas'}_{\phi'} O^{s'az}_{\psi'}\left|\tilde{\alpha}'(z)(\mathcal{P}_\varepsilon(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})) - \alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right| + \varepsilon \\ \leq & \gamma \sup_{z\in Z, s'\in S}\left|\tilde{\alpha}'(z)(\mathcal{P}_\varepsilon(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})) - \alpha'(z)(s',\phi'+\delta^a_{ss'},\psi'+\delta^a_{s'z})\right| + \varepsilon \\ \leq & \gamma \sup_{\alpha_{t-1}\in\Gamma_{t-1}, (s',\phi'',\psi'')\in S'}|\tilde{\alpha}_{t-1}(\mathcal{P}_\varepsilon(s',\phi'',\psi'')) - \alpha_{t-1}(s',\phi'',\psi'')| + \varepsilon. \end{array}$$

Thus, we have that:

$$\begin{array}{ll} & \sup_{\alpha_t\in\Gamma_t, \sigma\in S'}|\tilde{\alpha}_t(\mathcal{P}_\varepsilon(\sigma)) - \alpha_t(\sigma)| \\ < & \gamma \sup_{\alpha_{t-1}\in\Gamma_{t-1}, \sigma'\in S'}|\tilde{\alpha}_{t-1}(\mathcal{P}_\varepsilon(\sigma')) - \alpha_{t-1}(\sigma')| + \varepsilon. \end{array}$$

This defines a recurrence. By unfolding it up to $t = 1$, where $\forall\sigma \in S'$, $\tilde{\alpha}_1(\mathcal{P}_\varepsilon(\sigma)) = \alpha_1(\sigma)$, we get that $\sup_{\alpha_t \in \Gamma_t, \sigma \in S'} |\tilde{\alpha}_t(\mathcal{P}_\varepsilon(\sigma)) - \alpha_t(\sigma)| < \varepsilon \sum_{i=0}^{t-2} \gamma^i$. Hence for all t, this is lower than $\frac{\varepsilon}{1-\gamma}$. ■

Theorem 6 *Given any $\varepsilon > 0$, and any horizon t, let $\tilde{\pi}_t$ be the optimal t-step policy computed from the finite POMDP $(\tilde{S}_\varepsilon, A, Z, \tilde{T}_\varepsilon, \tilde{O}_\varepsilon, \tilde{R}_\varepsilon, \gamma)$, then for any initial belief b the value of executing policy $\tilde{\pi}_t$ in the BAPOMDP $V_{\tilde{\pi}_t}(b) \geq V^*(b) - 2\frac{\varepsilon}{1-\gamma}$.*

Proof Pick any starting belief b in the BAPOMDP. Let α^* denote the optimal t-step condition plan in the BAPOMDP for b: $\alpha^* = \operatorname{argmax}_{\alpha \in \Gamma_t} \sum_{(s,\phi,\psi)} b(s,\phi,\psi)\alpha(s,\phi,\psi)$, such that the value of this optimal conditional plan is $\sum_{(s,\phi,\psi)} b(s,\phi,\psi)\alpha^*(s,\phi,\psi) = V^*(b)$. Denote $\tilde{\alpha}^*$ the corresponding α-vector representing the same t-step conditional plan in the finite POMDP approximation.

Now let $\tilde{\alpha}' = \operatorname{argmax}_{\tilde{\alpha} \in \tilde{\Gamma}_t} \sum_{(s,\phi,\psi)} b(s,\phi,\psi)\tilde{\alpha}(\mathcal{P}_\varepsilon(s,\phi,\psi))$ be the optimal t-step conditional plan in the finite POMDP approximation if we start in belief b. This conditional plan represents exactly what the policy $\tilde{\pi}_t$ would do over t-steps starting in b. Denote α' the corresponding α-function in the BAPOMDP representing the same t-step conditional plan. Then the value of executing $\tilde{\pi}_t$ starting in b in the BAPOMDP is $V_{\tilde{\pi}_t}(b) = \sum_{(s,\phi,\psi)} b(s,\phi,\psi)\alpha'(s,\phi,\psi)$. Using Theorem 5, this value is lower bounded as follows:

$$\begin{array}{ll} V_{\tilde{\pi}_t}(b) & \\ = & \sum_{(s,\phi,\psi)} b(s,\phi,\psi)\alpha'(s,\phi,\psi) \\ \geq & \sum_{(s,\phi,\psi)} b(s,\phi,\psi)\tilde{\alpha}'(\mathcal{P}_\varepsilon(s,\phi,\psi)) - \frac{\varepsilon}{1-\gamma} \\ \geq & \sum_{(s,\phi,\psi)} b(s,\phi,\psi)\tilde{\alpha}^*(\mathcal{P}_\varepsilon(s,\phi,\psi)) - \frac{\varepsilon}{1-\gamma} \\ \geq & \sum_{(s,\phi,\psi)} b(s,\phi,\psi)\alpha^*(\mathcal{P}_\varepsilon(s,\phi,\psi)) - 2\frac{\varepsilon}{1-\gamma} \\ = & V^*(b) - 2\frac{\varepsilon}{1-\gamma}. \end{array}$$

■

References

J. Asmuth, L. Li, M. Littman, A. Nouri, and D. Wingate. A bayesian sampling approach to exploration in reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

P. Auer and R. Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Neural Information Processing Systems (NIPS)*, volume 19, pages 49–56, 2006.

P. Auer, T. Jaksch, and R. Ortner. Near-optimal regret bounds for reinforcement learning. In *Neural Information Processing Systems (NIPS)*, volume 21, 2009.

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research (JAIR)*, 15:319–350, 2001.

R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

R. I. Brafman and M. Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 3:213–231, 2003.

G. Casella and R. Berger. *Statistical Inference*. Duxbury Resource Center, 2001.

P. S. Castro and D. Precup. Using linear programming for bayesian exploration in markov decision processes. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2437–2442, 2007.

R. Dearden, N. Friedman, and S. J. Russell. Bayesian Q-learning. In *AAAI Conference on Artificial Intelligence*, pages 761–768, 1998.

R. Dearden, N. Friedman, and D. Andre. Model based bayesian exploration. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 150–159, 1999.

E. Delage and S. Mannor. Percentile optimization in uncertain mdps with application to efficient exploration. In *International Conference on Machine Learning (ICML)*, 2007.

F. Doshi, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. In *International Conference on Machine Learning*, pages 256–263. ACM, 2008.

F. Doshi-Velez. The infinite partially observable markov decision process. In *Neural Information Processing Systems (NIPS)*, volume 22, 2010.

A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods In Practice*. Springer, 2001.

M. Duff. Monte-Carlo algorithms for the improvement of finite-state stochastic controllers: Application to bayes-adaptive Markov decision processes. In *International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2001.

M. Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts Amherst, Amherst, MA, 2002.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The gaussian process approach to temporal difference learning. In *International Conference on Machine Learning (ICML)*, pages 154–161, 2003.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. In *International Conference on Machine learning (ICML)*, pages 201–208, 2005.

A. A. Feldbaum. Dual control theory, parts i and ii. *Automation and Remote Control*, 21:874–880 and 1033–1039, 1961.

N. M. Filatov and H. Unbehauen. Survey of adaptive dual control methods. In *IEEE Control Theory and Applications*, volume 147, pages 118–128, 2000.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In *Neural Information Processing Systems (NIPS)*, volume 19, pages 457–464, 2007a.

M. Ghavamzadeh and Y. Engel. Bayesian actor-critic algorithms. In *International Conference on Machine Learning (ICML)*, pages 297–304, 2007b.

A. Greenfield and A. Brockwell. Adaptive control of nonlinear stochastic systems by particle filtering. In *International Conference on Control and Automation (ICCA)*, pages 887–890, 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.

M. Hutter. *Universal Artificial Intelligence*. Springer, 2005.

R. Jaulmes, J. Pineau, and D. Precup. Active learning in partially observable markov decision processes. *European Conference on Machine Learning*, pages 601–608, 2005.

E. T. Jaynes. Prior probabilities. *IEEE Transactions on Systems Science and Cybernetics*, 4:227–241, 1968.

H. Jeffreys. *Theory of Probability*. Oxford University Press, 1961.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *International Conference on Machine Learning (ICML)*, pages 260–268, 1998.

M. J. Kearns, Y. Mansour, and A. Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1324–1331, 1999.

J. Zico Kolter and Andrew Y. Ng. Near-bayesian exploration in polynomial time. In *International Conference on Machine Learning (ICML)*, 2009.

M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. In *Neural Information Processing Systems (NIPS)*, volume 14, pages 1555–1561, 2002.

A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.

S. Paquet, L. Tobin, and B. Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 970–977, 2005.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: an anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032, 2003.

P. Poupart and N. Vlassis. Model-based bayesian reinforcement learning in partially observable domains. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008.

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. In *International Conference on Machine learning (ICML)*, pages 697–704, 2006.

R. Ravikanth, S.P. Meyn, and L.J. Brown. Bayesian adaptive control of time varying systems. In *IEEE Conference on Decision and Control*, pages 705–709, 1992.

S. Ross, B. Chaib-draa, and J. Pineau. Bayes-adaptive POMDPs. In *Neural Information Processing Systems (NIPS)*, volume 20, pages 1225–1232, 2008a.

S. Ross, B. Chaib-draa, and J. Pineau. Bayesian reinforcement learning in continuous POMDPs. In *International Conference on Robotics and Automation (ICRA)*, 2008b.

S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 32:663–704, 2008c.

I. Rusnak. Optimal adaptive control of uncertain stochastic discrete linear systems. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 4521–4526, 1995.

D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Neural Information Processing Systems (NIPS)*, 2010.

R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, Sep/Oct 1973.

T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 520–527, 2004.

E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

M. T. J. Spaan and N. Vlassis. Perseus: randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 24:195–220, 2005.

A. L. Strehl and M. L. Littman. A theoretical analysis of model-based interval estimation. In *International Conference on Machine learning (ICML)*, pages 856–863, 2005.

M. Strens. A bayesian framework for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

C. Szepesvari. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.

A. Tewari and P. Bartlett. Optimistic linear programming gives logarithmic regret for irreducible MDPs. In *Neural Information Processing Systems (NIPS)*, volume 20, pages 1505–1512, 2008.

J. Veness, K. S. Ng, M. Hutter, W. Uther, and D. Silver. A monte-carlo aixi approximation. *Journal of Artificial Intelligence Research (JAIR)*, 2011.

T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *International Conference on Machine learning (ICML)*, pages 956–963, 2005.

O. Zane. Discrete-time bayesian adaptive control problems with complete information. In *IEEE Conference on Decision and Control*, pages 2748–2749, 1992.

Journal of Machine Learning Research 12 (2011) 1771-1812 Submitted 9/10; Revised 2/11; Published 5/11

Learning Latent Tree Graphical Models

Myung Jin Choi MYUNGJIN@MIT.EDU
Stochastic Systems Group
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Vincent Y. F. Tan VTAN@WISC.EDU
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706

Animashree Anandkumar A.ANANDKUMAR@UCI.EDU
Center for Pervasive Communications and Computing
Electrical Engineering and Computer Science
University of California, Irvine
Irvine, CA 92697

Alan S. Willsky WILLSKY@MIT.EDU
Stochastic Systems Group
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Editor: Marina Meilă

Abstract

We study the problem of learning a latent tree graphical model where samples are available only from a subset of variables. We propose two consistent and computationally efficient algorithms for learning *minimal* latent trees, that is, trees without any redundant hidden nodes. Unlike many existing methods, the observed nodes (or variables) are not constrained to be leaf nodes. Our algorithms can be applied to both discrete and Gaussian random variables and our learned models are such that all the observed and latent variables have the same domain (state space). Our first algorithm, *recursive grouping*, builds the latent tree recursively by identifying sibling groups using so-called information distances. One of the main contributions of this work is our second algorithm, which we refer to as *CLGrouping*. CLGrouping starts with a pre-processing procedure in which a tree over the observed variables is constructed. This global step groups the observed nodes that are likely to be close to each other in the true latent tree, thereby guiding subsequent recursive grouping (or equivalent procedures such as neighbor-joining) on much smaller subsets of variables. This results in more accurate and efficient learning of latent trees. We also present regularized versions of our algorithms that learn latent tree approximations of arbitrary distributions. We compare the proposed algorithms to other methods by performing extensive numerical experiments on various latent tree graphical models such as hidden Markov models and star graphs. In addition, we demonstrate the applicability of our methods on real-world data sets by modeling the dependency structure of monthly stock returns in the S&P index and of the words in the 20 newsgroups data set.

Keywords: graphical models, Markov random fields, hidden variables, latent tree models, structure learning

1. Introduction

The inclusion of latent variables in modeling complex phenomena and data is a well-recognized and a valuable construct in a variety of applications, including bio-informatics and computer vision, and the investigation of machine-learning methods for models with latent variables is a substantial and continuing direction of research.

There are three challenging problems in learning a model with latent variables: learning the number of latent variables; inferring the structure of how these latent variables relate to each other and to the observed variables; and estimating the parameters characterizing those relationships. Issues that one must consider in developing a new learning algorithm include developing tractable methods; incorporating the tradeoff between the fidelity to the given data and generalizability; deriving theoretical results on the performance of such algorithms; and studying applications that provide clear motivation and contexts for the models so learned.

One class of models that has received considerable attention in the literature is the class of *latent tree models*, that is, graphical models Markov on trees, in which variables at some nodes represent the original (observed) variables of interest while others represent the latent variables. The appeal of such models for computational tractability is clear: with a tree-structured model describing the statistical relationships, inference—processing noisy observations of some or all of the original variables to compute the estimates of all variables—is straightforward and scalable. Although the class of tree-structured models, with or without latent variables, is a constrained one, there are interesting applications that provide strong motivation for the work presented here. In particular, a very active avenue of research in computer vision is the use of context—for example, the nature of a scene to aid the reliable recognition of objects (and at the same time to allow the recognition of particular objects to assist in recognizing the scene). For example, if one knows that an image is that of an office, then one might expect to find a desk, a monitor on that desk, and perhaps a computer mouse. Hence if one builds a model with a latent variable representing that context ("office") and uses simple, noisy detectors for different object types, one would expect that the detection of a desk would support the likelihood that one is looking at an office and through that enhance the reliability of detecting smaller objects (monitors, keyboards, mice, etc.). Work along these lines, including by some of the authors of this paper (Parikh and Chen, 2007; Choi et al., 2010), show the promise of using tree-based models of context.

This paper considers the problem of learning tree-structured latent models. If all variables are observed in the tree under consideration, then the well-known algorithm of Chow and Liu (1968) provides a tractable algorithm for performing maximum likelihood (ML) estimation of the tree structure. However, if not all variables are observed, that is, for *latent* tree models, then ML estimation is NP-hard (Roch, 2006). This has motivated a number of investigations of other tractable methods for learning such trees as well as theoretical guarantees on performance. Our work represents a contribution to this area of investigation.

There are three main contributions in our paper. Firstly, by adopting a statistical distance-based framework, we develop two new algorithms for the learning of latent trees—recursive grouping and CLGrouping, which apply equally well to discrete and Gaussian models. Secondly, we provide consistency guarantees (both structural and parametric) as well as very favorable computational and sample complexity characterizations for both of our algorithms. Thirdly, through extensive numerical experiments on both synthetic and real-world data, we demonstrate the superiority of our approach for a wide variety of models ranging from ones with very large tree diameters (e.g., hidden Markov models (HMMs)) to star models and complete trees.[1]

Our first algorithm, which we refer to as *recursive grouping*, constructs a latent tree in a bottom-up fashion, grouping nodes into sibling groups that share the same parent node, recursively at each level of the resulting hierarchy (and allowing for some of the observed variables to play roles at arbitrary levels in the resulting hierarchy). Our second algorithm, *CLGrouping* first implements a global construction step, namely producing the Chow-Liu tree for the observed variables without any hidden nodes. This global step then provides guidance for groups of observed nodes that are likely to be topologically close to each other in the latent tree, thereby guiding subsequent recursive grouping or neighbor-joining (Saitou and Nei, 1987) computations. Each of these algorithms is consistent and has excellent sample and computational complexity.[2]

As Pearl (1988) points out, the identification of latent tree models has some built-in ambiguity, as there is an entire equivalence class of models in the sense that when all latent variables are marginalized out, each model in this class yields the same joint distribution over the observed variables. For example, we can take any such latent model and add another hidden variable as a leaf node connected to only one other (hidden or observed) node. Hence, much as one finds in fields such as state space dynamic systems (e.g., Luenberger, 1979, Section 8), there is a notion of minimality that is required here, and our results are stated in terms of consistent learning of such minimal latent models.

1.1 Related Work

The relevant literature on learning latent models is vast and in this section, we summarize the main lines of research in this area.

The classical *latent cluster models* (LCM) consider multivariate distributions in which there exists only *one* latent variable and each state of that variable corresponds to a cluster in the data (Lazarsfeld and Henry, 1968). Hierarchical latent class (HLC) models (Zhang and Kočka, 2004; Zhang, 2004; Chen et al., 2008) generalize these models by allowing multiple latent variables. HLC allows latent variables to have different number of states, but assume that all observed nodes are at the leaves of the tree. Their learning algorithm is based on a greedy approach of making one local move at a time (e.g., introducing one hidden node, or replacing an edge), which is computationally expensive and does not have consistency guarantees. A greedy learning algorithm for HLC called BIN is proposed in Harmeling and Williams (2010), which is computationally more efficient. In addition, Silva et al. (2006) considered the learning of directed latent models using so-called tetrad constraints, and there have also been attempts to tailor the learning of latent tree models in order to perform approximate inference accurately and efficiently downstream (Wang et al., 2008). In all these works, the latent variables can have different state spaces, but the observed nodes are required to be leaves of the tree. In contrast, we fix the state space of each hidden node, but allow the possibility that some observed nodes are internal nodes (non-leaves). This assumption leads to an identifiable model, and we provide algorithms with consistency guarantees which can recover the correct structure under mild conditions. In contrast, the works in Zhang and Kočka (2004);

1. A tree is called a *complete* k-ary tree (or k-complete tree), if all its internal nodes have degree k and there exists one node (commonly referred as the root node) that has the exactly same distance to all leaf nodes.
2. As we will see, depending on the true latent tree model, one or the other of these may be more efficient. Roughly speaking, for smaller diameter graphs (such as the star), recursive grouping is faster, and for larger diameter graphs (such as an HMM), CLgrouping is more efficient.

Zhang (2004); Chen et al. (2008); Harmeling and Williams (2010) do not provide such consistency guarantees.

Many authors also propose reconstructing latent trees using the expectation maximization (EM) algorithm (Elidan and Friedman, 2005; Kemp and Tenenbaum, 2008). However, as with all other EM-based methods, these approaches depend on the initialization and suffer from the possibility of being trapped in local optima and thus no consistency guarantees can be provided. At each iteration, a large number of candidate structures need to be evaluated, so these methods assume that all observed nodes are the leaves of the tree to reduce the number of candidate structures. Algorithms have been proposed (Hsu et al., 2009) with sample complexity guarantees for learning HMMs under the condition that the joint distribution of the observed variables generated by distinct hidden states are distinct.

Another related line of research is that of (hierarchical) clustering. See Jain et al. (1999), Balcan and Gupta (2010) and the references therein for extensive discussions. The primary objective of hierarchical clustering is to build a tree consisting of nested partitions of the observed data, where the leaves (typically) consist of single data points while the internal nodes represent coarser partitions. The difference from our work is that hierarchical clustering does not assume a probabilistic graphical model (Markov random field) on the data, but imposes constraints on the data points via a similarity matrix. We are interested in learning tree-structured graphical models with hidden variables.

The reconstruction of latent trees has been studied extensively by the *phylogenetic* community where sequences of extant species are available and the unknown phylogenetic tree is to be inferred from these sequences. See Durbin et al. (1999) for a thorough overview. Efficient algorithms with provable performance guarantees are available (Erdős et al., 1999; Daskalakis et al., 2006). However, the works in this area mostly assume that only the leaves are observed and each internal node (which is hidden) has the same degree except for the root. The most popular algorithm for constructing phylogenetic trees is the *neighbor-joining (NJ) method* by Saitou and Nei (1987). Like our recursive grouping algorithm, the input to the algorithm is a set of statistical distances between observed variables. The algorithm proceeds by recursively pairing two nodes that are the closest neighbors in the true latent tree and introducing a hidden node as the parent of the two nodes. For more details on NJ, the reader is referred to Durbin et al. (1999, Section 7.3).

Another popular class of reconstruction methods used in the phylogenetic community is the family of *quartet-based distance methods* (Bandelth and Dress, 1986; Erdős et al., 1999; Jiang et al., 2001).[3] Quartet-based methods first construct a set of quartets for all subsets of four observed nodes. Subsequently, these quartets are then combined to form a latent tree. However, when we only have access to the samples at the observed nodes, then it is not straightforward to construct a latent tree from a set of quartets since the quartets may be not be consistent.[4] In fact, it is known that the problem of determining a latent tree that agrees with the maximum number of quartets is NP-hard (Steel, 1992), but many heuristics have been proposed (Farris, 1972; Sattath and Tversky, 1977). Also, in practice, quartet-based methods are usually much less accurate than NJ (St. John et al., 2003), and hence, we only compare our proposed algorithms to NJ in our experiments. For further comparisons (the sample complexity and other aspects of) between the quartet methods and NJ, the reader is referred to Csűrös (2000) and St. John et al. (2003).

3. A *quartet* is simply an unrooted binary tree on a set of four observed nodes.
4. The term *consistent* here is not the same as the estimation-theoretic one. Here, we say that a set of quartets is *consistent* if there exists a latent tree such that all quartets agree with the tree.

Another distance-based algorithm was proposed in Pearl (1988, Section 8.3.3). This algorithm is very similar in spirit to quartet-based methods but instead of finding quartets for *all* subsets of four observed nodes, it finds *just enough* quartets to determine the location of each observed node in the tree. Although the algorithm is consistent, it performs poorly when only the samples of observed nodes are available (Pearl, 1988, Section 8.3.5).

The learning of phylogenetic trees is related to the emerging field of *network tomography* (Castro et al., 2004) in which one seeks to learn characteristics (such as structure) from data which are only available at the end points (e.g., sources and sinks) of the network. However, again observations are only available at the leaf nodes and usually the objective is to estimate the delay distributions corresponding to nodes linked by an edge (Tsang et al., 2003; Bhamidi et al., 2009). The modeling of the delay distributions is different from the learning of latent tree graphical models discussed in this paper.

1.2 Paper Organization

The rest of the paper is organized as follows. In Section 2, we introduce the notations and terminologies used in the paper. In Section 3, we introduce the notion of information distances which are used to reconstruct tree models. In the subsequent two sections, we make two assumptions: Firstly, the true distribution is a latent tree and secondly, perfect knowledge of information distance of observed variables is available. We introduce recursive grouping in Section 4. This is followed by our second algorithm CLGrouping in Section 5. In Section 6, we relax the assumption that the information distances are known and develop sample based algorithms and at the same time provide sample complexity guarantees for recursive grouping and CLGrouping. We also discuss extensions of our algorithms for the case when the underlying model is not a tree and our goal is to learn an approximation to it using a latent tree model. We demonstrate the empirical performance of our algorithms in Section 7 and conclude the paper in Section 8. The appendix includes proofs for the theorems presented in the paper.

2. Latent Tree Graphical Models

In this section, we provide some background and introduce the notion of minimal-tree extensions and consistency.

2.1 Undirected Graphs

Let $G = (W, E)$ be an undirected graph with vertex (or node) set $W = \{1, \dots, M\}$ and edge set $E \subset \binom{W}{2}$. Let $\mathrm{nbd}(i; G)$ and $\mathrm{nbd}[i; G]$ be the set of neighbors of node i and the *closed neighborhood* of i respectively, that is, $\mathrm{nbd}[i; G] := \mathrm{nbd}(i; G) \cup \{i\}$. If an undirected graph does not include any loops, it is called a *tree*. A collection of disconnected trees is called a *forest*.[5] For a tree $T = (W, E)$, the set of leaf nodes (nodes with degree 1), the maximum degree, and the diameter are denoted by $\mathrm{Leaf}(T)$, $\Delta(T)$, and $\mathrm{diam}(T)$ respectively. The *path* between two nodes i and j in a tree $T = (W, E)$, which is unique, is the set of edges connecting i and j and is denoted as $\mathrm{Path}((i, j); E)$. The *distance* between any two nodes i and j is the number of edges in $\mathrm{Path}((i, j); E)$. In an undirected tree, we can choose a *root node* arbitrarily, and define the parent-child relationships with respect to the root:

5. Strictly speaking, a graph with no loops is called a forest, and it is called a tree only if every node is connected to each other.

for a pair neighboring nodes i and j, if i is closer to the root than j is, then i is called the *parent* of j, and j is called the *child* of i. Note that the root node does not have any parent, and for all other nodes in the tree, there exists exactly one parent. We use $\mathcal{C}(i)$ to denote the set of child nodes. A set of nodes that share the same parent is called a *sibling* group. A *family* is the union of the siblings and the associated parent.

A *latent tree* is a tree with node set $W := V \cup H$, the union of a set of observed nodes V (with $m = |V|$), and a set of latent (or hidden) nodes H. The *effective depth* $\delta(T;V)$ (with respect to V) is the maximum distance of a hidden node to its closest observed node, that is,

$$\delta(T;V) := \max_{i \in H} \min_{j \in V} |\mathrm{Path}((i,j);T)|. \tag{1}$$

2.2 Graphical Models

An *undirected graphical model* (Lauritzen, 1996) is a family of multivariate probability distributions that factorize according to a graph $G = (W,E)$. More precisely, let $\mathbf{X} = (X_1, \ldots, X_M)$ be a random vector, where each random variable X_i, which takes on values in an alphabet $\mathcal{X}$, corresponds to variable at node $i \in V$. The set of edges E encodes the set of conditional independencies in the model. The random vector $\mathbf{X}$ is said to be *Markov* on G if for every i, the random variable X_i is conditionally independent of all other variables given its neighbors, that is, if p is the joint distribution[6] of $\mathbf{X}$, then

$$p(x_i | x_{\mathrm{nbd}(i;G)}) = p(x_i | x_{\setminus i}), \tag{2}$$

where $x_{\setminus i}$ denotes the set of all variables[7] excluding x_i. Equation (2) is known as the *local Markov property*.

In this paper, we consider both discrete and Gaussian graphical models. For discrete models, the alphabet $\mathcal{X} = \{1, \ldots, K\}$ is a finite set. For Gaussian graphical models, $\mathcal{X} = \mathbb{R}$ and furthermore, without loss of generality, we assume that the mean is known to be the zero vector and hence, the joint distribution

$$p(\mathbf{x}) = \frac{1}{\det(2\pi\mathbf{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^T \mathbf{\Sigma}^{-1} \mathbf{x}\right)$$

depends only on the covariance matrix $\mathbf{\Sigma}$.

An important and tractable class of graphical models is the set of tree-structured graphical models, that is, multivariate probability distributions that are Markov on an undirected tree $T = (W,E)$. It is known from junction tree theory (Cowell et al., 1999) that the joint distribution p for such a model factorizes as

$$p(x_1, \ldots, x_M) = \prod_{i \in W} p(x_i) \prod_{(i,j) \in E} \frac{p(x_i, x_j)}{p(x_i)p(x_j)}. \tag{3}$$

That is, the sets of marginal $\{p(x_i) : i \in W\}$ and pairwise joints on the edges $\{p(x_i, x_j) : (i,j) \in E\}$ fully characterize the joint distribution of a tree-structured graphical model.

A special class of a discrete tree-structured graphical models is the set of *symmetric discrete distributions*. This class of models is characterized by the fact that the pairs of variables (X_i, X_j) on

6. We abuse the term *distribution* to mean a probability mass function in the discrete case (density with respect to the counting measure) and a probability density function (density with respect to the Lebesgue measure) in the continuous case.

7. We will use the terms node, vertex and variable interchangeably in the sequel.

all the edges $(i,j) \in E$ follow the conditional probability law:

$$p(x_i|x_j) = \begin{cases} 1-(K-1)\theta_{ij}, & \text{if } x_i = x_j, \\ \theta_{ij}, & \text{otherwise,} \end{cases} \tag{4}$$

and the marginal distribution of *every* variable in the tree is uniform, that is, $p(x_i) = 1/K$ for all $x_i \in \mathcal{X}$ and for all $i \in V \cup H$. The parameter $\theta_{ij} \in (0, 1/K)$ in (4), which does not depend on the state values $x_i, x_j \in \mathcal{X}$ (but can be different for different pairs $(i,j) \in E$), is known as the *crossover probability*.

Let $\mathbf{x}^n := \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ be a set of n i.i.d. samples drawn from a graphical model (distribution) p, Markov on a latent tree $T_p = (W, E_p)$, where $W = V \cup H$. Each sample $\mathbf{x}^{(l)} \in \mathcal{X}^M$ is a length-M vector. In our setup, the learner only has access to samples drawn from the observed node set V, and we denote this set of sub-vectors containing only the elements in V, as $\mathbf{x}_V^n := \{\mathbf{x}_V^{(1)}, \ldots, \mathbf{x}_V^{(n)}\}$, where each observed sample $\mathbf{x}_V^{(l)} \in \mathcal{X}^m$ is a length-m vector. Our algorithms learn latent tree structures using the information distances (defined in Section 3) between pairs of observed variables, which can be estimated from samples.

We now comment on the above model assumptions. Note that we assume that the the hidden variables have the *same* domain as the observed ones (all of which also have a common domain). We do not view this as a serious modeling restriction since we develop efficient algorithms with strong theoretical guarantees, and these algorithms have very good performance on real-world data (see Section 7). Nonetheless, it may be possible to develop a unified framework to incorporate variables with different state spaces (i.e., both continuous and discrete) under a reproducing kernel Hilbert space (RKHS) framework along the lines of Song et al. (2010). We defer this to future work.

2.3 Minimal Tree Extensions

Our ultimate goal is to recover the graphical model p, that is, the latent tree structure and its parameters, given n i.i.d. samples of the observed variables $\mathbf{x}_V^n$. However, in general, there can be multiple latent tree models which result in the same observed statistics, that is, the same joint distribution p_V of the observed variables. We consider the class of tree models where it is possible to recover the latent tree model uniquely and provide necessary conditions for structure identifiability, that is, the identifiability of the edge set E.

Firstly, we limit ourselves to the scenario where *all* the random variables (both observed and latent) take values on a common alphabet $\mathcal{X}$. Thus, in the Gaussian case, each hidden and observed variable is a univariate Gaussian. In the discrete case, each variable takes on values in the same finite alphabet $\mathcal{X}$. Note that the model may not be identifiable if some of the hidden variables are allowed to have arbitrary alphabets. As an example, consider a discrete latent tree model with binary observed variables ($K = 2$). A latent tree with the simplest structure (fewest number of nodes) is a tree in which all m observed binary variables are connected to one hidden variable. If we allow the hidden variable to take on 2^m states, then the tree can describe all possible statistics among the m observed variables, that is, the joint distribution p_V can be arbitrary.[8]

A probability distribution $p_V(\mathbf{x}_V)$ is said to be *tree-decomposable* if it is the marginal (of variables in V) of a tree-structured graphical model $p(\mathbf{x}_V, \mathbf{x}_H)$. In this case, p (over variables in W) is said to be a *tree extension* of p_V (Pearl, 1988). A distribution p is said to have a *redundant* hidden node $h \in H$ if we can remove h and the marginal on the set of visible nodes V remains as p_V.

8. This follows from a elementary parameter counting argument.

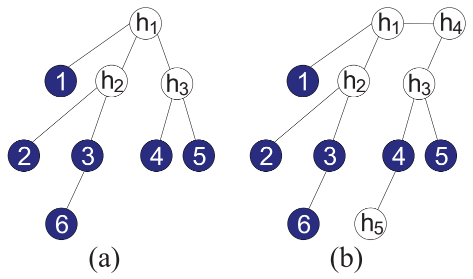


Figure 1: Examples of minimal latent trees. Shaded nodes are observed and unshaded nodes are hidden. (a) An identifiable tree. (b) A non-identifiable tree because h_4 and h_5 have degrees less than 3.

The following conditions ensure that a latent tree does not include a redundant hidden node (Pearl, 1988):

(C1) Each hidden variable has at least three neighbors (which can be either hidden or observed). Note that this ensures that all leaf nodes are observed (although not all observed nodes need to be leaves).

(C2) Any two variables connected by an edge in the tree model are neither perfectly dependent nor independent.

Figure 1(a) shows an example of a tree satisfying (C1). If (C2), which is a condition on parameters, is also satisfied, then the tree in Figure 1(a) is identifiable. The tree shown in Figure 1(b) does not satisfy (C1) because h_4 and h_5 have degrees less than 3. In fact, if we marginalize out the hidden variables h_4 and h_5, then the resulting model has the same tree structure as in Figure 1(a).

We assume throughout the paper that (C2) is satisfied for all probability distributions. Let $\mathcal{T}_{\geq 3}$ be the set of (latent) trees satisfying (C1). We refer to $\mathcal{T}_{\geq 3}$ as the set of *minimal (or identifiable) latent trees*. Minimal latent trees do not contain redundant hidden nodes. The distribution p (over W and Markov on some tree in $\mathcal{T}_{\geq 3}$) is said to be a *minimal tree extension* of p_V. As illustrated in Figure 1, using marginalization operations, any non-minimal latent tree distribution can be reduced to a minimal latent tree model.

Proposition 1 (Minimal Tree Extensions) (Pearl, 1988, Section 8.3)

(i) For every tree-decomposable distribution p_V, there exists a minimal tree extension p Markov on a tree $T \in \mathcal{T}_{\geq 3}$, which is unique up to the renaming of the variables or their values.

(ii) For Gaussian and binary distributions, if p_V is known exactly, then the minimal tree extension p can be recovered.

(iii) The structure of T is uniquely determined by the pairwise distributions of observed variables $p(x_i, x_j)$ for all $i, j \in V$.

2.4 Consistency

We now define the notion of consistency. In Section 6, we show that our latent tree learning algorithms are consistent.

Definition 2 (Consistency) *A* latent tree reconstruction algorithm $\mathcal{A}$ *is a map from the observed samples* $\mathbf{x}_V^n$ *to an estimated tree* $\widehat{T}^n$ *and an estimated tree-structured graphical model* $\widehat{p}^n$*. We say that a latent tree reconstruction algorithm* $\mathcal{A}$ *is* structurally consistent *if there exists a graph homomorphism*[9] *h such that*

$$\lim_{n\to\infty} \Pr(h(\widehat{T}^n) \neq T_p) = 0. \tag{5}$$

Furthermore, we say that $\mathcal{A}$ *is* risk consistent *if to every* $\varepsilon > 0$*,*

$$\lim_{n\to\infty} \Pr\left(D(p\,||\,\widehat{p}^n) > \varepsilon\right) = 0, \tag{6}$$

where $D(p\,||\,\widehat{p}^n)$ *is the KL-divergence (Cover and Thomas, 2006) between the true distribution p and the estimated distribution* $\widehat{p}^n$*.*

In the following sections, we design structurally and risk consistent algorithms for (minimal) Gaussian and symmetric discrete latent tree models, defined in (4). Our algorithms use pairwise distributions between the observed nodes. However, for general discrete models, pairwise distributions between observed nodes are, in general, not sufficient to recover the parameters (Chang and Hartigan, 1991). Therefore, we only prove structural consistency, as defined in (5), for general discrete latent tree models. For such distributions, we consider a two-step procedure for structure and parameter estimation: Firstly, we estimate the structure of the latent tree using the algorithms suggested in this paper. Subsequently, we use the Expectation Maximization (EM) algorithm (Dempster et al., 1977) to infer the parameters. Note that, as mentioned previously, risk consistency will not be guaranteed in this case.

3. Information Distances

The proposed algorithms in this paper receive as inputs the set of so-called (exact or estimated) *information distances*, which are functions of the pairwise distributions. These quantities are defined in Section 3.1 for the two classes of tree-structured graphical models discussed in this paper, namely the Gaussian and discrete graphical models. We also show that the information distances have a particularly simple form for symmetric discrete distributions. In Section 3.2, we use the information distances to infer the relationships between the observed variables such as j is a child of i or i and j are siblings.

3.1 Definitions of Information Distances

We define *information distances* for Gaussian and discrete distributions and show that these distances are additive for tree-structured graphical models. Recall that for two random variables X_i and X_j, the *correlation coefficient* is defined as

$$\rho_{ij} := \frac{\mathrm{Cov}(X_i, X_j)}{\sqrt{\mathrm{Var}(X_i)\mathrm{Var}(X_j)}}. \tag{7}$$

9. A graph homomorphism is a mapping between graphs that respects their structure. More precisely, a *graph homomorphism h* from a graph $G = (W, E)$ to a graph $G' = (V', E')$, written $h : G \to G'$ is a mapping $h : V \to V'$ such that $(i, j) \in E$ implies that $(h(i), h(j)) \in E'$.

For Gaussian graphical models, the information distance associated with the pair of variables X_i and X_j is defined as:

$$d_{ij} := -\log|\rho_{ij}|. \tag{8}$$

Intuitively, if the information distance d_{ij} is large, then X_i and X_j are weakly correlated and vice-versa.

For discrete random variables, let $\mathbf{J}^{ij}$ denote the joint probability matrix between X_i and X_j (i.e., $J^{ij}_{ab} = p(x_i = a, x_j = b), a, b \in \mathcal{X}$). Also let $\mathbf{M}^i$ be the diagonal marginal probability matrix of X_i (i.e., $M^i_{aa} = p(x_i = a)$). For discrete graphical models, the information distance associated with the pair of variables X_i and X_j is defined as Lake (1994):

$$d_{ij} := -\log \frac{|\det \mathbf{J}^{ij}|}{\sqrt{\det \mathbf{M}^i \det \mathbf{M}^j}}. \tag{9}$$

Note that for binary variables, that is, $K = 2$, the value of d_{ij} in (9) reduces to the expression in (8), that is, the information distance is a function of the correlation coefficient, defined in (7), just as in the Gaussian case.

For symmetric discrete distributions defined in (4), the information distance defined for discrete graphical models in (9) reduces to

$$d_{ij} := -(K-1)\log(1 - K\theta_{ij}). \tag{10}$$

Note that there is one-to-one correspondence between the information distances d_{ij} and the model parameters for Gaussian distributions (parametrized by the correlation coefficient ρ_{ij}) in (8) and the symmetric discrete distributions (parametrized by the crossover probability θ_{ij}) in (10). Thus, these two distributions are completely characterized by the information distances d_{ij}. On the other hand, this does not hold for general discrete distributions.

Moreover, if the underlying distribution is a symmetric discrete model or a Gaussian model, the information distance d_{ij} and the mutual information $I(X_i; X_j)$ (Cover and Thomas, 2006) are monotonic, and we will exploit this result in Section 5. For general distributions, this is not valid. See Section 5.5 for further discussions.

Equipped with these definitions of information distances, assumption (C2) in Section 2.3 can be rewritten as the following: There exists constants $0 < l, u < \infty$, such that

$$\text{(C2)} \qquad l \leq d_{ij} \leq u, \qquad \forall (i,j) \in E_p. \tag{11}$$

Proposition 3 (Additivity of Information Distances) *The information distances d_{ij} defined in (8), (9), and (10) are* additive tree metrics *(Erdős et al., 1999). In other words, if the joint probability distribution $p(\mathbf{x})$ is a tree-structured graphical model Markov on the tree $T_p = (W, E_p)$, then the information distances are additive on T_p:*

$$d_{kl} = \sum_{(i,j) \in \text{Path}((k,l); E_p)} d_{ij}, \quad \forall k, l \in W. \tag{12}$$

The property in (12) implies that if each pair of vertices $i, j \in W$ is assigned the weight d_{ij}, then T_p is a minimum spanning tree on W, denoted as $\text{MST}(W; \mathbf{D})$, where $\mathbf{D}$ is the information distance matrix with elements d_{ij} for all $i, j \in V$.

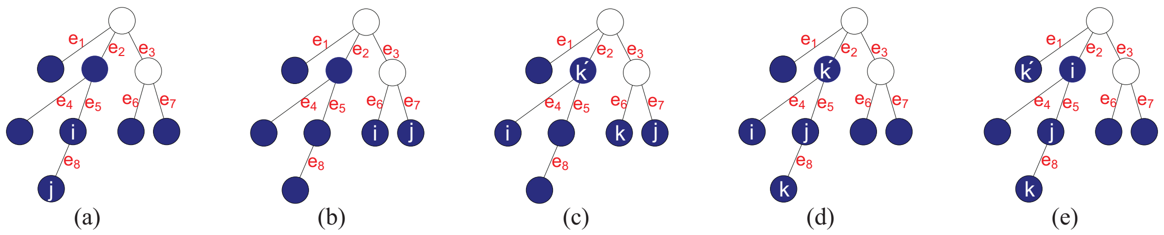


Figure 2: Examples for each case in TestNodeRelationships. For each edge, e_i represents the information distance associated with the edge. (a) Case 1: $\Phi_{ijk} = -e_8 = -d_{ij}$ for all $k \in V \setminus \{i,j\}$. (b) Case 2: $\Phi_{ijk} = e_6 - e_7 \neq d_{ij} = e_6 + e_7$ for all $k \in V \setminus \{i,j\}$ (c) Case 3a: $\Phi_{ijk} = e_4 + e_2 + e_3 - e_7 \neq \Phi_{ijk'} = e_4 - e_2 - e_3 - e_7$. (d) Case 3b: $\Phi_{ijk} = e_4 + e_5 \neq \Phi_{ijk'} = e_4 - e_5$. (e) Case 3c: $\Phi_{ijk} = e_5 \neq \Phi_{ijk'} = -e_5$.

It is straightforward to show that the information distances are additive for the Gaussian and symmetric discrete cases using the local Markov property of graphical models. For general discrete distributions with information distance as in (9), see Lake (1994) for the proof. In the rest of the paper, we map the parameters of Gaussian and discrete distributions to an information distance matrix $\mathbf{D} = [d_{ij}]$ to unify the analyses for both cases.

3.2 Testing Inter-Node Relationships

In this section, we use Proposition 3 to ascertain child-parent and sibling (cf., Section 2.1) relationships between the variables in a latent tree-structured graphical model. To do so, for any three variables $i,j,k \in V$, we define $\Phi_{ijk} := d_{ik} - d_{jk}$ to be the difference between the information distances d_{ik} and d_{jk}. The following lemma suggests a simple procedure to identify the set of relationships between the nodes.

Lemma 4 (Sibling Grouping) *For distances d_{ij} for all $i,j \in V$ on a tree $T \in \mathcal{T}_{\geq 3}$, the following two properties on $\Phi_{ijk} = d_{ik} - d_{jk}$ hold:*

(i) $\Phi_{ijk} = d_{ij}$ for all $k \in V \setminus \{i,j\}$ if and only if i is a leaf node and j is its parent.

(i) $\Phi_{ijk} = -d_{ij}$ for all $k \in V \setminus \{i,j\}$ if and only if j is a leaf node and i is its parent.

(ii) $-d_{ij} < \Phi_{ijk} = \Phi_{ijk'} < d_{ij}$ for all $k,k' \in V \setminus \{i,j\}$ if and only if both i and j are leaf nodes and they have the same parent, that is, they belong to the same sibling group.

The proof of the lemma uses Proposition 3 and is provided in Appendix A.1. Given Lemma 4, we can first determine all the values of Φ_{ijk} for triples $i,j,k \in V$. Now we can determine the relationship between nodes i and j as follows: Fix the pair of nodes $i,j \in V$ and consider all the other nodes $k \in V \setminus \{i,j\}$. Then, there are three cases for the set $\{\Phi_{ijk} : k \in V \setminus \{i,j\}\}$:

1. $\Phi_{ijk} = d_{ij}$ for all $k \in V \setminus \{i,j\}$. Then, i is a leaf node and j is a parent of i. Similarly, if $\Phi_{ijk} = -d_{ij}$ for all $k \in V \setminus \{i,j\}$, j is a leaf node and i is a parent of j.

2. Φ_{ijk} is constant for all $k \in V \setminus \{i,j\}$ but *not* equal to either d_{ij} or $-d_{ij}$. Then i and j are leaf nodes and they are siblings.

3. Φ_{ijk} is not equal for all $k \in V \setminus \{i,j\}$. Then, there are three cases: Either

    (a) Nodes i and j are not siblings nor have a parent-child relationship or,

    (b) Nodes i and j are siblings but at least one of them is not a leaf or,

    (c) Nodes i and j have a parent-child relationship but the child is not a leaf.

Thus, we have a simple test to determine the relationship between i and j and to ascertain whether i and j are leaf nodes. We call the above test TestNodeRelationships. See Figure 2 for examples. By running this test for all i and j, we can determine all the relationships among all pairs of observed variables.

In the following section, we describe a recursive algorithm that is based on the above TestNodeRelationships procedure to reconstruct the entire latent tree model assuming that the true model is a latent tree and that the true distance matrix $\mathbf{D} = [d_{ij}]$ are known. In Section 5, we provide improved algorithms for the learning of latent trees again assuming that $\mathbf{D}$ is known. Subsequently, in Section 6, we develop algorithms for the *consistent* reconstruction of latent trees when information distances are unknown and we have to estimate them from the samples $\mathbf{x}_V^n$. In addition, in Section 6.6 we discuss how to extend these algorithms for the case when p_V is not necessarily tree-decomposable, that is, the original graphical model is not assumed to be a latent tree.

4. Recursive Grouping Algorithm Given Information Distances

This section is devoted to the development of the first algorithm for reconstructing latent tree models, recursive grouping (RG). At a high level, RG is a recursive procedure in which at each step, TestNodeRelationships is used to identify nodes that belong to the same family. Subsequently, RG introduces a parent node if a family of nodes (i.e., a sibling group) does not contain an observed parent. This newly introduced parent node corresponds to a hidden node in the original unknown latent tree. Once such a parent (i.e., hidden) node h is introduced, the information distances from h to all other observed nodes can be computed.

The inputs to RG are the vertex set V and the matrix of information distances $\mathbf{D}$ corresponding to a latent tree. The algorithm proceeds by recursively grouping nodes and adding hidden variables. In each iteration, the algorithm acts on a so-called active set of nodes Y, and in the process constructs a new active set Y_{new} for the next iteration.[10] The steps are as follows:

1. Initialize by setting $Y := V$ to be the set of observed variables.

2. Compute $\Phi_{ijk} = d_{ik} - d_{jk}$ for all $i,j,k \in Y$.

3. Using the TestNodeRelationships procedure, define $\{\Pi_l\}_{l=1}^L$ to be the coarsest partition[11] of Y such that for every subset Π_l (with $|\Pi_l| \geq 2$), any two nodes in Π_l are either siblings which

10. Note that the current active set is also used (in Step 6) after the new active set has been defined. For clarity, we also introduce the quantity Y_{old} in Steps 5 and 6.

11. Recall that a *partition* P of a set Y is a collection of nonempty subsets $\{\Pi_l \subset Y\}_{l=1}^L$ such that $\cup_{l=1}^L \Pi_l = Y$ and $\Pi_l \cap \Pi_{l'} = \emptyset$ for all $l \neq l'$. A partition P is said to be *coarser than* another partition P' if every element of P' is a subset of some element of P.

are leaf nodes or they have a parent-child relationship[12] in which the child is a leaf.[13] Note that for some l, Π_l may consist of a single node. Begin to construct the new active set by adding nodes in these single-node partitions: $Y_{\text{new}} \leftarrow \bigcup_{l:|\Pi_l|=1} \Pi_l$.

4. For each $l = 1,\dots,L$ with $|\Pi_l| \geq 2$, if Π_l contains a parent node u, update $Y_{\text{new}} \leftarrow Y_{\text{new}} \cup \{u\}$. Otherwise, introduce a new hidden node h, connect h (as a parent) to every node in Π_l, and set $Y_{\text{new}} \leftarrow Y_{\text{new}} \cup \{h\}$.

5. Update the active set: $Y_{\text{old}} \leftarrow Y$ and $Y \leftarrow Y_{\text{new}}$.

6. For each new hidden node $h \in Y$, compute the information distances d_{hl} for all $l \in Y$ using (13) and (14) described below.

7. If $|Y| \geq 3$, return to step 2. Otherwise, if $|Y| = 2$, connect the two remaining nodes in Y with an edge then stop. If instead $|Y| = 1$, do nothing and stop.

We now describe how to compute the information distances in Step 6 for each new hidden node $h \in Y$ and all other active nodes $l \in Y$. Let $i, j \in \mathcal{C}(h)$ be two children of h, and let $k \in Y_{\text{old}} \setminus \{i, j\}$ be any other node in the previous active set. From Lemma 4 and Proposition 3, we have that $d_{ih} - d_{jh} = d_{ik} - d_{jk} = \Phi_{ijk}$ and $d_{ih} + d_{jh} = d_{ij}$, from which we can recover the information distances between a previously active node $i \in Y_{\text{old}}$ and its new hidden parent $h \in Y$ as follows:

$$d_{ih} = \frac{1}{2}\left(d_{ij} + \Phi_{ijk}\right). \tag{13}$$

For any other active node $l \in Y$, we can compute d_{hl} using a child node $i \in \mathcal{C}(h)$ as follows:

$$d_{hl} = \begin{cases} d_{il} - d_{ih}, & \text{if } l \in Y_{\text{old}}, \\ d_{ik} - d_{ih} - d_{lk}, & \text{otherwise, where } k \in \mathcal{C}(l). \end{cases} \tag{14}$$

Using Equations (13) and (14), we can infer all the information distances d_{hl} between a newly introduced hidden node h to all other active nodes $l \in Y$. Consequently, we have all the distances d_{ij} between all pairs of nodes in the active set Y. It can be shown that this algorithm recovers all minimal latent trees. The proof of the following theorem is provided in Appendix A.2.

Theorem 5 (Correctness and Computational Complexity of RG) *If $T_p \in \mathcal{T}_{\geq 3}$ and the matrix of information distances $\mathbf{D}$ (between nodes in V) is available, then RG outputs the true latent tree T_p correctly in time $O(\text{diam}(T_p)m^3)$.*

We now use a concrete example to illustrate the steps involved in RG. In Figure 3(a), the original unknown latent tree is shown. In this tree, nodes $1,\dots,6$ are the observed nodes and h_1, h_2, h_3 are the hidden nodes. We start by considering the set of observed nodes as active nodes $Y := V = \{1,\dots,6\}$. Once Φ_{ijk} are computed from the given distances d_{ij}, TestNodeRelationships is used to determine that Y is partitioned into four subsets: $\Pi_1 = \{1\}, \Pi_2 = \{2,4\}, \Pi_3 = \{5,6\}, \Pi_4 = \{3\}$. The subsets Π_1

12. In an undirected tree, the parent-child relationships can be defined with respect to a root node. In this case, the node in the final active set in Step 7 before the algorithm terminates (or one of the two final nodes if $|Y| = 2$) is selected as the root node.

13. Note that since we use the active set Y in the TestNodeRelationships procedure, the leaf nodes are defined with respect to Y, that is, a node is considered as a leaf node if it has only one neighbor in Y or in the set of nodes that have not yet been in an active set.

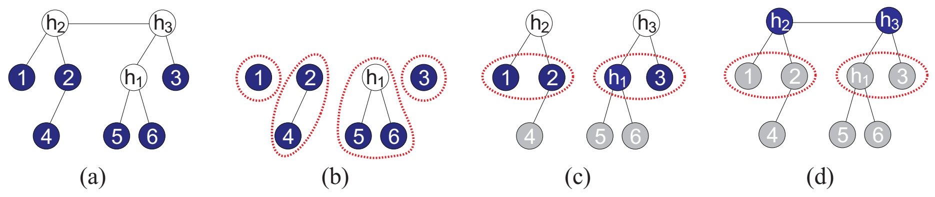


Figure 3: An illustrative example of RG. Solid nodes indicate the active set Y for each iteration. (a) Original latent tree. (b) Output after the first iteration of RG. Red dotted lines indicate the subsets Π_l in the partition of Y. (c) Output after the second iteration of RG. Note that h_3, which was introduced in the first iteration, is an active node for the second iteration. Nodes 4,5, and 6 do not belong to the current active set and are represented in grey. (d) Output after the third iteration of RG, which is same as the original latent tree.

and Π_4 contain only one node. The subset Π_3 contains two siblings that are leaf nodes. The subset Π_2 contains a parent node 2 and a child node 4, which is a leaf node. Since Π_3 does not contain a parent, we introduce a new hidden node h_1 and connect h_1 to 5 and 6 as shown in Figure 3(b). The information distances d_{5h_1} and d_{6h_1} can be computed using (13), for example, $d_{5h_1} = \frac{1}{2}(d_{56} + \Phi_{561})$. The new active set is the union of all nodes in the single-node subsets, a parent node, and a new hidden node $Y_{\text{new}} = \{1,2,3,h_1\}$. Distances among the pairs of nodes in Y_{new} can be computed using (14) (e.g., $d_{1h_1} = d_{15} - d_{5h_1}$). In the second iteration, we again use TestNodeRelationships to ascertain that Y can be partitioned into $\Pi_1 = \{1,2\}$ and $\Pi_2 = \{h_1,3\}$. These two subsets do not have parents so h_2 and h_3 are added to Π_1 and Π_2 respectively. Parent nodes h_2 and h_3 are connected to their children in Π_1 and Π_2 as shown in Figure 3(c). Finally, we are left with the active set as $Y = \{h_2,h_3\}$ and the algorithm terminates after h_2 and h_3 are connected by an edge. The hitherto unknown latent tree is fully reconstructed as shown in Figure 3(d).

A potential drawback of RG is that it involves multiple *local* operations, which may result in a high computational complexity. Indeed, from Theorem 5, the worst-case complexity is $O(m^4)$ which occurs when T_p, the true latent tree, is a hidden Markov model (HMM). This may be computationally prohibitive if m is large. In Section 5 we design an algorithm which uses a *global* pre-processing step to reduce the overall complexity substantially, especially for trees with large diameters (of which HMMs are extreme examples).

5. CLGrouping Algorithm Given Information Distances

In this section, we present CLGrouping, an algorithm for reconstructing latent trees more efficiently than RG. As in Section 4, in this section, we assume that $\mathbf{D}$ is known exactly; the extension to inexact knowledge of $\mathbf{D}$ is discussed in Section 6.5. CLGrouping is a two-step procedure, the first of which is a global pre-processing step that involves the construction of a so-called *Chow-Liu tree* (Chow and Liu, 1968) over the set of observed nodes V. This step identifies nodes that do not belong to the same sibling group. In the second step, we complete the recovery of the latent tree by applying a distance-based latent tree reconstruction algorithm (such as RG or NJ) repeatedly on smaller subsets of nodes. We review the Chow-Liu algorithm in Section 5.1, relate the Chow-Liu tree to the true latent tree in Section 5.2, derive a simple transformation of the Chow-Liu tree to obtain the latent tree in Section 5.3 and propose CLGrouping in Section 5.4. For simplicity, we focus on the Gaussian distributions and the symmetric discrete distributions first, and discuss the extension to general discrete models in Section 5.5.

5.1 A Review of the Chow-Liu Algorithm

In this section, we review the Chow-Liu tree reconstruction procedure. To do so, define $\mathcal{T}(V)$ to be the set of trees with vertex set V and $\mathcal{P}(\mathcal{T}(V))$ to be the set of tree-structured graphical models whose graph has vertex set V, that is, every $q \in \mathcal{P}(\mathcal{T}(V))$ factorizes as in (3).

Given an arbitrary multivariate distribution $p_V(\mathbf{x}_V)$, Chow and Liu (1968) considered the following *KL-divergence minimization* problem:

$$p_{\text{CL}} := \underset{q \in \mathcal{P}(\mathcal{T}(V))}{\text{argmin}} \; D(p_V \,||\, q). \tag{15}$$

That is, among all the tree-structured graphical models with vertex set V, the distribution p_{CL} is the closest one to p_V in terms of the KL-divergence. By using the factorization property in (3), we can easily verify that p_{CL} is Markov on the *Chow-Liu tree* $T_{\text{CL}} = (V, E_{\text{CL}})$ which is given by the optimization problem:[14]

$$T_{\text{CL}} = \underset{T \in \mathcal{T}(V)}{\text{argmax}} \sum_{(i,j) \in T} I(X_i\,;X_j). \tag{16}$$

In (16), $I(X_i\,;X_j) = D(p(x_i,x_j)\,||\,p(x_i)\,p(x_j))$ is the *mutual information* (Cover and Thomas, 2006) between random variables X_i and X_j. The optimization in (16) is a max-weight spanning tree problem (Cormen et al., 2003) which can be solved efficiently in time $O(m^2 \log m)$ using either Kruskal's algorithm (Kruskal, 1956) or Prim's algorithm (Prim, 1957). The edge weights for the max-weight spanning tree are precisely the mutual information quantities between random variables. Note that once the optimal tree T_{CL} is formed, the parameters of p_{CL} in (15) are found by setting the pairwise distributions $p_{\text{CL}}(x_i,x_j)$ on the edges to $p_V(x_i,x_j)$, that is, $p_{\text{CL}}(x_i,x_j) = p_V(x_i,x_j)$ for all $(i,j) \in E_{\text{CL}}$. We now relate the Chow-Liu tree on the observed nodes and the information distance matrix $\mathbf{D}$.

Lemma 6 (Correspondence between T_{CL} and MST**)** *If p_V is a Gaussian distribution or a symmetric discrete distribution, then the Chow-Liu tree in* (16) *reduces to the minimum spanning tree (MST) where the edge weights are the information distances d_{ij}, that is,*

$$T_{\text{CL}} = \text{MST}(V;\mathbf{D}) := \underset{T \in \mathcal{T}(V)}{\text{argmin}} \sum_{(i,j) \in T} d_{ij}. \tag{17}$$

Lemma 6, whose proof is omitted, follows because for Gaussian and symmetric discrete models, the mutual information[15] $I(X_i\,;X_j)$ is a monotonically decreasing function of the information distance d_{ij}.[16] For other graphical models (e.g., non-symmetric discrete distributions), this relationship is not necessarily true. See Section 5.5 for a discussion. Note that when all nodes are observed (i.e., $W = V$), Lemma 6 reduces to Proposition 3.

14. In (16) and the rest of the paper, we adopt the following simplifying notation; If $T = (V,E)$ and if $(i,j) \in E$, we will also say that $(i,j) \in T$.
15. Note that, unlike information distances d_{ij}, the mutual information quantities $I(X_i\,;X_j)$ do not form an additive metric on T_p.
16. For example, in the case of Gaussians, $I(X_i\,;X_j) = -\frac{1}{2}\log(1-\rho_{ij}^2)$ (Cover and Thomas, 2006).

5.2 Relationship between the Latent Tree and the Chow-Liu Tree (MST)

In this section, we relate $\mathrm{MST}(V;\mathbf{D})$ in (17) to the original latent tree T_p. To relate the two trees, $\mathrm{MST}(V;\mathbf{D})$ and T_p, we first introduce the notion of a surrogate node.

Definition 7 (Surrogate Node) *Given the latent tree $T_p = (W, E_p)$ and any node $i \in W$, the* surrogate node *of i with respect to V is defined as*

$$\mathrm{Sg}(i;T_p,V) := \operatorname*{argmin}_{j\in V} d_{ij}.$$

Intuitively, the surrogate node of a hidden node $h \in H$ is an observed node $j \in V$ that is most strongly correlated to h. In other words, the information distance between h and j is the smallest. Note that if $i \in V$, then $\mathrm{Sg}(i;T_p,V) = i$ since $d_{ii} = 0$. The map $\mathrm{Sg}(i;T_p,V)$ is a many-to-one function, that is, several nodes may have the same surrogate node, and its inverse is the *inverse surrogate set of i* denoted as

$$\mathrm{Sg}^{-1}(i;T_p,V) := \{h \in W : \mathrm{Sg}(h;T_p,V) = i\}.$$

When the tree T_p and the observed vertex set V are understood from context, the surrogate node of h and the inverse surrogate set of i are abbreviated as $\mathrm{Sg}(h)$ and $\mathrm{Sg}^{-1}(i)$ respectively. We now relate the original latent tree $T_p = (W, E_p)$ to the Chow-Liu tree (also termed the MST) $\mathrm{MST}(V;\mathbf{D})$ formed using the distance matrix $\mathbf{D}$.

Lemma 8 (Properties of the MST) *The MST in* (17) *and surrogate nodes satisfy the following properties:*

(i) The surrogate nodes of any two neighboring nodes in E_p are neighbors in the MST, that is, for all $i, j \in W$ with $\mathrm{Sg}(i) \neq \mathrm{Sg}(j)$,

$$(i,j) \in E_p \Rightarrow (\mathrm{Sg}(i), \mathrm{Sg}(j)) \in \mathrm{MST}(V;\mathbf{D}). \tag{18}$$

(ii) If $j \in V$ and $h \in \mathrm{Sg}^{-1}(j)$, then every node along the path connecting j and h belongs to the inverse surrogate set $\mathrm{Sg}^{-1}(j)$.

(iii) The maximum degree of the MST satisfies

$$\Delta(\mathrm{MST}(V;\mathbf{D})) \le \Delta(T_p)^{1+\frac{u}{l}\delta(T_p;V)}, \tag{19}$$

where $\delta(T_p;V)$ is the effective depth defined in (1) *and l, u are the bounds on the information distances on edges in T_p defined in* (11).

The proof of this result can be found in Appendix A.3. As a result of Lemma 8, the properties of $\mathrm{MST}(V;\mathbf{D})$ can be expressed in terms of the original latent tree T_p. For example, in Figure 5(a), a latent tree is shown with its corresponding surrogacy relationships, and Figure 5(b) shows the corresponding MST over the observed nodes.

The properties in Lemma 8(i-ii) can also be regarded as *edge-contraction operations* (Robinson and Foulds, 1981) in the original latent tree to obtain the MST. More precisely, an edge-contraction operation on an edge $(j,h) \in V \times H$ in the latent tree T_p is defined as the "shrinking" of (j,h) to a single node whose label is the observed node j. Thus, the edge (j,h) is "contracted" to a single

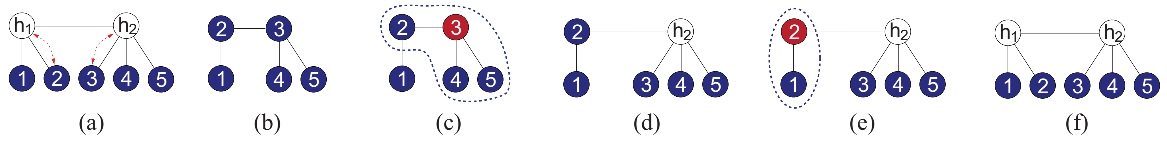


Figure 4: An illustration of CLBlind. The shaded nodes are the observed nodes and the rest are hidden nodes. The dotted lines denote surrogate mappings for the hidden nodes. (a) Original latent tree, which belongs to the class of blind latent graphical models, (b) Chow-Liu tree over the observed nodes, (c) Node 3 is the input to the blind transformation, (d) Output after the blind transformation, (e) Node 2 is the input to the blind transformation, (f) Output after the blind transformation, which is same as the original latent tree.

node j. By using Lemma 8(i-ii), we observe that the Chow-Liu tree $\mathrm{MST}(V;\mathbf{D})$ is formed by applying edge-contraction operations to each (j,h) pair for all $h \in \mathrm{Sg}^{-1}(j) \cap H$ sequentially until all pairs have been contracted to a single node j. For example, the MST in Figure 5(b) is obtained by contracting edges $(3,h_3)$, $(5,h_2)$, and then $(5,h_1)$ in the latent tree in Figure 5(a).

The properties in Lemma 8 can be used to design efficient algorithms based on transforming the MST to obtain the latent tree T_p. Note that the maximum degree of the MST, $\Delta(\mathrm{MST}(V;\mathbf{D}))$, is bounded by the maximum degree in the original latent tree. The quantity $\Delta(\mathrm{MST}(V;\mathbf{D}))$ determines the computational complexity of one of our proposed algorithms (CLGrouping) and it is small if the depth of the latent tree $\delta(T_p;V)$ is small (e.g., HMMs) and the information distances d_{ij} satisfy tight bounds (i.e., u/l is close to unity). The latter condition holds for (almost) homogeneous models in which all the information distances d_{ij} on the edges are almost equal.

5.3 Chow-Liu Blind Algorithm for a Subclass of Latent Trees

In this section, we present a simple and intuitive transformation of the Chow-Liu tree that produces the original latent tree. However, this algorithm, called Chow-Liu Blind (or CLBlind), is applicable only to a subset of latent trees called *blind latent tree-structured graphical models* $\mathcal{P}(\mathcal{T}_{\mathrm{blind}})$. Equipped with the intuition from CLBlind, we generalize it in Section 5.4 to design the CLGrouping algorithm that produces the correct latent tree structure from the MST *for all* minimal latent tree models.

If $p \in \mathcal{P}(\mathcal{T}_{\mathrm{blind}})$, then its structure $T_p = (W, E_p)$ and the distance matrix $\mathbf{D}$ satisfy the following properties:

(i) The true latent tree $T_p \in \mathcal{T}_{\geq 3}$ and all the internal nodes[17] are hidden, that is, $V = \mathrm{Leaf}(T_p)$.

(ii) The surrogate node of (i.e., the observed node with the strongest correlation with) each hidden node is one of its children, that is, $\mathrm{Sg}(h) \in \mathcal{C}(h)$ for all $h \in H$.

We now describe the CLBlind algorithm, which involves two main steps. Firstly, $\mathrm{MST}(V;\mathbf{D})$ is constructed using the distance matrix $\mathbf{D}$. Secondly, we apply the blind transformation of the Chow-Liu tree $\mathsf{BlindTransform}(\mathrm{MST}(V;\mathbf{D}))$, which proceeds as follows:

17. Recall that an internal node is one whose degree is greater than or equal to 2, that is, a non-leaf.

1. Identify the set of internal nodes in $\mathrm{MST}(V;\mathbf{D})$. We perform an operation for each internal node as follows:

2. For internal node i, add a hidden node h to the tree.

3. Connect an edge between h and i (which now becomes a leaf node) and also connect edges between h and the neighbors of i in the *current* tree model.

4. Repeat steps 2 and 3 until all internal nodes have been operated on.

See Figure 4 for an illustration of CLBlind. We use the adjective *blind* to describe the transformation $\mathsf{BlindTransform}(\mathrm{MST}(V;\mathbf{D}))$ since it does not depend on the distance matrix $\mathbf{D}$ but uses *only* the structure of the MST. The following theorem whose proof can be found in Appendix A.4 states the correctness result for CLBlind.

Theorem 9 (Correctness and Computational Complexity of CLBlind) *If the distribution $p \in \mathcal{P}(\mathcal{T}_{\text{blind}})$ is a blind tree-structured graphical model Markov on T_p and the matrix of distances $\mathbf{D}$ is known, then CLBlind outputs the true latent tree T_p correctly in time $O(m^2 \log m)$.*

The first condition on $\mathcal{P}(\mathcal{T}_{\text{blind}})$ that all internal nodes are hidden is not uncommon in applications. For example, in phylogenetics, (DNA or amino acid) sequences of extant species at the leaves are observed, while the sequences of the extinct species are hidden (corresponding to the internal nodes), and the evolutionary (phylogenetic) tree is to be reconstructed. However, the second condition is more restrictive[18] since it implies that each hidden node is directly connected to at least one observed node and that it is closer (i.e., more correlated) to one of its observed children compared to any other observed node. If the first constraint is satisfied but not the second, then the blind transformation $\mathsf{BlindTransform}(\mathrm{MST}(V;\mathbf{D}))$ does not overestimate the number of hidden variables in the latent tree (the proof follows from Lemma 8 and is omitted).

Since the computational complexity of constructing the MST is $O(m^2 \log m)$ where $m = |V|$, and the blind transformation is at most linear in m, the overall computational complexity is $O(m^2 \log m)$. Thus, CLBlind is a computationally efficient procedure compared to RG, described in Section 4.

5.4 Chow-Liu Grouping Algorithm

Even though CLBlind is computationally efficient, it only succeeds in recovering latent trees for a restricted subclass of minimal latent trees. In this section, we propose an efficient algorithm, called CLGrouping that reconstructs *all* minimal latent trees. We also illustrate CLGrouping using an example. CLGrouping uses the properties of the MST as described in Lemma 8.

At a high-level, CLGrouping involves two distinct steps: Firstly, we construct the Chow-Liu tree $\mathrm{MST}(V;\mathbf{D})$ over the set of observed nodes V. Secondly, we apply RG or NJ to reconstruct a latent subtree over the closed neighborhoods of every internal node in $\mathrm{MST}(V;\mathbf{D})$. If RG (respectively NJ) is used, we term the algorithm CLRG (respectively CLNJ). In the rest of the section, we only describe CLRG for concreteness since CLNJ proceeds along similar lines. Formally, CLRG proceeds as follows:

1. Construct the Chow-Liu tree $\mathrm{MST}(V;\mathbf{D})$ as in (17). Set $T = \mathrm{MST}(V;\mathbf{D})$.

18. The second condition on $\mathcal{P}(\mathcal{T}_{\text{blind}})$ holds when the tree is (almost) homogeneous.

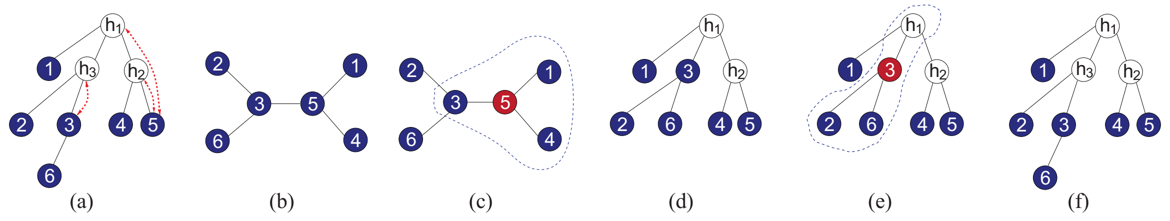


Figure 5: Illustration of CLRG. The shaded nodes are the observed nodes and the rest are hidden nodes. The dotted lines denote surrogate mappings for the hidden nodes so for example, node 3 is the surrogate of h_3. (a) The original latent tree, (b) The Chow-Liu tree (MST) over the observed nodes V, (c) The closed neighborhood of node 5 is the input to RG, (d) Output after the first RG procedure, (e) The closed neighborhood of node 3 is the input to the second iteration of RG, (f) Output after the second RG procedure, which is same as the original latent tree.

2. Identify the set of internal nodes in $\mathrm{MST}(V;\mathbf{D})$.
3. For each internal node i, let $\mathrm{nbd}[i;T]$ be its closed neighborhood in T and let $S = \mathrm{RG}(\mathrm{nbd}[i;T],\mathbf{D})$ be the output of RG with $\mathrm{nbd}[i;T]$ as the set of input nodes.
4. Replace the subtree over node set $\mathrm{nbd}[i;T]$ in T with S. Denote the new tree as T.
5. Repeat steps 3 and 4 until all internal nodes have been operated on.

Note that the only difference between the algorithm we just described and CLNJ is Step 3 in which the subroutine NJ replaces RG. Also, observe in Step 3 that RG is only applied to a small subset of nodes which have been identified in Step 1 as possible neighbors in the true latent tree. This reduces the computational complexity of CLRG compared to RG, as seen in the following theorem whose proof is provided in Appendix A.5. Let $|J| := |V \setminus \mathrm{Leaf}(\mathrm{MST}(V;\mathbf{D}))| < m$ be the number of internal nodes in the MST.

Theorem 10 (Correctness and Computational Complexity of CLRG) *If the distribution $T_p \in \mathcal{T}_{\geq 3}$ is a minimal latent tree and the matrix of information distances $\mathbf{D}$ is available, then CLRG outputs the true latent tree T_p correctly in time $O(m^2 \log m + |J|\Delta^3(\mathrm{MST}(V;\mathbf{D})))$.*

Thus, the computational complexity of CLRG is low when the latent tree T_p has a small maximum degree and a small effective depth (such as the HMM) because (19) implies that $\Delta(\mathrm{MST}(V;\mathbf{D}))$ is also small. Indeed, we demonstrate in Section 7 that there is a significant speedup compared to applying RG over the entire observed node set V.

We now illustrate CLRG using the example shown in Figure 5. The original minimal latent tree $T_p = (W,E)$ is shown in Figure 5(a) with $W = \{1,2,\ldots,6,h_1,h_2,h_3\}$. The set of observed nodes is $V = \{1,\ldots,6\}$ and the set of hidden nodes is $H = \{h_1,h_2,h_3\}$. The Chow-Liu tree $T_{\mathrm{CL}} = \mathrm{MST}(V;\mathbf{D})$ formed using the information distance matrix $\mathbf{D}$ is shown in Figure 5(b). Since nodes 3 and 5 are the only internal nodes in $\mathrm{MST}(V;\mathbf{D})$, two RG operations will be executed on the closed neighborhoods of each of these two nodes. In the first iteration, the closed neighborhood of node 5 is the input to

Latent variables	Distribution	$\mathrm{MST}(V;\mathbf{D}) = T_{\mathrm{CL}}$?	Structure	Parameter
Non-latent	Gaussian	✓	✓	✓
Non-latent	Symmetric Discrete	✓	✓	✓
Non-latent	General Discrete	×	✓	×
Latent	Gaussian	✓	✓	✓
Latent	Symmetric Discrete	✓	✓	✓
Latent	General Discrete	×	✓	×

Table 1: Comparison between various classes of distributions. In the last two columns, we state whether CLGrouping is consistent for learning either the structure or parameters of the model, namely whether CLGrouping is structurally consistent or risk consistent respectively (cf., Definition 2). Note that the first two cases reduce exactly to the algorithm proposed by Chow and Liu (1968) in which the edge weights are the mutual information quantities.

RG. This is shown in Figure 5(c) where $\mathrm{nbd}[5;\mathrm{MST}(V;\mathbf{D})] = \{1,3,4,5\}$, which is then replaced by the output of RG to obtain the tree shown in Figure 5(d). In the next iteration, RG is applied to the closed neighborhood of node 3 in the current tree $\mathrm{nbd}[3;T] = \{2,3,6,h_1\}$ as shown in Figure 5(e). Note that $\mathrm{nbd}[3;T]$ includes $h_1 \in H$, which was introduced by RG in the previous iteration. The distance from h_1 to other nodes in $\mathrm{nbd}[3;T]$ can be computed using the distance between h_1 and its surrogate node 5, which is part of the output of RG, for example, $d_{2h_1} = d_{25} - d_{5h_1}$. The closed neighborhood $\mathrm{nbd}[3;T]$ is then replaced by the output of the second RG operation and the original latent tree T_p is obtained as shown in Figure 5(f).

Observe that the trees obtained at each iteration of CLRG can be related to the original latent tree in terms of edge-contraction operations (Robinson and Foulds, 1981), which were defined in Section 5.2. For example, the Chow-Liu tree in Figure 5(b) is obtained from the latent tree T_p in Figure 5(a) by sequentially contracting all edges connecting an observed node to its inverse surrogate set (cf., Lemma 8(ii)). Upon performing an iteration of RG, these contraction operations are inverted and new hidden nodes are introduced. For example, in Figure 5(d), the hidden nodes h_1, h_2 are introduced after performing RG on the closed neighborhood of node 5 on $\mathrm{MST}(V;\mathbf{D})$. These newly introduced hidden nodes in fact, turn out to be the inverse surrogate set of node 5, that is, $\mathrm{Sg}^{-1}(5) = \{5, h_1, h_2\}$. This is not merely a coincidence and we formally prove in Appendix A.5 that at each iteration, the set of hidden nodes introduced corresponds exactly to the inverse surrogate set of the internal node.

We conclude this section by emphasizing that CLGrouping (i.e., CLRG or CLNJ) has two primary advantages. Firstly, as demonstrated in Theorem 10, the structure of all minimal tree-structured graphical models can be recovered by CLGrouping in contrast to CLBlind. Secondly, it typically has much lower computational complexity compared to RG.

5.5 Extension to General Discrete Models

For general (i.e., not symmetric) discrete models, the mutual information $I(X_i;X_j)$ is in general not monotonic in the information distance d_{ij}, defined in (9).[19] As a result, Lemma 6 does not hold,

19. The mutual information, however, is monotonic in d_{ij} for asymmetric binary discrete models.

that is, the Chow-Liu tree T_{CL} is not necessarily the same as $\text{MST}(V;\mathbf{D})$. However, Lemma 8 does hold for all minimal latent tree models. Therefore, for general (non-symmetric) discrete models, we compute $\text{MST}(V;\mathbf{D})$ (instead of the Chow-Liu tree T_{CL} with edge weights $I(X_i\,;X_j)$), and apply RG or NJ to each internal node and its neighbors. This algorithm guarantees that the structure learned using CLGrouping is the same as T_p if the distance matrix $\mathbf{D}$ is available. These observations are summarized clearly in Table 1. Note that in *all* cases, the latent structure is recovered consistently.

6. Sample-Based Algorithms for Learning Latent Tree Structures

In Sections 4 and 5, we designed algorithms for the exact reconstruction of latent trees assuming that p_V is a tree-decomposable distribution and the matrix of information distances $\mathbf{D}$ is available. In most (if not all) machine learning problems, the pairwise distributions $p(x_i,x_j)$ are unavailable. Consequently, $\mathbf{D}$ is also unavailable so RG, NJ and CLGrouping as stated in Sections 4 and 5 are not directly applicable. In this section, we consider extending RG, NJ and CLGrouping to the case when only samples $\mathbf{x}_V^n$ are available. We show how to modify the previously proposed algorithms to accommodate ML estimated distances and we also provide sample complexity results for *relaxed* versions of RG and CLGrouping.

6.1 ML Estimation of Information Distances

The canonical method for deterministic parameter estimation is via maximum-likelihood (ML) (Serfling, 1980). We focus on Gaussian and symmetric discrete distributions in this section. The generalization to general discrete models is straightforward. For Gaussians graphical models, we use ML to estimate the entries of the covariance matrix,[20] that is,

$$\widehat{\Sigma}_{ij} = \frac{1}{n}\sum_{k=1}^{n} x_i^{(k)} x_j^{(k)}, \qquad \forall i,j \in V. \tag{20}$$

The ML estimate of the correlation coefficient is defined as $\widehat{\rho}_{ij} := \widehat{\Sigma}_{ij}/(\widehat{\Sigma}_{ii}\widehat{\Sigma}_{jj})^{1/2}$. The estimated information distance is then given by the analog of (8), that is, $\widehat{d}_{ij} = -\log|\widehat{\rho}_{ij}|$. For symmetric discrete distributions, we estimate the crossover probability θ_{ij} via ML as[21]

$$\widehat{\theta}_{ij} = \frac{1}{n}\sum_{k=1}^{n} \mathbb{I}\{x_i^{(k)} \neq x_j^{(k)}\}, \qquad \forall i,j \in V.$$

The estimated information distance is given by the analogue of (10), that is, $\widehat{d}_{ij} = -(K-1)\log(1-K\widehat{\theta}_{ij})$. For both classes of models, it can easily be verified from the Central Limit Theorem and continuity arguments (Serfling, 1980) that $\widehat{d}_{ij} - d_{ij} = O_p(n^{-1/2})$, where n is the number of samples. This means that the estimates of the information distances are consistent with rate of convergence being $n^{-1/2}$. The $m \times m$ matrix of estimated information distances is denoted as $\widehat{\mathbf{D}} = [\widehat{d}_{ij}]$.

6.2 Post-processing Using Edge Contractions

For all sample-based algorithms discussed in this section, we apply a common post-processing step using edge-contraction operations. Recall from (11) that l is the minimum bound on the information

20. Recall that we assume that the mean of the true random vector $\mathbf{X}$ is known and equals to the zero vector so we do not need to subtract the empirical mean in (20).
21. We use $\mathbb{I}\{\cdot\}$ to denote the indicator function.

distances on edges. After learning the latent tree, if we find that there exists an edge $(i,h) \in W \times H$ with the estimated distance $\widehat{d_{ih}} < l$, then (i,h) is contracted to a single node whose label is i, that is, the hidden node h is removed and merged with node i. This edge contraction operation removes a hidden node if it is too close in information distances to another node. For Gaussian and binary variables, $\widehat{d_{ih}} = -\log|\widehat{\rho}_{ih}|$, so in our experiments, we use $l = -\log 0.9$ to contract an edge (i,h) if the correlation between the two nodes is higher than 0.9.

6.3 Relaxed Recursive Grouping (RG) Given Samples

We now show how to relax the canonical RG algorithm described in Section 4 to handle the case when only $\widehat{\mathbf{D}}$ is available. Recall that RG calls the TestNodeRelationships procedure recursively to ascertain child-parent and sibling relationships via equality tests $\Phi_{ijk} = d_{ik} - d_{jk}$ (cf., Section 3.2). These equality constraints are, in general, not satisfied with the estimated differences $\widehat{\Phi}_{ijk} := \widehat{d_{ik}} - \widehat{d_{jk}}$, which are computed based on the estimated distance in $\widehat{\mathbf{D}}$. Besides, not all estimated distances are equally accurate. Longer distance estimates (i.e., lower correlation estimates) are less accurate for a given number of samples.[22] As such, not all estimated distances can be used for testing inter-node relationships reliably. These observations motivate the following three modifications to the RG algorithm:

1. Consider using a smaller subset of nodes to test whether $\widehat{\Phi}_{ijk}$ is constant (across k).
2. Apply a threshold (inequality) test to the $\widehat{\Phi}_{ijk}$ values.
3. Improve on the robustness of the estimated distances $\widehat{d_{ih}}$ in (13) and (14) by averaging.

We now describe each of these modifications in greater detail. Firstly, in the relaxed RG algorithm, we only compute $\widehat{\Phi}_{ijk}$ for those estimated distances $\widehat{d_{ij}}$, $\widehat{d_{ik}}$ and $\widehat{d_{jk}}$ that are below a prescribed threshold $\tau > 0$ since longer distance estimates are unreliable. As such, for each pair of nodes (i, j) such that $\widehat{d_{ij}} < \tau$, associate the set

$$\mathcal{K}_{ij} := \left\{ k \in V\backslash\{i, j\} : \max\{\widehat{d_{ik}}, \widehat{d_{jk}}\} < \tau \right\}. \tag{21}$$

This is the subset of nodes in V whose estimated distances to i and j are less than τ. Compute $\widehat{\Phi}_{ijk}$ for all $k \in \mathcal{K}_{ij}$ only.

Secondly, instead of using equality tests in TestNodeRelationships to determine the relationship between nodes i and j, we relax this test and consider the statistic

$$\widehat{\Lambda}_{ij} := \max_{k\in\mathcal{K}_{ij}} \widehat{\Phi}_{ijk} - \min_{k\in\mathcal{K}_{ij}} \widehat{\Phi}_{ijk} \tag{22}$$

Intuitively, if $\widehat{\Lambda}_{ij}$ in (22) is close to zero, then nodes i and j are likely to be in the same family. Thus, declare that nodes $i, j \in V$ are in the same family if

$$\widehat{\Lambda}_{ij} < \varepsilon, \tag{23}$$

22. In fact, by using a large deviation result in Shen (2007, Theorem 1), we can formally show that a larger number of samples is required to get a good approximation of ρ_{ik} if it is small compared to when ρ_{ik} is large.

for another threshold $\varepsilon > 0$. Similarly, an observed node k is identified as a parent node if $|\widehat{d_{ik}} + \widehat{d_{kj}} - \widehat{d_{ij}}| < \varepsilon$ for all i and j in the same family. If such an observed node does not exists for a group of family, then a new hidden node is introduced as the parent node for the group.

Thirdly, in order to further improve on the quality of the distance estimate $\widehat{d_{ih}}$ of a newly introduced hidden node to observed nodes, we compute $\widehat{d_{ih}}$ using (13) with different pairs of $j \in \mathcal{C}(h)$ and $k \in \mathcal{K}_{ij}$, and take the average as follows:

$$\widehat{d_{ih}} = \frac{1}{2(|\mathcal{C}(h)|-1)} \left(\sum_{j \in \mathcal{C}(h)} \widehat{d_{ij}} + \frac{1}{|\mathcal{K}_{ij}|} \sum_{k \in \mathcal{K}_{ij}} \widehat{\Phi}_{ijk} \right). \tag{24}$$

Similarly, for any other node $k \notin \mathcal{C}(h)$, we compute $\widehat{d_{kh}}$ using all child nodes in $\mathcal{C}(h)$ and $\mathcal{C}(k)$ (if $\mathcal{C}(k) \neq \emptyset$) as follows:

$$\widehat{d_{kh}} = \begin{cases} \frac{1}{|\mathcal{C}(h)|} \sum_{i \in \mathcal{C}(h)} (\widehat{d_{ik}} - \widehat{d_{ih}}), & \text{if } k \in V, \\ \frac{1}{|\mathcal{C}(h)||\mathcal{C}(k)|} \sum_{(i,j) \in \mathcal{C}(h) \times \mathcal{C}(k)} (\widehat{d_{ij}} - \widehat{d_{ih}} - \widehat{d_{jk}}), & \text{otherwise.} \end{cases} \tag{25}$$

It is easy to verify that if $\widehat{d_{ih}}$ and $\widehat{d_{kh}}$ are equal to d_{ih} and d_{kh} respectively, then (24) and (25) reduce to (13) and (14) respectively.

The following theorem shows that relaxed RG is consistent, and with appropriately chosen thresholds ε and τ, it has the sample complexity logarithmic in the number of observed variables. The proof follows from standard Chernoff bounds and is provided in Appendix A.6.

Theorem 11 (Consistency and Sample Complexity of Relaxed RG) *(i) Relaxed RG is structurally consistent for all $T_p \in \mathcal{T}_{\geq 3}$. In addition, it is risk consistent for Gaussian and symmetric discrete distributions. (ii) Assume that the effective depth is $\delta(T_p;V) = O(1)$ (i.e., constant in m) and relaxed RG is used to reconstruct the tree given $\widehat{\mathbf{D}}$. For every $\eta > 0$, there exists thresholds $\varepsilon, \tau > 0$ such that if*

$$n > C \log(m/\sqrt[3]{\eta}) \tag{26}$$

for some constant $C > 0$, the error probability for structure reconstruction in (5) *is bounded above by η. If, in addition, p is a Gaussian or symmetric discrete distribution and $n > C' \log(m/\sqrt[3]{\eta})$, the error probability for distribution reconstruction in* (6) *is also bounded above by η. Thus, the sample complexity of relaxed RG, which is the number of samples required to achieve a desired level of accuracy, is logarithmic in m, the number of observed variables.*

As we observe from (26), the sample complexity for RG is logarithmic in m for shallow trees (i.e., trees where the effective depth is constant). This is in contrast to NJ where the sample complexity is super-polynomial in the number of observed nodes for the HMM (St. John et al., 2003; Lacey and Chang, 2006).

6.3.1 RG WITH *k*-MEANS CLUSTERING

In practice, if the number of samples is limited, the distance estimates $\widehat{d_{ij}}$ are noisy and it is difficult to select the threshold ε in Theorem 11 to identify sibling nodes reliably. In our experiments, we employ a modified version of the k-means clustering algorithm to cluster a set of nodes with small $\widehat{\Lambda}_{ij}$, defined in (22), as a group of siblings. Recall that we test each $\widehat{\Lambda}_{ij}$ locally with a fixed threshold ε in (23). In contrast, the k-means algorithm provides a *global* scheme and circumvents the need to select the threshold ε. We adopt the *silhouette method* (Rousseeuw, 1987) with dissimilarity measure $\widehat{\Lambda}_{ij}$ to select optimal the number of clusters k.

6.4 Relaxed Neighbor-Joining Given Samples

In this section, we describe how NJ can be relaxed when the true distances are unavailable. We relax the NJ algorithm by using ML estimates of the distances $\widehat{d_{ij}}$ in place of unavailable distances d_{ij}. NJ typically assume that all observed nodes are at the leaves of the latent tree, so after learning the latent tree, we perform the post-processing step described in Section 6.2 to identify internal nodes that are observed.[23] The sample complexity of NJ is known to be $O(\exp(\text{diam}(T_p))\log m)$ (St. John et al., 2003) and thus does not scale well when the latent tree T_p has a large diameter. Comparisons between the sample complexities of other closely related latent tree learning algorithms are discussed in Atteson (1999), Erdős et al. (1999), Csűrös (2000) and St. John et al. (2003).

6.5 Relaxed CLGrouping Given Samples

In this section, we discuss how to modify CLGrouping (CLRG and CLNG) when we only have access to the estimated information distance $\widehat{\mathbf{D}}$. The relaxed version of CLGrouping differs from CLGrouping in two main aspects. Firstly, we replace the edge weights in the construction of the MST in (17) with the estimated information distances $\widehat{d_{ij}}$, that is,

$$\widehat{T_{\text{CL}}} = \text{MST}(V;\widehat{\mathbf{D}}) := \underset{T\in\mathcal{T}(V)}{\text{argmin}} \sum_{(i,j)\in T} \widehat{d_{ij}}. \tag{27}$$

The procedure in (27) can be shown to be equivalent to the learning of the ML tree structure given samples $\mathbf{x}_V^n$ if p_V is a Gaussian or symmetric discrete distribution.[24] It has also been shown that the error probability of structure learning $\Pr(\widehat{T_{\text{CL}}} \neq T_{\text{CL}})$ converges to zero exponentially fast in the number of samples n for both discrete and Gaussian data (Tan et al., 2010, 2011). Secondly, for CLRG (respectively CLNJ), we replace RG (respectively NJ) with the relaxed version of RG (respectively NJ). The sample complexity result of CLRG (and its proof) is similar to Theorem 11 and the proof is provided in Appendix A.7.

Theorem 12 (Consistency and Sample Complexity of Relaxed CLRG) *(i) Relaxed CLRG is structurally consistent for all $T_p \in \mathcal{T}_{\geq 3}$. In addition, it is risk consistent for Gaussian and symmetric discrete distributions. (ii) Assume that the effective depth is $\delta(T_p;V) = O(1)$ (i.e., constant in m). Then the sample complexity of relaxed CLRG is logarithmic in m.*

6.6 Regularized CLGrouping for Learning Latent Tree Approximations

For many practical applications, it is of interest to learn a latent tree that *approximates* the given empirical distribution. In general, introducing more hidden variables enables better fitting to the empirical distribution, but it increases the model complexity and may lead to overfitting. The Bayesian Information Criterion (Schwarz, 1978) provides a trade-off between model fitting and model complexity, and is defined as follows:

$$\text{BIC}(\widehat{T}) = \log p(\mathbf{x}_V^n;\widehat{T}) - \frac{\kappa(\widehat{T})}{2}\log n \tag{28}$$

23. The processing (contraction) of the internal nodes can be done in any order.

24. This follows from the observation that the ML search for the optimal structure is equivalent to the KL-divergence minimization problem in (15) with p_V replaced by $\widehat{p}_V$, the empirical distribution of $\mathbf{x}_V^n$.

where $\widehat{T}$ is a latent tree structure and $\kappa(\widehat{T})$ is the number of free parameters, which grows linearly with the number of hidden variables because $\widehat{T}$ is a tree. Here, we describe *regularized CLGrouping*, in which we use the BIC in (28) to specify a stopping criterion on the number of hidden variables added.

For each internal node and its neighbors in the Chow-Liu tree, we use relaxed NJ or RG to learn a latent subtree. Unlike in regular CLGrouping, before we integrate this subtree into our model, we compute its BIC score. Computing the BIC score requires estimating the maximum likelihood parameters for the models, so for general discrete distributions, we run the EM algorithm on the subtree to estimate the parameters.[25] After we compute the BIC scores for all subtrees corresponding to all internal nodes in the Chow-Liu tree, we choose the subtree that results in the highest BIC score and incorporate that subtree into the current tree model.

The BIC score can be computed efficiently on a tree model with a few hidden variables. Thus, for computational efficiency, each time a set of hidden nodes is added to the model, we generate samples of hidden nodes conditioned on the samples of observed nodes, and use these augmented samples to compute the BIC score approximately when we evaluate the next subtree to be integrated in the model.

If none of the subtrees increases the BIC score (i.e., the current tree has the highest BIC score), the procedure stops and outputs the estimated latent tree. Alternatively, if we wish to learn a latent tree with a given number of hidden nodes, we can used the BIC-based procedure mentioned in the previous paragraph to learn subtrees until the desired number of hidden nodes is introduced. Depending on whether we use NJ or RG as the subroutine, we denote the specific regularized CLGrouping algorithm as *regCLNJ* or *regCLRG*.

This approach of using an approximation of the BIC score has been commonly used to learn a graphical model with hidden variables (Elidan and Friedman, 2005; Zhang and Kočka, 2004). However, for these algorithms, the BIC score needs to be evaluated for a large subset of nodes, whereas in CLGrouping, the Chow-Liu tree among observed variables prunes out many subsets, so we need to evaluate BIC scores only for a small number of candidate subsets (the number of internal nodes in the Chow-Liu tree).

7. Experimental Results

In this section, we compare the performances of various latent tree learning algorithms. We first show simulation results on synthetic data sets with known latent tree structures to demonstrate the consistency of our algorithms. We also analyze the performance of these algorithms when we change the underlying latent tree structures. Then, we show that our algorithms can approximate arbitrary multivariate probability distributions with latent trees by applying them to two real-world data sets, a monthly stock returns example and the 20 newsgroups data set.

7.1 Simulations using Synthetic Data Sets

In order to analyze the performances of different tree reconstruction algorithms, we generate samples from known latent tree structures with varying sample sizes and apply reconstruction algorithms. We compare the neighbor-joining method (NJ) (Saitou and Nei, 1987) with recursive

25. Note that for Gaussian and symmetric discrete distributions, the model parameters can be recovered from information distances directly using (8) or (10).

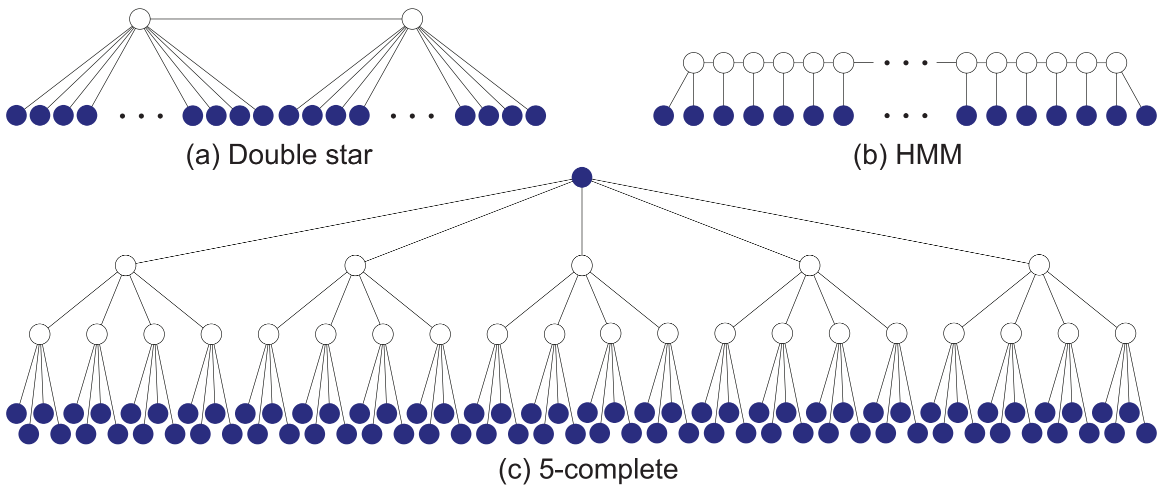


Figure 6: Latent tree structures used in our simulations.

grouping (RG), Chow-Liu Neighbor Joining (CLNJ), and Chow-Liu Recursive Grouping (CLRG). Since the algorithms are given only samples of observed variables, we use the sample-based algorithms described in Section 6. For all our experiments, we use the same edge-contraction threshold $\varepsilon' = -\log 0.9$ (see Sections 6.4 and 6.5), and set τ in Section 6.3 to grow logarithmically with the number of samples.

Figure 6 shows the three latent tree structures used in our simulations. The double-star has 2 hidden and 80 observed nodes, the HMM has 78 hidden and 80 observed nodes, and the 5-complete tree has 25 hidden and 81 observed nodes including the root node. For simplicity, we present simulation results only on Gaussian models but note that the behavior on discrete models is similar. All correlation coefficients on the edges ρ_{ij} were independently drawn from a uniform distribution supported on $[0.2, 0.8]$. The performance of each method is measured by averaging over 200 independent runs with different parameters. We use the following performance metrics to quantify the performance of each algorithm in Figure 7:

(i) **Structure recovery error rate**: This is the proportion of times that the proposed algorithm fails to recover the true latent tree structure. Note that this is a very strict measure since even a single wrong hidden node or misplaced edge results in an error for the entire structure.

(ii) **Robinson Foulds metric** (Robinson and Foulds, 1981): This popular phylogenetic tree-distortion metric computes the number of graph transformations (edge contraction or expansion) needed to be applied to the estimated graph in order to get the correct structure. This metric quantifies the difference in the structures of the estimated and true models.

(iii) **Error in the number of hidden variables**: We compute the average number of hidden variables introduced by each method and plot the absolute difference between the average estimated hidden variables and the number of hidden variables in the true structure.

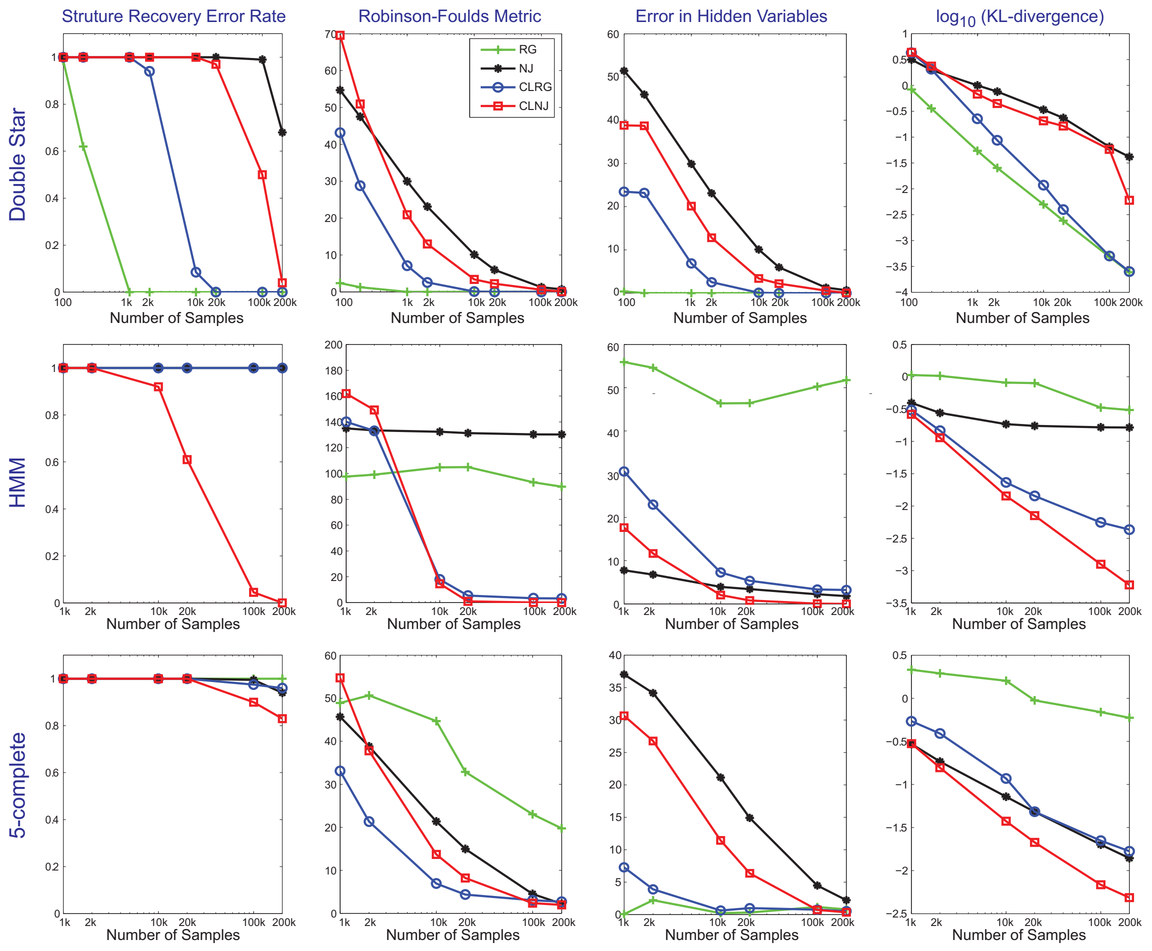


Figure 7: Performance of RG, NJ, CLRG, and CLNJ for the latent trees shown in Figure 6.

(iv) **KL-divergence** $D(p_V \,||\, \widehat{p}^n_V)$: This is a measure of the distance between the estimated and the true models over the set of observed nodes V.[26]

We first note that from the structural error rate plots that the double star is the easiest structure to recover and the 5-complete tree is the hardest. In general, given the same number of observed variables, a latent tree with more hidden variables or larger effective depth (see Section 2) is more difficult to recover.

For the double star, RG clearly outperforms all other methods. With only 1,000 samples, it recovers the true structure exactly in all 200 runs. On the other hand, CLGrouping performs significantly better than RG for the HMM. There are two reasons for such performance differences. Firstly, for Gaussian distributions, it was shown (Tan et al., 2010) that given the same number of variables and their samples, the Chow-Liu algorithm is most accurate for a chain and least accurate for a star. Since the Chow-Liu tree of a latent double star graph is close to a star, and the Chow-Liu

26. Note that this is not the same quantity as in (6) because if the number of hidden variables is estimated incorrectly, $D(p\,||\,\widehat{p}^n)$ is infinite so we plot $D(p_V\,||\,\widehat{p}^n_V)$ instead. However, for Gaussian and symmetric discrete distributions, $D(p\,||\,\widehat{p}^n)$ converges to zero in probability since the number of hidden variables is estimated correctly asymptotically.

	RG	NJ	CLRG	CLNJ
HMM	10.16	0.02	0.10	0.05
5-complete	7.91	0.02	0.26	0.06
Double star	1.43	0.01	0.76	0.20

Table 2: Average running time of each algorithm in seconds.

tree of a latent HMM is close to a chain, the Chow-Liu tree tend to be more accurate for the HMM than for the double star. Secondly, the internal nodes in the Chow-Liu tree of the HMM tend to have small degrees, so we can apply RG or NJ to a very small neighborhood, which results in a significant improvement in both accuracy and computational complexity.

Note that NJ is particularly poor at recovering the HMM structure. In fact, it has been shown that even if the number of samples grows polynomially with the number of observed variables (i.e., $n = O(m^B)$ for any $B > 0$), it is insufficient for NJ to recover HMM structures (Lacey and Chang, 2006). The 5-complete tree has two layers of hidden nodes, making it very difficult to recover the exact structure using any method. CLNJ has the best structure recovery error rate and KL divergence, while CLRG has the smallest Robinson-Foulds metric.

Table 2 shows the running time of each algorithm averaged over 200 runs and all sample sizes. All algorithms are implemented in MATLAB. As expected, we observe that CLRG is significantly faster than RG for HMM and 5-complete graphs. NJ is fastest, but CLNJ is also very efficient and leads to much more accurate reconstruction of latent trees.

Based on the simulation results, we conclude that for a latent tree with a few hidden variables, RG is most accurate, and for a latent tree with a large diameter, CLNJ performs the best. A latent tree with multiple layers of hidden variables is more difficult to recover correctly using any method, and CLNJ and CLRG outperform NJ and RG.

7.2 Monthly Stock Returns

In this and the next section, we test our algorithms on real-world data sets. The probability distributions that govern these data sets of course do not satisfy the assumptions required for consistent learning of the latent tree models. Nonetheless the experiments here demonstrate that our algorithms are also useful in *approximating* complex probability distributions by latent models in which the hidden variables have the same domain as the observed ones.

We apply our latent tree learning algorithms to model the dependency structure of monthly stock returns of 84 companies in the S&P 100 stock index.[27] We use the samples of the monthly returns from 1990 to 2007. As shown in Table 3 and Figure 8, CLNJ achieves the highest log-likelihood and BIC scores. NJ introduces more hidden variables than CLNJ and has lower log-likelihoods, which implies that starting from a Chow-Liu tree helps to get a better latent tree approximation. Figure 11 shows the latent tree structure learned using the CLNJ method. Each observed node is labeled with the ticker of the company. Note that related companies are closely located on the tree. Many hidden nodes can be interpreted as industries or divisions. For example, h1 has Verizon, Sprint, and T-mobile as descendants, and can be interpreted as the telecom industry, and h3 correspond to the technology division with companies such as Microsoft, Apple, and IBM as descendants. Nodes h26 and h27 group commercial banks together, and h25 has all retail stores as child nodes.

27. We disregard 16 companies that have been listed on S&P 100 only after 1990.

	Log-Likelihood	BIC	# Hidden	# Parameters	Time (secs)
CL	-13,321	-13,547	0	84	0.15
NJ	-12,400	-12,747	45	129	**0.02**
RG	-14,042	-14,300	12	96	21.15
CLNJ	**-11,990**	**-12,294**	29	113	0.24
CLRG	-12,879	-13,174	26	110	0.40

Table 3: Comparison of the log-likelihood, BIC, number of hidden variables introduced, number of parameters, and running time for the monthly stock returns example.

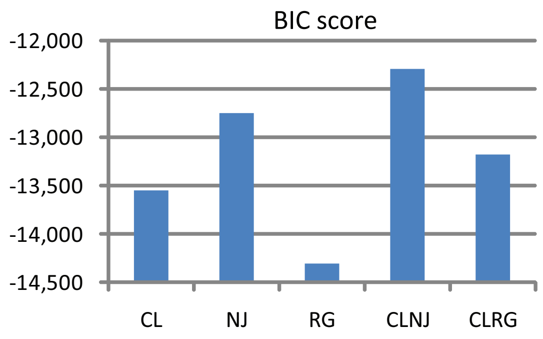


Figure 8: Plot of BIC scores for the monthly stock returns example.

7.3 20 Newsgroups with 100 Words

For our last experiment, we apply our latent tree learning algorithms to the 20 Newsgroups data set with 100 words.[28] The data set consists of 16,242 binary samples of 100 words, indicating whether each word appears in each posting or not. In addition to the Chow-Liu tree (CL), NJ, RG, CLNJ, and CLRG, we also compare the performances with the regCLNJ and regCLRG (described in Section 6.6), the latent cluster model (LCM) (Lazarsfeld and Henry, 1968), and BIN, which is a greedy algorithm for learning latent trees (Harmeling and Williams, 2010).

Table 4 shows the performance of different algorithms, and Figure 9 plots the BIC score. We use the MATLAB code (a small part of it is implemented in C) provided by Harmeling and Williams (2010)[29] to run LCM and BIN. Note that although LCM has only one hidden node, the hidden node has 16 states, resulting in many parameters. We also tried to run the algorithm by Chen et al. (2008), but their JAVA implementation on this data set did not complete even after several days. For NJ, RG, CLNJ, and CLRG, we learned the structures using only information distances (defined in (9)) and then used the EM algorithm to fit the parameters. For regCLNJ and regCLRG, the model parameters are learned during the structure learning procedure by running the EM algorithm locally, and once the structure learning is over, we refine the parameters by running the EM algorithm for the entire latent tree. All methods are implemented in MATLAB except the E-step of the EM algorithm, which is implemented in C++.

28. The data set can be found at `http://cs.nyu.edu/~roweis/data/20news_w100.mat`.
29. Code can be found at `http://people.kyb.tuebingen.mpg.de/harmeling/code/ltt-1.3.tar`.

	Log-Likelihood	BIC	Hidden	Params	Time (s)		
					Total	Structure	EM
CL	-238,713	-239,677	0	199	8.9	-	-
LCM	**-223,096**	**-230,925**	1	1,615	8,835.9	-	-
BIN	-232,042	-233,952	98	394	3,022.6	-	-
NJ	-230,575	-232,257	74	347	1,611.2	3.3	1,608.2
RG	-239,619	-240,875	30	259	927.1	30.8	896.4
CLNJ	-230,858	-232,540	74	347	1,479.6	2.7	1,476.8
CLRG	-231,279	-232,738	51	301	1,224.6	3.1	1,224.6
regCLNJ	-235,326	-236,553	27	253	630.8	449.7	181.1
regCLRG	-234,012	-235,229	26	251	606.9	493.0	113.9

Table 4: Comparison between various algorithms on the newsgroup set.

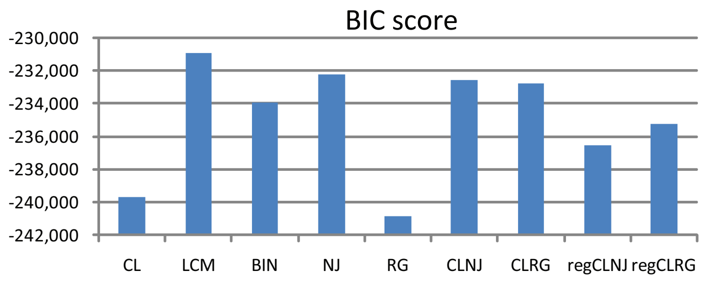


Figure 9: The BIC scores of various algorithms on the newsgroup set.

Despite having many parameters, the models learned via LCM have the best BIC score. However, it does not reveal any interesting structure and is computationally more expensive to learn. In addition, it may result in overfitting. In order to show this, we split the data set randomly and use half as the training set and the other half as the test set. Table 5 shows the performance of applying the latent trees learned from the training set to the test set, and Figure 10 shows the log-likelihood on the training and the test sets. For LCM, the test log-likelihood drops significantly compared to the training log-likelihood, indicating that LCM is overfitting the training data. NJ, CLNJ, and CLRG achieve high log-likelihood scores on the test set. Although regCLNJ and regCLRG do not result in a better BIC score, they introduce fewer hidden variables, which is desirable if we wish to learn a latent tree with small computational complexity, or if we wish to discover a few hidden variables that are meaningful in explaining the dependencies of observed variables.

Figure 12 shows the latent tree structure learned using regCLRG from the entire data set. Many hidden variables in the tree can be roughly interpreted as topics—h5 as sports, h9 as computer technology, h13 as medical, etc. Note that some words have multiple meanings and appear in different topics—for example, `program` can be used in the phrase “space program” as well as “computer program”, and `win` may indicate the windows operating system or winning in sports games.

	Train		Test		Hidden	Params	Time (s)		
	Log-Like	BIC	Log-Like	BIC			Total	Struct	EM
CL	-119,013	-119,909	-120,107	-121,003	0	199	3.0	-	-
LCM	**-112,746**	-117,288	-116,884	-120,949	1	1,009	3,197.7	-	-
BIN	-117,172	-118,675	-117,957	-119,460	78	334	1,331.3	-	-
NJ	-115,319	**-116,908**	**-116,011**	-117,600	77	353	802.8	1.3	801.5
RG	-118,280	-119,248	-119,181	-120,149	8	215	137.6	7.6	130.0
CLNJ	-115,372	-116,987	-116,036	-117,652	80	359	648.0	1.5	646.5
CLRG	-115,565	-116,920	-116,199	**-117,554**	51	301	506.0	1.7	504.3
regCLNJ	-117,723	-118,924	-118,606	-119,808	34	267	425.5	251.3	174.2
regCLRG	-116,980	-118,119	-117,652	-118,791	27	253	285.7	236.5	49.2

Table 5: Comparison between various algorithms on the newsgroup data set with a train/test split.

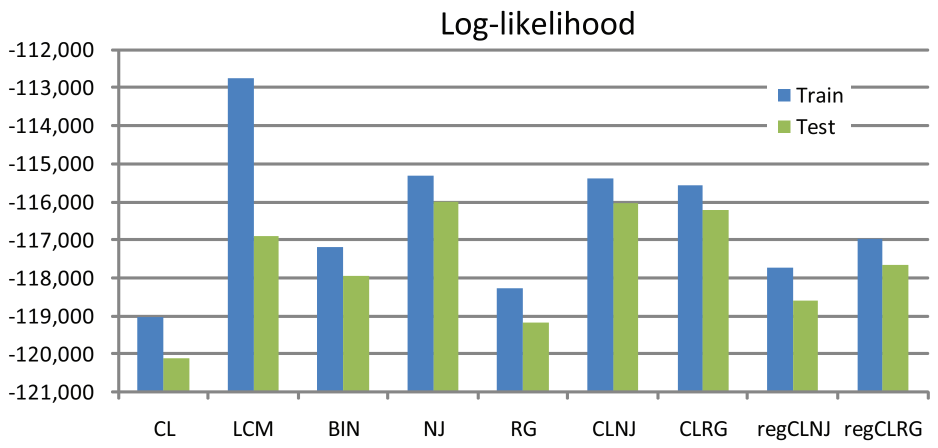


Figure 10: Train and test log-likelihood scores of various algorithms on the newsgroup data set with a train/test split.

8. Discussion and Conclusion

In this paper, we proposed algorithms to learn a latent tree model from the information distances of observed variables. Our first algorithm, recursive grouping (RG), identifies sibling and parent-child relationships and introduces hidden nodes recursively. Our second algorithm, CLGrouping, maintains a tree in each iteration and adds hidden variables by locally applying latent-tree learning procedures such as recursive grouping. These algorithms are structurally consistent (and risk consistent as well in the case of Gaussian and discrete symmetric distributions), and have sample complexity logarithmic in the number of observed variables for constant depth trees.

Using simulations on synthetic data sets, we showed that RG performs well when the number of hidden variables is small, while CLGrouping performs significantly better than other algorithms when there are many hidden variables in the latent tree. We compared our algorithms to other EM-based approaches and the neighbor-joining method on real-world data sets, under both Gaussian and discrete data modeling. Our proposed algorithms show superior results in both accuracy (measured by KL-divergence and graph distance) and computational efficiency. In addition, we introduced regularized CLGrouping, which can learn a latent tree approximation by trading off model complexity (number of hidden nodes) with data fidelity. This is very relevant for

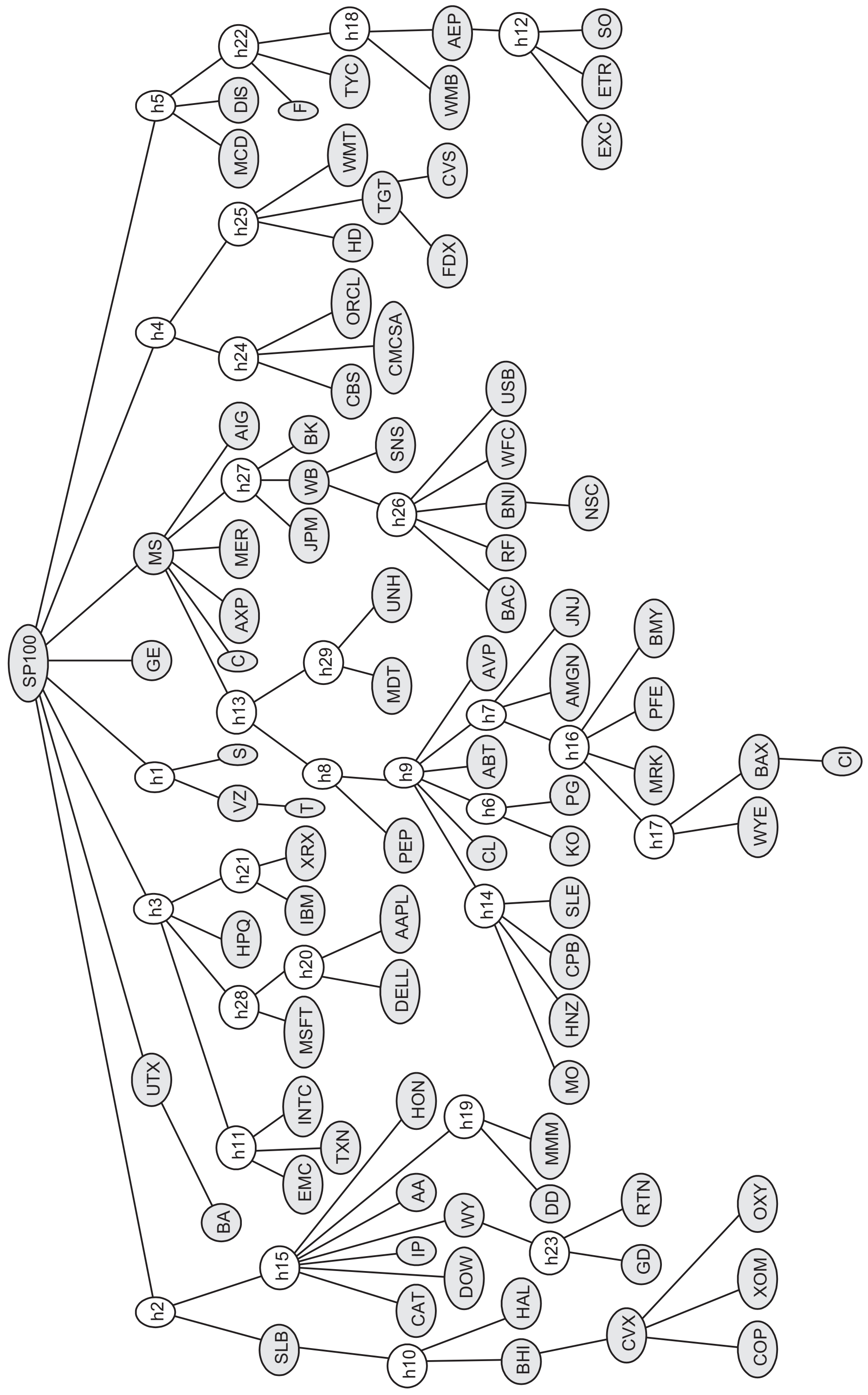


Figure 11: Tree structure learned from monthly stock returns using CLNJ.

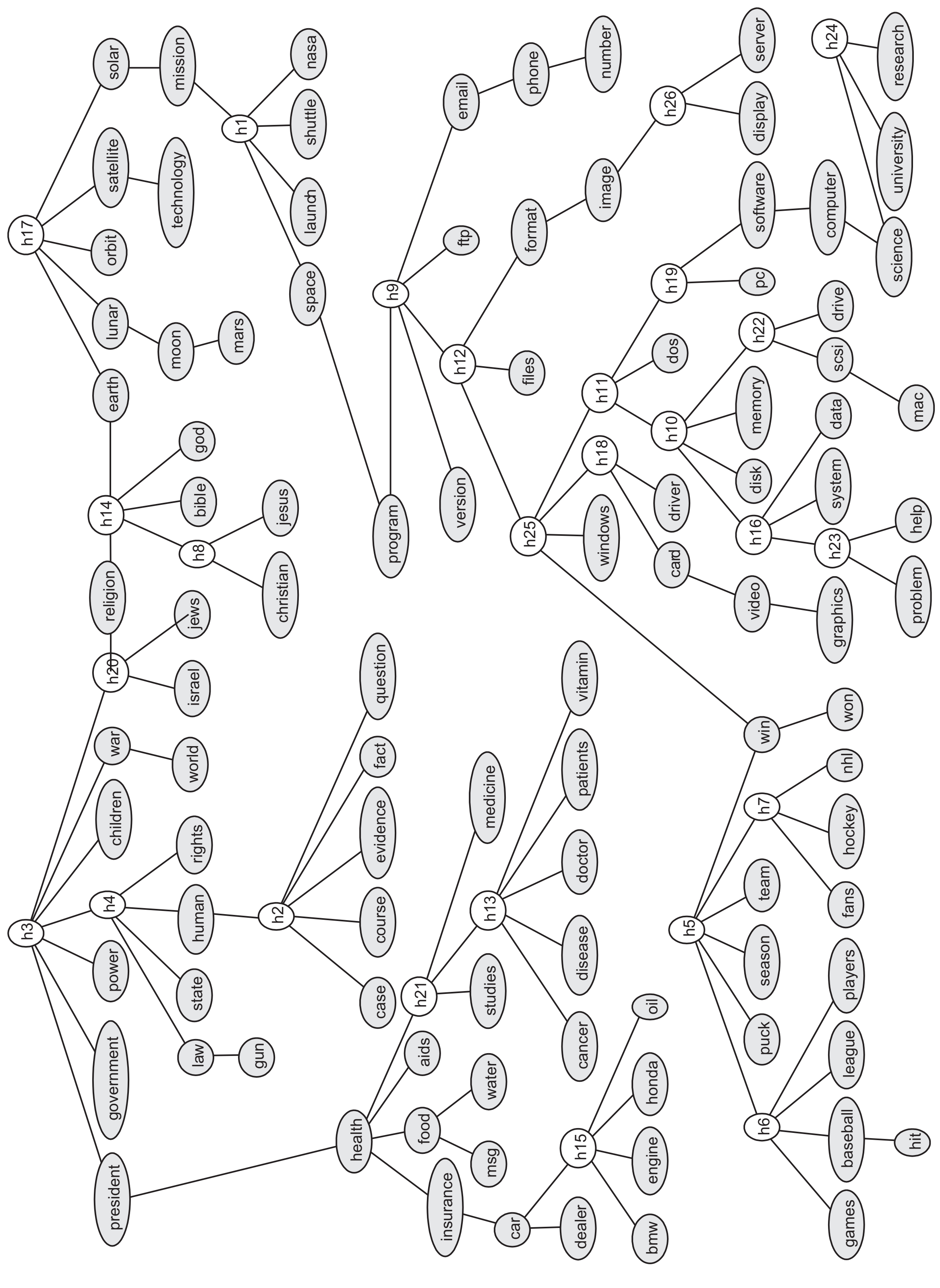


Figure 12: Tree structure learned from 20 newsgroup data set using regCLRG.

practical implementation on real-world data sets. In future, we plan to develop a unified framework for learning latent trees where each random variable (node) may be continuous or discrete. The MATLAB implementation of our algorithms can be downloaded from the project webpage `http://people.csail.mit.edu/myungjin/latentTree.html`.

Acknowledgments

This research was supported in part by Shell International Exploration and Production, Inc. and in part by the Air Force Office of Scientific Research under Award No. FA9550-06-1-0324. This work was also supported in part by AFOSR under Grant FA9550-08-1-1080 and in part by MURI under AFOSR Grant FA9550-06-1-0324. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the Air Force. Vincent Tan and Animashree Anandkumar are supported by A*STAR, Singapore and by the setup funds at U.C. Irvine respectively.

Appendix A. Proofs

In this appendix, we provide proofs for the theorems presented in the paper.

A.1 Proof of Lemma 4: Sibling Grouping

We prove statement (i) in Lemma 4 using (12) in Proposition 3. Statement (ii) follows along similar lines and its proof is omitted for brevity.

If : From the additive property of information distances in (12), if i is a leaf node and j is its parent, $d_{ik} = d_{ij} + d_{jk}$ and thus $\Phi_{ijk} = d_{ij}$ for all $k \neq i, j$.

Only If: Now assume that $\Phi_{ijk} = d_{ij}$ for all $k \in V \setminus \{i, j\}$. In order to prove that i is a leaf node and j is its parent, assume to the contrary, that i and j are not connected with an edge, then there exists a node $u \neq i, j$ on the path connecting i and j. If $u \in V$, then let $k = u$. Otherwise, let k be an observed node in the subtree away from i and j (see Figure 13(a)), which exists since $T_p \in \mathcal{T}_{\geq 3}$. By the additive property of information distances in (12) and the assumption that all distances are positive,

$$d_{ij} = d_{iu} + d_{uj} > d_{iu} - d_{uj} = d_{ik} - d_{kj} = \Phi_{ijk}$$

which is a contradiction. If i is not a leaf node in T_p, then there exist a node $u \neq i, j$ such that $(i, u) \in E_p$. Let $k = u$ if $u \in V$, otherwise, let k be an observed node in the subtree away from i and j (see Figure 13(b)). Then,

$$\Phi_{ijk} = d_{ik} - d_{jk} = -d_{ij} < d_{ij},$$

which is again a contradiction. Therefore, $(i, j) \in E_p$ and i is a leaf node. □

A.2 Proof of Theorem 5: Correctness and Computational Complexity of RG

The correctness of RG follows from the following observations: Firstly, from Proposition 3, for all i, j in the active set Y, the information distances d_{ij} can be computed exactly with Equations (13) and (14). Secondly, at each iteration of RG, the sibling groups within Y are identified correctly using the information distances by Lemma 4. Since the new parent node added to a partition that does not

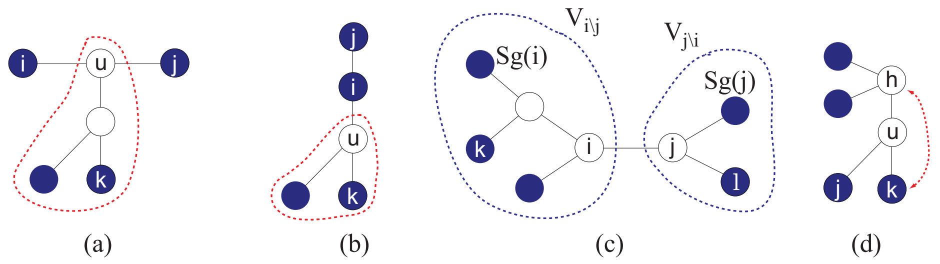


Figure 13: Shaded nodes indicate observed nodes and the rest indicate hidden nodes. (a),(b) Figures for Proof of Lemma 4. Dashed red line represent the subtrees away from i and j. (c) Figure for Proof of Lemma 8(i). (d) Figure for Proof of Lemma 8(iI)

contain an observed parent corresponds to a hidden node (in the original latent tree), a subforest of T_p is recovered at each iteration, and when $|Y| \leq 2$, and the entire latent tree is recovered.

The computational complexity follows from the fact there are a maximum of $O(m^3)$ differences $\Phi_{ijk} = d_{ik} - d_{jk}$ that we have to compute at each iteration of RG. Furthermore, there are at most $\text{diam}(T_p)$ subsets in the coarsest partition (cf., step 3) of Y at the first iteration, and the number of subsets reduce at least by 2 from one iteration to the next due to the assumption that $T_p \in \mathcal{T}_{\geq 3}$. This proves the claim that the computational complexity is upper bounded by $O(\text{diam}(T_p)m^3)$. □

A.3 Proof of Lemma 8: Properties of the MST

(i) For an edge $(i,j) \in E_p$ such that $\text{Sg}(i) \neq \text{Sg}(j)$, let $V_{i\setminus j} \subset V$ and $V_{j\setminus i} \subset V$ denote observed nodes in the subtrees obtained by the removal of edge (i,j), where the former includes node i and excludes node j and vice versa (see Figure 13(c)). Using part (ii) of the lemma and the fact that $\text{Sg}(i) \neq \text{Sg}(j)$, it can be shown that $\text{Sg}(i) \in V_{i\setminus j}$ and $\text{Sg}(j) \in V_{j\setminus i}$. Since (i,j) lies on the unique path from k to l on T_p, for all observed nodes $k \in V_{i\setminus j}, l \in V_{j\setminus i}$, we have

$$d_{kl} = d_{ki} + d_{ij} + d_{jl} \geq d_{\text{Sg}(i),i} + d_{ij} + d_{\text{Sg}(j),j} = d_{\text{Sg}(i),\text{Sg}(j)},$$

where the inequality is from the definition of surrogacy and the final equality uses the fact that $\text{Sg}(i) \neq \text{Sg}(j)$. By using the property of the MST that $(\text{Sg}(i),\text{Sg}(j))$ is the shortest edge from $V_{i\setminus j}$ to $V_{j\setminus i}$, we have (18).

(ii) First assume that we have a tie-breaking rule consistent across all hidden nodes so that if $d_{uh} = d_{vh} = \min_{i\in V} d_{ih}$ and $d_{uh'} = d_{vh'} = \min_{i\in V} d_{ih'}$ then both h and h' choose the same surrogate node. Let $j \in V, h \in \text{Sg}^{-1}(j)$, and let u be a node on the path connecting h and j (see Figure 13(d)). Assume that $\text{Sg}(u) = k \neq j$. If $d_{uj} > d_{uk}$, then

$$d_{hj} = d_{hu} + d_{uj} > d_{hu} + d_{uk} = d_{hk},$$

which is a contradiction since $j = \text{Sg}(h)$. If $d_{uj} = d_{uk}$, then $d_{hj} = d_{hk}$, which is again a contradiction to the consistent tie-breaking rule. Thus, the surrogate node of u is j.

(iii) First we claim that

$$|\text{Sg}^{-1}(i)| \leq \Delta(T_p)^{\frac{u}{l}\delta(T_p;V)}. \tag{29}$$

To prove this claim, let γ be the longest (worst-case) graph distance of any hidden node $h \in H$ from its surrogate, that is,

$$\gamma := \max_{h \in H} |\text{Path}(h, \text{Sg}(h); T_p)|. \tag{30}$$

From the degree bound, for each $i \in V$, there are at most $\Delta(T_p)^\gamma$ hidden nodes that are within the graph distance of γ,[30] so

$$|\text{Sg}^{-1}(i)| \leq \Delta(T_p)^\gamma \tag{31}$$

for all $i \in V$. Let $d^* := \max_{h \in H} d_{h,\text{Sg}(h)}$ be the longest (worst-case) information distance between a hidden node and its surrogate. From the bounds on the information distances, $l\gamma \leq d^*$. In addition, for each $h \in H$, let $z(h) := \text{argmin}_{j \in V} |\text{Path}((h, j); T_p)|$ be the observed node that is closest to h in graph distance. Then, by definition of the effective depth, $d_{h,\text{Sg}(h)} \leq d_{h,z(h)} \leq u\delta$ for all $h \in H$, and we have $d^* \leq u\delta$. Since $l\gamma \leq d^* \leq u\delta$, we also have

$$\gamma \leq u\delta / l. \tag{32}$$

Combining this result with (31) establishes the claim in (29). Now consider

$$\Delta(\text{MST}(V; \mathbf{D})) \overset{(a)}{\leq} \Delta(T_p) \max_{i \in V} |\text{Sg}^{-1}(i)| \overset{(b)}{\leq} \Delta(T_p)^{1 + \frac{u}{l}\delta(T_p; V)}$$

where (a) is a result of the application of (18) and (b) results from (29). This completes the proof of the claim in (19) in Lemma 8. $\square$

A.4 Proof of Theorem 9: Correctness and Computational Complexity of CLBlind

It suffices to show that the Chow-Liu tree $\text{MST}(V; \mathbf{d})$ is a transformation of the true latent tree T_p (with parameters such that $p \in \mathcal{P}(\mathcal{T}_{\text{blind}})$) as follows: contract the edge connecting each hidden variable h with its surrogate node $\text{Sg}(h)$ (one of its children and a leaf by assumption). Note that the blind transformation on the MST is merely the inverse mapping of the above. From (18), all the children of a hidden node h, except its surrogate $\text{Sg}(h)$, are neighbors of its surrogate node $\text{Sg}(h)$ in $\text{MST}(V; \mathbf{d})$. Moreover, these children of h which are not surrogates of any hidden nodes are leaf nodes in the MST. Similarly for two hidden nodes $h_1, h_2 \in H$ such that $(h_1, h_2) \in E_p$, $(\text{Sg}(h_1), \text{Sg}(h_2)) \in \text{MST}(V; \mathbf{d})$ from Lemma 8(i). Hence, CLBlind outputs the correct tree structure T_p. The computational complexity follows from the fact that the blind transformation is linear in the number of internal nodes, which is less than the number of observed nodes, and that learning the Chow-Liu tree takes $O(m^2 \log m)$ operations. $\square$

A.5 Proof of Theorem 10: Correctness and Computational Complexity of CLRG

We first define some new notations.

Notation: Let $I := V \setminus \text{Leaf}(\text{MST}(V; \mathbf{d}))$ be the set of internal nodes. Let $v^r \in I$ be the internal node visited at iteration r, and let H^r be all hidden nodes in the inverse surrogate set $\text{Sg}^{-1}(v^r)$, that is, $H^r = \text{Sg}^{-1}(v^r) \setminus \{v^r\}$. Let $A^r := \text{nbd}[v^r; T^{r-1}]$, and hence A^r is the node set input to the recursive grouping routine at iteration r, and let $\text{RG}(A^r, \mathbf{d})$ be the output latent tree learned by recursive grouping. Define T^r as the tree output at the end of r iterations of CLGrouping. Let $V^r := \{v^{r+1}, v^{r+2}, \ldots, v^{|I|}\}$ be the set of internal nodes that have not yet been visited by CLGrouping

30. The maximum size of the inverse surrogate set in (30) is attained by a $\Delta(T_p)$-ary complete tree.

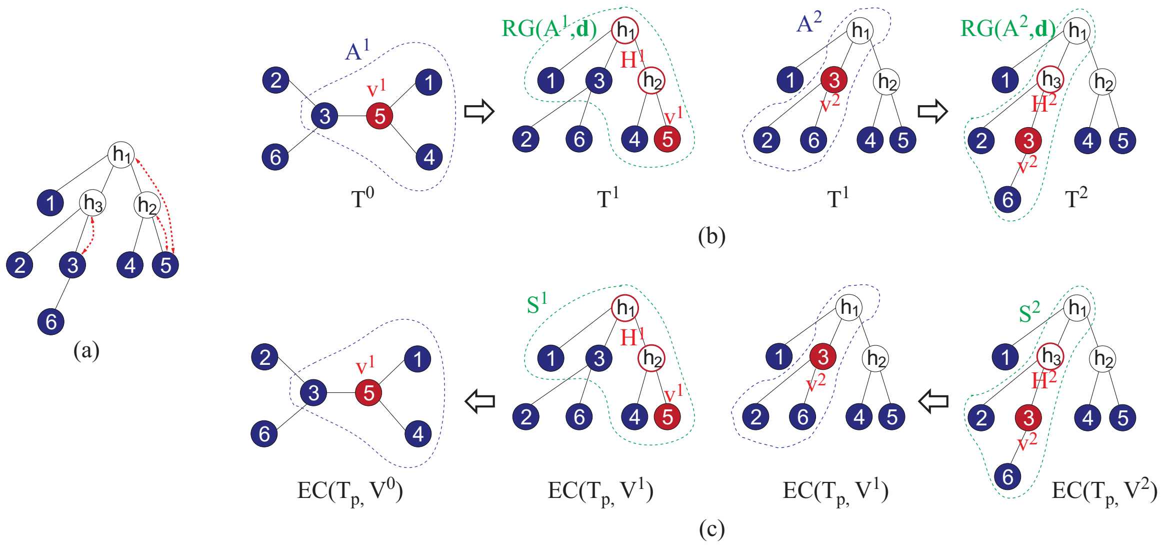


Figure 14: Figure for Proof of Theorem 10. (a) Original latent tree. (b) Illustration of CLGrouping. (c) Illustration of the trees constructed using edge contractions.

at the end of r iterations. Let $\mathrm{EC}(T_p, V^r)$ be the tree constructed using edge contractions as follows: in the latent tree T_p, we contract edges corresponding to each node $u \in V^r$ and all hidden nodes in its inverse surrogate set $\mathrm{Sg}^{-1}(u)$. Let S^r be a subtree of $\mathrm{EC}(T_p, V^r)$ spanning v^r, H^r and their neighbors.

For example, in Figure 14, the original latent tree T_p is shown in Figure 14(a), and T^0, T^1, T^2 are shown in Figure 14(b). The set of internal nodes is $I = \{3, 5\}$. In the first iteration, $v^1 = 5$, $A^1 = \{1, 3, 4, 5\}$ and $H^1 = \{h_1, h_2\}$. In the second iteration, $v^2 = 3$, $A^2 = \{2, 3, 6, h_1\}$ and $H^1 = \{h_3\}$. $V^0 = \{3, 5\}$, $V^1 = \{3\}$, and $V^2 = \emptyset$, and in Figure 14(c), we show $\mathrm{EC}(T_p, V^0)$, $\mathrm{EC}(T_p, V^1)$, and $\mathrm{EC}(T_p, V^2)$. In $\mathrm{EC}(T_p, V^1)$, S^1 is the subtree spanning $5, h_1, h_2$ and their neighbors, that is, $\{1, 3, 4, 5, h_1, h_2\}$. In $\mathrm{EC}(T_p, V^2)$, S^2 is the subtree spanning $3, h_3$ and their neighbors, that is, $\{2, 3, 6, h_1, h_3\}$. Note that $T^0 = \mathrm{EC}(T_p, V^0)$, $T^1 = \mathrm{EC}(T_p, V^1)$, and $T^2 = \mathrm{EC}(T_p, V^2)$; we show below that this holds for all CLGrouping iterations in general.

We prove the theorem by induction on the iterations $r = 1, \ldots, |I|$ of the CLGrouping algorithm.

Induction Hypothesis: At the end of k iterations of CLGrouping, the tree obtained is

$$T^k = \mathrm{EC}(T_p, V^k), \qquad \forall k = 0, 1, \ldots, |I|. \tag{33}$$

In words, the latent tree after k iterations of CLGrouping can be constructed by contracting each surrogate node in T_p that has not been visited by CLGrouping with its inverse surrogate set. Note that $V^{|I|} = \emptyset$ and $\mathrm{EC}(T_p, V^{|I|})$ is equivalent to the original latent tree T_p. Thus, if the above induction in (33) holds, then the output of CLGrouping $T^{|I|}$ is the original latent tree.

Base Step $r = 0$: The claim in (33) holds since $V^0 = I$ and the input to the CLGrouping procedure is the Chow-Liu tree $\mathrm{MST}(V; \mathbf{D})$, which is obtained by contracting all surrogate nodes and their inverse surrogate sets (see Section 5.2).

Induction Step: Assume (33) is true for $k = 1, \ldots, r-1$. Now consider $k = r$.

We first compare the two latent trees $\mathrm{EC}(T_p, V^r)$ and $\mathrm{EC}(T_p, V^{r-1})$. By the definition of EC, if we contract edges with v^r and the hidden nodes in its inverse surrogate set H^r on the tree $\mathrm{EC}(T_p, V^r)$, then we obtain $\mathrm{EC}(T_p, V^{r-1})$, which is equivalent to T^{r-1} by the induction assumption. Note that as shown in Figure 14, this transformation is local to the subtree S^r: contracting v^r with H^r on $\mathrm{EC}(T_p, V^r)$ transforms S^r into a star graph with v^r at its center and the hidden nodes H^r removed (contracted with v^r).

Recall that the CLGrouping procedure replaces the induced subtree of A^r in T^{r-1} (which is precisely the star graph mentioned above by the induction hypothesis) with $\mathrm{RG}(A^r, \mathbf{d})$ to obtain T^r. Thus, to prove that $T^r = \mathrm{EC}(T_p, V^r)$, we only need to show that RG reverses the edge-contraction operations on v^r and H^r, that is, the subtree $S^r = \mathrm{RG}(A^r, \mathbf{d})$. We first show that $S^r \in \mathcal{T}_{\geq 3}$, that is, it is identifiable (minimal) when A^r is the set of visible nodes. This is because an edge contraction operation does not decrease the degree of any existing nodes. Since $T_p \in \mathcal{T}_{\geq 3}$, all hidden nodes in $\mathrm{EC}(T_p, V^r)$ have degrees equal to or greater than 3, and since we are including all neighbors of H^r in the subtree S^r, we have $S^r \in \mathcal{T}_{\geq 3}$. By Theorem 5, RG reconstructs all latent trees in $\mathcal{T}_{\geq 3}$ and hence, $S^r = \mathrm{RG}(A^r, \mathbf{d})$.

The computational complexity follows from the corresponding result in recursive grouping. The Chow-Liu tree can be constructed with $O(m^2 \log m)$ complexity. The recursive grouping procedure has complexity $\max_r |A^r|^3$ and $\max_r |A^r| \leq \Delta(\mathrm{MST}(V; \widehat{d}))$. $\square$

A.6 Proof of Theorem 11: Consistency and Sample Complexity of Relaxed RG

(i) Structural consistency follows from Theorem 5 and the fact that the ML estimates of information distances $\widehat{d_{ij}}$ approach d_{ij} (in probability) for all $i, j \in V$ as the number of samples tends to infinity.

Risk consistency for Gaussian and symmetric discrete distributions follows from structural consistency. If the structure is correctly recovered, we can use the equations in (13) and (14) to infer the information distances. Since the distances are in one-to-one correspondence to the correlation coefficients and the crossover probability for Gaussian and symmetric discrete distribution respectively, the parameters are also consistent. This implies that the KL-divergence between p and $\widehat{p}^n$ tends to zero (in probability) as the number of samples n tends to infinity. This completes the proof.

(ii) The theorem follows by using the assumption that the effective depth $\delta = \delta(T_p; V)$ is constant. Recall that $\tau > 0$ is the threshold used in relaxed RG (see (21) in Section 6.3). Let the set of triples (i, j, k) whose pairwise information distances are less than τ apart be $\mathcal{J}$, that is, $(i, j, k) \in \mathcal{J}$ if and only if $\max\{d_{ij}, d_{jk}, d_{ki}\} < \tau$. Since we assume that the true information distances are uniformly bounded, there exist $\tau > 0$ and some sufficiently small $\lambda > 0$ so that if $|\widehat{\Phi}_{ijk} - \Phi_{ijk}| \leq \lambda$ for all $(i, j, k) \in \mathcal{J}$, then RG recovers the correct latent structure.

Define the error event $\mathcal{E}_{ijk} := \{|\widehat{\Phi}_{ijk} - \Phi_{ijk}| > \lambda\}$. We note that the probability of the event $\mathcal{E}_{ijk}$ decays exponentially fast, that is, there exists $J_{ijk} > 0$ such that for all $n \in \mathbb{N}$,

$$\Pr(\mathcal{E}_{ijk}) \leq \exp(-nJ_{ijk}). \tag{34}$$

The proof of (34) follows readily for Chernoff bounds (Hoeffding, 1958) and is omitted. The error probability associated to structure learning can be bounded as follows:

$$\begin{aligned}\Pr\left(h(\widehat{T}^n) \neq T_p\right) &\overset{(a)}{\leq} \Pr\left(\bigcup_{(i,j,k)\in\mathcal{J}} \mathcal{E}_{ijk}\right) \overset{(b)}{\leq} \sum_{(i,j,k)\in\mathcal{J}} \Pr(\mathcal{E}_{ijk}) \\ &\leq m^3 \max_{(i,j,k)\in\mathcal{J}} \Pr(\mathcal{E}_{ijk}) \overset{(c)}{\leq} \exp(3\log m)\exp\left[-n \min_{(i,j,k)\in\mathcal{J}} J_{ijk}\right],\end{aligned}$$

where (a) follows from the fact that if the event $\{h(\widehat{T}^n) \neq T_p\}$ occurs, then there is at least one sibling or parent-child relationship that is incorrect, which corresponds to the union of the events $\mathcal{E}_{ijk}$, that is, there exists a triple $(i,j,k) \in \mathcal{J}$ is such that $\widehat{\Phi}_{ijk}$ differs from Φ_{ijk} by more than λ. Inequality (b) follows from the union bound and (c) follows from (34).

Because the information distances are uniformly bounded, there also exists a constant $J_{\min} > 0$ (independent of m) such that $\min_{(i,j,k)\in\mathcal{J}} J_{ijk} \geq J_{\min}$ for all $m \in \mathbb{N}$. Hence for every $\eta > 0$, if the number of samples satisfies $n > 3(\log(m/\sqrt[3]{\eta}))/J_{\min}$, the error probability is bounded above by η. Let $C := 3/J_{\min}$ to complete the proof of the sample complexity result in (26). The proof for the logarithmic sample complexity of distribution reconstruction for Gaussian and symmetric discrete models follows from the logarithmic sample complexity result for structure learning and the fact that the information distances are in a one-to-one correspondence with the correlation coefficients (for Gaussian models) or crossover probabilities (for symmetric discrete models).

A.7 Proof of Theorem 12: Consistency and Sample Complexity of Relaxed CLRG

(i) Structural consistency of CLGrouping follows from structural consistency of RG (or NJ) and the consistency of the Chow-Liu algorithm. Risk consistency of CLGrouping for Gaussian or symmetric distributions follows from the structural consistency, and the proof is similar to the proof of Theorem 11(i).

(ii) The input to the CLGrouping procedure $\widehat{T}_{\mathrm{CL}}$ is the Chow-Liu tree and has $O(\log m)$ sample complexity (Tan et al., 2010, 2011), where m is the size of the tree. This is true for both discrete and Gaussian data. From Theorem 11, the recursive grouping procedure has $O(\log m)$ sample complexity (for appropriately chosen thresholds) when the input information distances are uniformly bounded. In any iteration of the CLGrouping, the information distances satisfy $d_{ij} \leq \gamma u$, where γ, defined in (30), is the worst-case graph distance of any hidden node from its surrogate. Since γ satisfies (32), $d_{ij} \leq u^2\delta/l$. If the effective depth $\delta = O(1)$ (as assumed), the distances $d_{ij} = O(1)$ and the sample complexity is $O(\log m)$. □

References

K. Atteson. The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, 25(2):251–278, 1999.

M. F. Balcan and P. Gupta. Robust hierarchical clustering. In *Intl. Conf. on Learning Theory (COLT)*, 2010.

H.-J. Bandelth and A. Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Adv. Appl. Math*, 7:309–43, 1986.

S. Bhamidi, R. Rajagopal, and S. Roch. Network delay inference from additive metrics. *To appear in Random Structures and Algorithms, Arxiv preprint math/0604367*, 2009.

R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network tomography: Recent developments. *Stat. Science*, 19:499–517, 2004.

J. T. Chang and J. A. Hartigan. Reconstruction of evolutionary trees from pairwise distributions on current species. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pages 254–257, 1991.

T. Chen, N. L. Zhang, and Y. Wang. Efficient model evaluation in the search based approach to latent structure discovery. In *4th European Workshop on Probabilistic Graphical Models*, 2008.

M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky. Exploiting hierarchical context on a large database of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, June 2010.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, 3:462–467, 1968.

T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Science/Engineering/Math, 2nd edition, 2003.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag, New York, 1999.

M. Csűrös. *Reconstructing Phylogenies in Markov Models of Sequence Evolution*. PhD thesis, Yale University, 2000.

C. Daskalakis, E. Mossel, and S. Roch. Optimal phylogenetic reconstruction. In *STOC '06: Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 159–168, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge Univ. Press, 1999.

G. Elidan and N. Friedman. Learning hidden variable networks: The information bottleneck approach. *Journal of Machine Learning Research*, 6:81–127, 2005.

P. L. Erdős, L. A. Székely, M. A. Steel, and T. J. Warnow. A few logs suffice to build (almost) all trees: Part ii. *Theoretical Computer Science*, 221:153–184, 1999.

J. Farris. Estimating phylogenetic trees from distance matrices. *Amer. Natur.*, 106(951):645–67, 1972.

S. Harmeling and C. K. I. Williams. Greedy learning of binary latent trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1958.

D. Hsu, S.M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. In *Intl. Conf. on Learning Theory (COLT)*, 2009.

A. K. Jain, M. N. Murthy, and P. J. Flynn. Data clustering: A review. *ACM Computing Reviews*, 1999.

T. Jiang, P. E. Kearney, and M. Li. A polynomial-time approximation scheme for inferring evolutionary trees from quartet topologies and its application. *SIAM J. Comput.*, 30(6):194261, 2001.

C. Kemp and J. B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Science*, 105(31):10687–10692, 2008.

J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1), Feb 1956.

M. R. Lacey and J. T. Chang. A signal-to-noise analysis of phylogeny estimation by neighbor-joining: Insufficiency of polynomial length sequences. *Mathematical Biosciences*, 199:188–215, 2006.

J. A. Lake. Reconstructing evolutionary trees from dna and protein sequences: Parallnear distances. *Proceedings of the National Academy of Science*, 91:1455–1459, 1994.

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

P. F. Lazarsfeld and N.W. Henry. *Latent Structure Analysis*. Boston: Houghton Mifflin, 1968.

D. G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications*. Wiley, 1979.

D. Parikh and T. H. Chen. Hierarchical Semantics of Objects (hSOs). In *ICCV*, pages 1–8, 2007.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Network of Plausible inference*. Morgan Kaufmann, 1988.

R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36, 1957.

D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53: 131–147, 1981.

S. Roch. A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(1), 2006.

P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics*, 20:53–65, 1987.

N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4(4):406–425, Jul 1987.

S. Sattath and A. Tversky. Additive similarity trees. *Psychometrika*, 42:319–45, 1977.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

R. J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley-Interscience, Nov 1980.

S. Shen. Large deviation for the empirical correlation coefficient of two Gaussian random variables. *Acta Mathematica Scientia*, 27(4):821–828, Oct 2007.

R. Silva, R. Scheine, C. Glymour, and P. Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7:191–246, Feb 2006.

L. Song, B. Boots, S. M. Siddiqi, G. Gordon, and A. Smola. Hilbert space embeddings of hidden Markov models. In *Proc. of Intl. Conf. on Machine Learning*, 2010.

K. St. John, T. Warnow, B. M. E. Moret, and L. Vawter. Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. *J. Algorithms*, 48(1):173–193, 2003.

M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.

V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning Gaussian tree models: Analysis of error exponents and extremal structures. *IEEE Transactions on Signal Processing*, 58(5):2701–2714, May 2010.

V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning high-dimensional Markov forest distributions: Analysis of error rates. *Journal of Machine Learning Research (In Press)*, 2011.

Y. Tsang, M. Coates, and R. D. Nowak. Network delay tomography. *IEEE Trans. Signal Processing*, 51:21252136, 2003.

Y. Wang, N. L. Zhang, and T. Chen. Latent tree models and approximate inference in Bayesian networks. *Journal of Artificial Intelligence Research*, 32:879–900, Aug 2008.

N. L. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.

N. L. Zhang and T Kočka. Efficient learning of hierarchical latent class models. In *ICTAI*, 2004.

Journal of Machine Learning Research 12 (2011) 1813-1833 Submitted 12/09; Revised 9/10; Published 6/11

Hyper-Sparse Optimal Aggregation

Stéphane Gaïffas STEPHANE.GAIFFAS@UPMC.FR
Laboratoire de Statistique Théorique et Appliquée
Université Pierre et Marie Curie - Paris 6
75005, Paris, FRANCE

Guillaume Lecué GUILLAUME.LECUE@UNIV-MLV.FR
CNRS, Laboratoire d'Analyse et Mathématiques appliquées
Université Paris-Est - Marne-la-vallée
77454, Marne-la-Valle Cedex 2, FRANCE

Editor: John Shawe-Taylor

Abstract

Given a finite set F of functions and a learning sample, the aim of an aggregation procedure is to have a risk as close as possible to risk of the best function in F. Up to now, optimal aggregation procedures are convex combinations of every elements of F. In this paper, we prove that optimal aggregation procedures combining only two functions in F exist. Such algorithms are of particular interest when F contains many irrelevant functions that should not appear in the aggregation procedure. Since selectors are suboptimal aggregation procedures, this proves that two is the minimal number of elements of F required for the construction of an optimal aggregation procedure in every situations. Then, we perform a numerical study for the problem of selection of the regularization parameters of the Lasso and the Elastic-net estimators. We compare on simulated examples our aggregation algorithms to aggregation with exponential weights, to Mallow's C_p and to cross-validation selection procedures.

Keywords: aggregation, exact oracle inequality, empirical risk minimization, empirical process theory, sparsity, Lasso, Lars

1. Introduction

Let (Ω,μ) be a probability space and ν be a probability measure on $\Omega\times\mathbb{R}$ such that μ is its marginal on Ω. Assume (X,Y) and $D_n := (X_i,Y_i)_{i=1}^n$ to be $n+1$ independent random variables distributed according to ν, and that we are given a finite set $F=\{f_1,\dots,f_M\}$ of real-valued functions on Ω, usually called a *dictionary*, or a set of *weak learners*. This set of functions is often a set of estimators computed on a *training* sample, which is independent of the sample D_n (*learning* sample).

We consider the problem of prediction of Y from X using the functions given in F and the sample D_n. If $f:\Omega\to\mathbb{R}$, we measure its error of prediction, or risk, by the expectation of the squared loss

$$R(f)=\mathbb{E}(f(X)-Y)^2.$$

If $\widehat{f}$ depends on D_n, its risk is the conditional expectation

$$R(\widehat{f})=\mathbb{E}[(\widehat{f}(X)-Y)^2|D_n].$$

The aim of the problem of aggregation is to construct a procedure $\tilde{f}$ (called an *aggregate*) using D_n and F with a risk which is very close to the smallest risk over F. Namely, one wants to prove that $\tilde{f}$

satisfies an inequality of the form

$$R(\tilde{f}) \leq \min_{f \in F} R(f) + r(F,n) \tag{1}$$

with a large probability or in expectation. Inequalities of the form (1) are called *exact oracle inequalities* and $r(F,n)$ is called the *residue*. A classical result (Juditsky et al., 2008) says that aggregates with values in F cannot satisfy an inequality like (1) with a residue smaller than $((\log M)/n)^{1/2}$ for every F. Nevertheless, it is possible to mimic the oracle (an *oracle* is a element in F achieving the minimal risk over F) up to the residue $(\log M)/n$ (see Juditsky et al., 2008 and Lecué and Mendelson, 2009, among others) using an aggregate $\tilde{f}$ that combines all the elements of F. In this case, we say that $\tilde{f}$ is an *optimal aggregation procedure*. This notion of optimality is given in Tsybakov (2003) and Lecué and Mendelson (2009), and it is the one we will refer to in this paper.

Given the set of functions F, a natural way to predict Y is to compute the empirical risk minimization procedure (ERM), the one that minimizes the empirical risk

$$R_n(f) := \frac{1}{n} \sum_{i=1}^{n} (Y_i - f(X_i))^2$$

over F. This very basic principle is at the core of aggregation procedures (for regression with squared loss). An aggregate is typically represented as a convex combination of the elements of F. Namely,

$$\widehat{f} := \sum_{j=1}^{M} \theta_j(D_n, F) f_j,$$

where $(\theta_j(D_n,F))_{j=1}^{M}$ is a vector of non-negative coordinates suming to 1. Up to now, most of the optimal aggregation procedures are based on exponential weights: aggregation with cumulated exponential weights (ACEW), see Catoni (2001), Yang (2004), Yang (2000), Juditsky et al. (2008), Juditsky et al. (2005), Audibert (2009) and aggregation with exponential weights (AEW), see Leung and Barron (2006) and Dalalyan and Tsybakov (2007), among others. The weights of the ACEW are given by

$$\theta_j^{(\mathrm{ACEW})} := \frac{1}{n} \sum_{k=1}^{n} \frac{\exp(-R_k(f_j)/T)}{\sum_{l=1}^{M} \exp(-R_k(f_l)/T)},$$

where T is the so-called *temperature* parameter. The weights of the AEW are given by

$$\theta_j^{(\mathrm{AEW})} := \frac{\exp(-R_n(f_j)/T)}{\sum_{l=1}^{M} \exp(-R_n(f_j)/T)}.$$

The ACEW satisfies (1) for $r(F,n) \sim (\log M)/n$, see references above, so it is optimal in the sense of Tsybakov (2003). The AEW has been proved to be optimal in the regression model with deterministic design for large temperatures in Dalalyan and Tsybakov (2007). Altough, for small temperatures, AEW can be suboptimal both in expectation and with large probability (cf. Lecué and Mendelson, 2010).

In these aggregates, no coefficient θ_j is equal to zero, although they can be very small, depending on the value of $R_n(f_j)$ and T (this makes in particular the choice of T of importance). So, even the worse elements of F have an influence on the aggregate. This can be a problem when one wants to use aggregation to construct adaptive procedures. Indeed, one could imagine large dictionaries containing many different types of estimators (kernel estimators, projection estimators, etc.) with many different parameters (smoothing parameters, groups of variables, etc.). Some of the estimators are likely to be more adapted than the others, depending on the kind of models that fits well the data, and, there may be only few of them among a large dictionary. An aggregate that combines only the most adapted estimators from the dictionary and that removes the irrelevant ones is suitable in this case. The challenge is then to find such a procedure which is still an optimal aggregate. An improvement going in this direction has been made using a preselection step in Lecué and Mendelson (2009). This preselection step allows to remove all the estimators in F which performs badly on a learning subsample. In this paper, we want to go a step further: we look for an aggregation algorithm that shares the same property of optimality, but with as few non-zero coefficients θ_j as possible, hence the name *hyper-sparse aggregate*. This leads to the following question:

Question 1 *What is the minimal number of non-zero coefficients θ_j such that an aggregation procedure $\tilde{f} = \sum_{j=1}^{M} \theta_j f_j$ is optimal?*

It turns out that the answer to Question 1 is two. Indeed, if every coefficient is zero, excepted for one, the aggregate coincides with an element of F, and as we mentioned before, such a procedure can only achieve the rate $((\log M)/n)^{1/2}$ (unless extra properties are satisfied by F and ν). In Definition 1 below (see Section 2) we construct three procedures, where two of them (see (6) and (7)) only have two non-zero coefficients θ_j. We prove in Theorem 2 below that these procedures are optimal, since they achieve the rate $(\log M)/n$.

2. Definition of the Aggregates and Results

First, we need to introduce some notations and assumptions. Let us recall that the ψ_1-norm of a random variable Z is given by $\|Z\|_{\psi_1} := \inf\{c > 0 : \mathbb{E}[\exp(|Z|/c))] \leq 2\}$. We say that Z is sub-exponential when $\|Z\|_{\psi_1} < +\infty$. We work under the following assumptions.

Assumption 1 *We can write*

$$Y = f_0(X) + \varepsilon,$$

where ε is such that $\mathbb{E}(\varepsilon|X) = 0$ and $\mathbb{E}(\varepsilon^2|X) \leq \sigma_\varepsilon^2$ a.s. for some constant $\sigma_\varepsilon > 0$. Moreover, we assume that one of the following points holds.

- *(Bounded setup) There is a constant $b > 0$ such that:*

$$\max\left(\|Y\|_\infty, \sup_{f\in F} \|f(X)\|_{L_\infty}\right) \leq b. \tag{2}$$

- *(Sub-exponential setup) There is a constant $b > 0$ such that:*

$$\max\left(\|\varepsilon\|_{\psi_1}, \sup_{f\in F} \|f(X) - f_0(X)\|_{L_\infty}\right) \leq b. \tag{3}$$

Note that Assumption (3) allows for an unbounded output Y. The results given below differ a bit depending on the considered assumption (there is an extra $\log n$ term in the sub-exponential case). To simplify the notations, we assume from now on that we have $2n$ observations from a sample $D_{2n} = (X_i, Y_i)_{i=1}^{2n}$. Let us define our aggregation procedures.

Definition 1 (Aggregation procedures) *Follow the following steps:*

(0. Initialization) *Choose a confidence level $x > 0$. If* (2) *holds, define*

$$\phi = \phi_{n,M}(x) = b\sqrt{\frac{\log M + x}{n}}.$$

If (3) *holds, define*

$$\phi = \phi_{n,M}(x) = (\sigma_\varepsilon + b)\sqrt{\frac{(\log M + x)\log n}{n}}.$$

(1. Splitting) *Split the sample D_{2n} into $D_{n,1} = (X_i, Y_i)_{i=1}^n$ and $D_{n,2} = (X_i, Y_i)_{i=n+1}^{2n}$.*

(2. Preselection) *Use $D_{n,1}$ to define a random subset of F :*

$$\widehat{F}_1 = \Big\{ f \in F : R_{n,1}(f) \le R_{n,1}(\widehat{f}_{n,1}) + c \max\big(\phi \| \widehat{f}_{n,1} - f\|_{n,1}, \phi^2\big)\Big\}, \tag{4}$$

where $\|f\|_{n,1}^2 = n^{-1}\sum_{i=1}^n f(X_i)^2$, $R_{n,1}(f) = n^{-1}\sum_{i=1}^n (f(X_i) - Y_i)^2$, $\widehat{f}_{n,1} \in \operatorname{argmin}_{f\in F} R_{n,1}(f)$.

(3. Aggregation) *Choose $\widehat{\mathcal{F}}$ as one of the following sets:*

$$\widehat{\mathcal{F}} = \operatorname{conv}(\widehat{F}_1) = \text{ the convex hull of } \widehat{F}_1 \tag{5}$$

$$\widehat{\mathcal{F}} = \operatorname{seg}(\widehat{F}_1) = \text{ the segments between the functions in } \widehat{F}_1 \tag{6}$$

$$\widehat{\mathcal{F}} = \operatorname{star}(\widehat{f}_{n,1}, \widehat{F}_1) = \text{ the segments between } \widehat{f}_{n,1} \text{ with the elements of } \widehat{F}_1, \tag{7}$$

and return the ERM relative to $D_{n,2}$:

$$\tilde{f} \in \operatorname*{argmin}_{g \in \widehat{\mathcal{F}}} R_{n,2}(g),$$

where $R_{n,2}(f) = n^{-1}\sum_{i=n+1}^{2n} (f(X_i) - Y_i)^2$.

These algorithms are illustrated in Figures 1 and 2. In Figure 1 we summarize the aggregation steps in the three cases. In Figure 2 we give a simulated illustration of the preselection step, and we show the value of the weights of the AEW for a comparison. As mentioned above, the Step 3 of the algorithm returns, when $\widehat{\mathcal{F}}$ is given by (6) or (7), an aggregate which is a convex combination of only two functions in F, among the ones remaining after the preselection step. The preselection step was introduced in Lecué and Mendelson (2009), with the use of (5) only for the aggregation step.

From the computational point of view, the procedure (7) is the most appealing: an ERM in $\operatorname{star}(\widehat{f}_{n,1}, \widehat{F})$ can be computed in a fast and explicit way, see Algorithm 1 below. The next Theorem proves that each procedure given in Definition 1 are optimal.

Theorem 2 *Let $x > 0$ be a confidence level, F be a dictionary with cardinality M and $\tilde{f}$ be one of the aggregation procedure given in Definition 1. If* (2) *holds, we have, with ν^{2n}-probability at least $1 - 2e^{-x}$:*

$$R(\tilde{f}) \le \min_{f \in F} R(f) + c_b \frac{(1+x)\log M}{n},$$

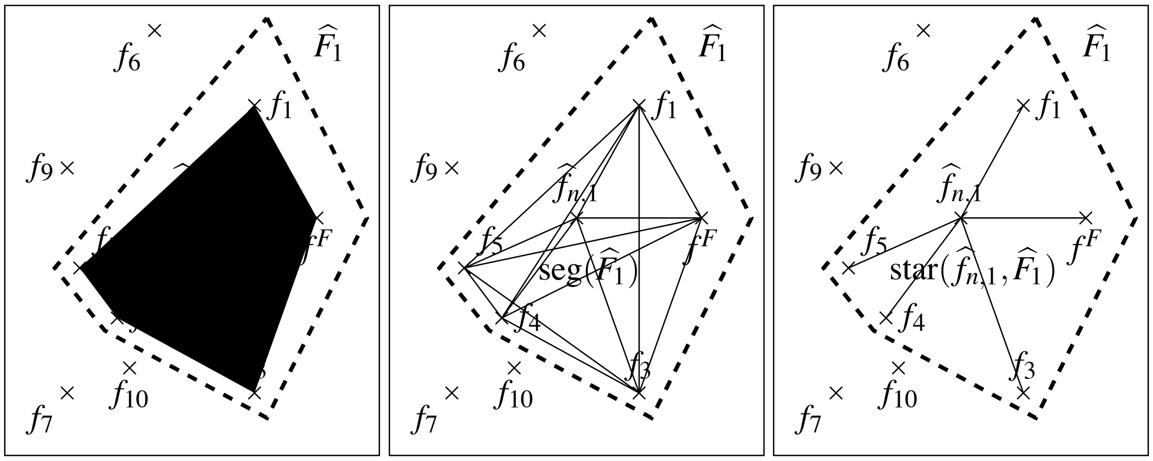


Figure 1: Aggregation algorithms: ERM over $\mathrm{conv}(\widehat{F}_1)$, $\mathrm{seg}(\widehat{F}_1)$, or $\mathrm{star}(\widehat{f}_{n,1}, \widehat{F}_1)$.

where c_b is a constant depending on b.

If (3) *holds, we have, with ν^{2n}-probability at least $1-4e^{-x}$:*

$$R(\tilde{f}) \leq \min_{f \in F} R(f) + c_{\sigma_\varepsilon, b} \frac{(1+x) \log M \log n}{n}.$$

Remark 3 *Note that the definition of the set $\widehat{F}_1$, and thus $\tilde{f}$, depends on the confidence x through the factor $\phi_{n,M}(x)$.*

Remark 4 *To simplify the proofs, we don't give the explicit values of the constants. However, when* (2) *holds, one can choose $c = 4(1+9b)$ in* (4) *and $c = c_1(1+b)$ when* (3) *holds (where c_1 is the absolute constant appearing in Theorem 6). Of course, this is not likely to be the optimal choice.*

Now, we give details for the computation of the star-shaped aggregate, namely the aggregate $\tilde{f}$ given by Definition 1 when $\widehat{\mathcal{F}}$ is (7). Indeed, if $\lambda \in [0,1]$, we have

$$R_{n,2}(\lambda f + (1-\lambda) g) = \lambda R_{n,2}(f) + (1-\lambda) R_{n,2}(g) - \lambda(1-\lambda) \|f-g\|_{n,2}^2,$$

so the minimum of $\lambda \mapsto R_{n,2}(\lambda f + (1-\lambda) g)$ is achieved at

$$\lambda_{n,2}(f,g) = 0 \vee \frac{1}{2}\Big(\frac{R_{n,2}(g) - R_{n,2}(f)}{\|f-g\|_{n,2}^2} + 1\Big) \wedge 1,$$

where $a \vee b = \max(a,b)$, $a \wedge b = \min(a,b)$. So,

$$\min_{\lambda \in [0,1]} R_{n,2}(\lambda f + (1-\lambda) g) = R_{n,2}(\lambda_{n,2}(f,g) f + (1 - \lambda_{n,2}(f,g)) g),$$

which is equal to

$$\begin{array}{lll} R_{n,2}(g) & \text{if} & R_{n,2}(f) - R_{n,2}(g) > \|f-g\|_{n,2}^2, \\ R_{n,2}(f) & \text{if} & R_{n,2}(f) - R_{n,2}(g) < -\|f-g\|_{n,2}^2, \end{array}$$

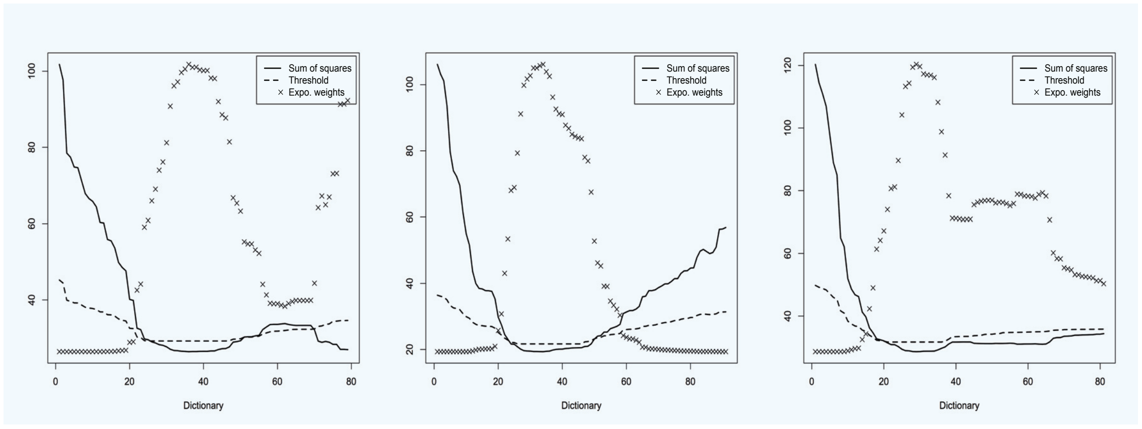


Figure 2: Empirical risk $R_{n,1}(f)$, value of the threshold $R_{n,1}(\widehat{f}_{n,1}) + 2\max(\phi\|\widehat{f}_{n,1} - f\|_{n,1}, \phi^2)$ and weights of the AEW (rescaled) for $f \in F$, where F is a dictionary obtained using LARS, see Section 3 below. Only the elements of F with an empirical risk smaller than the threshold are kept from the dictionary for the construction of the aggregates of Definition (1). The first and third examples correspond to a case where an aggregate with preselection step improves upon AEW, while in the second example, both procedures behaves similarly.

and to

$$\frac{R_{n,2}(f) + R_{n,2}(g)}{2} - \frac{(R_{n,2}(f) - R_{n,2}(g))^2}{4\|f-g\|_{n,2}^2} - \frac{\|f-g\|_{n,2}^2}{4}$$

if $|R_{n,2}(f) - R_{n,2}(g)| \le \|f-g\|_{n,2}^2$. This leads to the next Algorithm 1 for the computation of $\tilde{f}$.

3. Simulation Study

In machine learning, the choice of the tuning parameters in a procedure based on penalization is a main issue. If the procedure is able to perform variable selection (such as the Lasso, see Tibshirani, 1996), then the tuning parameters determines which variables are selected. In many cases, including the Lasso, this choice is commonly done using a Mallow's C_p heuristic (see Efron et al., 2004) or using the V-fold or the leave-one-out cross validations. Since aggregation procedures are known (see references above) to outperform selectors in terms of prediction error, it is tempting to use aggregation for the choice of the tuning parameters. Unfortunately, as we mentioned before, most aggregation procedures provide non-zero weights to many non relevant element in a dictionary: this is a problem for variable selection. Indeed, if we use, for instance, the AEW on a dictionary consisting of the full path of Lasso estimators (provided by the Lars algorithm, see Efron et al., 2004), then the resulting aggregate is likely to select all the variables since the Lasso with a small regularization parameter is very close (and equal if it is zero) to ordinary least-squares (which does not perform any variables selection). So, in this context, the hyper-sparse aggregate of Section 2 is of particular interest. In this section, we compare the prediction error and the accuracy of variable selection of our star-shaped aggregation algorithm to Mallow's C_p heuristic, leave-one-out cross-validation and 10-fold cross-validation. In Section 3.2 we consider a dictionary consisting of the

```
Algorithm 1: Computation of the star-shaped aggregate.
    Input: dictionary $F$, data $(X_i, Y_i)_{i=1}^{2n}$, and a confidence level $x > 0$
    Output: star-shaped aggregate $\tilde{f}$
    Split $D_{2n}$ into two samples $D_{n,1}$ and $D_{n,2}$
    foreach $j \in \{1, \dots, M\}$ do
        Compute $R_{n,1}(f_j)$ and $R_{n,2}(f_j)$, and use this loop to find $\widehat{f}_{n,1} \in \operatorname{argmin}_{f \in F} R_{n,1}(f)$
    end
    foreach $j \in \{1, \dots, M\}$ do
        Compute $\|f_j - \widehat{f}_{n,1}\|_{n,1}$ and $\|f_j - \widehat{f}_{n,1}\|_{n,2}$
    end
    Construct the set of preselected elements
        $\widehat{F}_1 = \Big\{ f \in F : R_{n,1}(f) \le R_{n,1}(\widehat{f}_{n,1}) + c \max\big(\phi \|\widehat{f}_{n,1} - f\|_{n,1}, \phi^2\big) \Big\},$
    where $\phi$ is given in Definition 1.
    foreach $f \in \widehat{F}_1$ do
        compute
            $R_{n,2}(\lambda_{n,2}(\widehat{f}_{n,1}, f)\widehat{f}_{n,1} + (1 - \lambda_{n,2}(\widehat{f}_{n,1}, f))f)$
        and keep the element $f_{\widehat{j}} \in \widehat{F}_1$ that minimizes this quantity
    end
    return
        $\tilde{f} = \lambda_{n,2}(\widehat{f}_{n,1}, f_{\widehat{j}})\widehat{f}_{n,1} + (1 - \lambda_{n,2}(\widehat{f}_{n,1}, f_{\widehat{j}}))f_{\widehat{j}},$
```

entire sequence of Lasso estimators and a dictionary consisting of entire sequences of the elastic-net estimators (see Zou and Hastie, 2005) corresponding to several ridge penalization parameters, so this dictionary contains the Lasso, the elastic-net, the ridge and the ordinary least-squares estimators.

Remark 5 *Note that since an aggregation algorithm is "generic", in the sense that it can be applied to any dictionary, one could consider larger dictionaries, containing many instances of different type of estimators, for several choices of the tuning parameters, like the Adaptive Lasso (see Zou, 2006) among many other instances of the Lasso. We believe that the conclusion of the numerical study proposed here would be the same as for a much larger dictionary. Indeed, let us recall that here, the focus is on the comparison of selection and aggregation procedures for the choice of tuning parameters, and not on the comparison of the procedures inside the dictionary themselves.*

3.1 Examples of Models

We simulate n independent copies of the linear regression model

$$Y = \beta^\top X + \varepsilon,$$

where $\beta \in \mathbb{R}^p$. Several settings are considered, see Models 1-6 below, including sparse and non-sparse vectors β and several signal-to-noise ratios. Models 1-4 are from Tibshirani (1996).

Model 1 (A few effects). We set $\beta = (3, 1.5, 0, 0, 2, 0, 0, 0)$, so $p = 8$, and we let n to be 20 and 60. The vector $X = (X^1, \dots, X^d)$ is a centered normal vector with covariance matrix $\mathrm{Cov}(X^i, X^j) = \rho^{|i-j|}$, with $\rho = 1/2$. The noise ε_i is $N(0, \sigma^2)$ with σ equal to 1 or 3.

Model 2 (Every effects). This example is the same as Model 1, but with $\beta = (2, 2, 2, 2, 2, 2, 2, 2)$.

Model 3 (A single effect). This example is the same as Model 1, but with $\beta = (5, 0, 0, 0, 0, 0, 0, 0)$.

Model 4 (A larger model). We set $\beta = (0^{10}, 2^{10}, 0^{10}, 2^{10})$, where x^y stands for the vector of dimension y with each coordinate equal to x, so $p = 40$. We let n to be 100 and 200. We consider covariates $X_i^j = Z_{i,j} + Z_i$ where $Z_{i,j}$ and Z_i are independent $N(0,1)$ variables. This induces pairwise correlation equal to 0.5 among the covariates. The noise ε_i is $N(0, \sigma^2)$ with σ equal to 15 or 7.

Model 5 (Sparse vector in high dimension). We set $\beta = (2.5^5, 1.5^5, 0.5^5, 0^{185})$, so $p = 200$. We let n to be 50 and 100. The first 15 covariates $(X^1, \dots, X^{15})$ and the remaining 185 covariates $(X^{16}, \dots, X^{200})$ are independent. Each of these are Gaussian vectors with the same covariance matrix as in Model 1 with $\rho = 0.5$. The noise is $N(0, \sigma^2)$ with σ equal to 3 and 1.5.

Model 6 (Sparse vector in high dimension, stronger correlation). This example is the same as Model 5, but with $\rho = 0.95$.

3.2 Procedures

We consider a dictionary consisting of the entire sequence of Lasso estimators and a dictionary with several sequences of elastic-net estimators, corresponding to ridge parameters in the set of values $\{0, 0.01, 0.1, 1, 5, 10, 20, 50, 100\}$ (these dictionaries are computed with the `lars` and `enet` routines from `R`).[1] For each dictionary, we compute the prediction errors $|X(\widehat{\beta} - \beta)|_2$ (where X is the matrix with rows $X_1^\top, \dots, X_n^\top$ and $|\cdot|_2$ is the ℓ_2^n-norm of 200 replications (this makes the results stable enough), where $\widehat{\beta}$ is one of the following:

- $\widehat{\beta}^{(\mathrm{Oracle})}$ = the element of the dictionary with smallest prediction error
- $\widehat{\beta}^{(C_p)}$ = the Lasso estimator selected by Mallows-C_p heuristic
- $\widehat{\beta}^{(10-\mathrm{Fold})}$ = the element of the dictionary selected by 10-fold cross-validation
- $\widehat{\beta}^{(\mathrm{Loo})}$ = the element of the dictionary selected by leave-one-out cross-validation
- $\widehat{\beta}^{(\mathrm{AEW})}$ = The aggregate with exponential weights applied to the dictionary, with temperature parameter equal to $4\sigma^2$, see for instance Dalalyan and Tsybakov (2007)
- $\widehat{\beta}^{(\mathrm{Star})}$ = the star-shaped aggregate applied to the dictionary.

For the AEW and the star-shaped aggregate, the splits are chosen at random with size $[n/2]$ for training and $n - [n/2]$ for learning. For both aggregates we use jackknife: we compute the mean of 100 aggregates obtained with several splits chosen at random. This makes the final aggregates less

1. R can be found at `www.r-project.org`.

dependent on the split. As a matter of fact, we observed in our numerical studies that Star-shaped aggregation with the preselection step and without it (see Definition 1) provides close estimators. So, in order to improve the computational burden, the numerical results of the Star-shaped aggregate reproduced here are the ones obtained without the preselection step.

We need to explain how variable selection is performed based on J star-shaped aggregates coming from J random splits (here we take $J = 100$). A Star-shaped aggregate $\widehat{f}^{(j)}$, corresponding to a split j, can be written as

$$\widehat{f}^{(j)} = \widehat{\lambda}^{(j)} \widehat{f}^{(j)}_{\text{ERM}} + (1 - \widehat{\lambda}^{(j)}) \widehat{f}^{(j)}_{\text{other}},$$

where $\widehat{f}^{(j)}_{\text{ERM}}$ is the ERM in F corresponding to the split j and $\widehat{f}^{(j)}_{\text{other}}$ is the other vertex of the segment where the empirical risk is minimized (recall that the aggregate minimizes the empirical risk over the set of segments $\text{star}(\widehat{f}^{(j)}_{\text{ERM}}, F)$). For each split j, we estimate the significance of each covariate using

$$\widehat{\pi}^{(j)} = \widehat{\lambda}^{(j)} \mathbf{1}_{\widehat{\beta}^{(j)}_{\text{ERM}} \neq 0} + (1 - \widehat{\lambda}^{(j)}) \mathbf{1}_{\widehat{\beta}^{(j)}_{\text{other}} \neq 0},$$

where $\mathbf{1}_{v \neq 0} = (\mathbf{1}_{v_1 \neq 0}, \ldots, \mathbf{1}_{v_d \neq 0})$. The vector $\widehat{\pi}^{(j)}$ does a simple average of the contributions of the supports of $\widehat{\beta}^{(j)}_{\text{ERM}}$ and $\widehat{\beta}^{(j)}_{\text{other}}$, weighted by $\widehat{\lambda}^{(j)}$. To take into consideration each split, we simply compute the mean of the significances of each split:

$$\widehat{\pi} = \frac{1}{J} \sum_{j=1}^{J} \widehat{\pi}^{(j)}.$$

The vector $\widehat{\pi}$ contains the final significances of each covariate. This procedure is close in spirit to the stability selection procedure described in Meinshausen and Bühlmann (2010), since each aggregate is related to a subsample. Finally, the selected covariates are the one in

$$\widehat{S} = \Big\{ k \in \{1, \ldots, p\} : \widehat{\pi}_k \geq \widehat{t} \Big\},$$

where $\widehat{t}$ is a random threshold given by

$$\widehat{t} = \frac{1}{2} \Big(1 + \frac{\widehat{q}^2}{p^2 \beta} \Big),$$

where $\widehat{q} = \min(\widehat{s}, \sqrt{0.7p})$, $\beta = p/10$ and $\widehat{s} = \frac{1}{J} \sum_{j=1}^{J} \sum_{k=1}^{p} \widehat{\pi}^{(j)}_k$ is the average sparsity (number of non-zero coefficients) for each splits. This choice of threshold follows the arguments from Meinshausen and Bühlmann (2010), together with some empirical tuning.

For each of the Models 1-6, the boxplots of the 200 prediction errors are given in Figures 3 and 4. Note that in a high dimensional setting ($p > n$), we don't reproduce the C_p's prediction errors, since in this case the `lars` package does not give it correctly. For the elastic-net dictionary, the boxplot of the predictions errors are given for Models 1-4 in Figure 5. The results concerning variables selection for the Lars and the Elastic-Net dictionaries are given in Tables 1 and 2. In these tables we reproduce the number of selected variables by each procedure, and the number of noise variables (the selected variables which are not active in the true model).

3.3 Conclusion

In most cases, the Star-Shaped aggregate improves upon the AEW and the considered selection procedures both in terms of prediction error and variable selection. The proposed variable selection algorithm based on star-shaped aggregation and stability selection tends to select smaller models than the C_p and cross-validation methods (see Table 1, Models 1-4) leading to less noise variables. In particular, in high-dimensional cases ($p > n$), it is much more stable regarding the sample size and noise level, and provides better results most of the time (see Table 1, Models 5-6). In terms of prediction error, the Star-Shaped always improve the AEW, and is better than the C_p and cross-validations in most cases. We can say that, roughly, the C_p and the cross-validations are better than the Star-Shaped aggregate only for non-sparse vectors (since these selection procedures tend to select larger models), in particular when n is small and σ is large. We can conclude by saying that, in the worst cases, the Star-shaped algorithm has prediction and selection performances which are comparable to cross-validations and C_p heuristic, but, on the other hand, it can improve them a lot (in particular for sparse vectors). One can think of the Star-Shaped aggregation algorithm as an alternative to cross-validation and C_p.

Acknowledgments

This work is supported by French Agence Nationale de la Recherce (ANR) ANR Grant "PROGNOSTIC" ANR-09-JCJC-0101-01.

Appendix A. Proofs

We will use the following notations. If $f^F \in \operatorname{argmin}_{f\in F} R(f)$, we will consider the excess loss

$$\mathcal{L}_f = \mathcal{L}_F(f)(X,Y) := (Y - f(X))^2 - (Y - f^F(X))^2,$$

and use the notations

$$P\mathcal{L}_f := \mathbb{E}\mathcal{L}_f(X,Y), \quad P_n\mathcal{L}_f := \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_f(X_i,Y_i).$$

A.1 Proof of Theorem 2

Let us prove the result in the ψ_1 case, the other case is similar. Fix $x > 0$ and let $\widehat{\mathcal{F}}$ be either (5), (6) or (7). Set $d := \operatorname{diam}(\widehat{F}_1, L_2(\mu))$. Consider the second half of the sample $D_{n,2} = (X_i,Y_i)_{i=n+1}^{2n}$. By Corollary 8 (see Appendix A.2 below), with probability at least $1-4\exp(-x)$ (relative to $D_{n,2}$), we have for every $f \in \widehat{\mathcal{F}}$

$$\Big|\frac{1}{n}\sum_{i=1+n}^{2n} \mathcal{L}_{\widehat{\mathcal{F}}}(f)(X_i,Y_i) - \mathbb{E}\big(\mathcal{L}_{\widehat{\mathcal{F}}}(f)(X,Y)|D_{n,1}\big)\Big| \le c(\sigma_\varepsilon + b)\max(d\phi, b\phi^2),$$

where $\mathcal{L}_{\widehat{\mathcal{F}}}(f)(X,Y) := (f(X)-Y)^2 - (f^{\widehat{\mathcal{F}}}(X)-Y)^2$ is the excess loss function relative to $\widehat{\mathcal{F}}$, $f^{\widehat{\mathcal{F}}} \in \operatorname{argmin}_{f\in\widehat{\mathcal{F}}} R(f)$ and where $\phi = \sqrt{((\log M + x)\log n)/n}$. By definition of $\tilde{f}$, we have

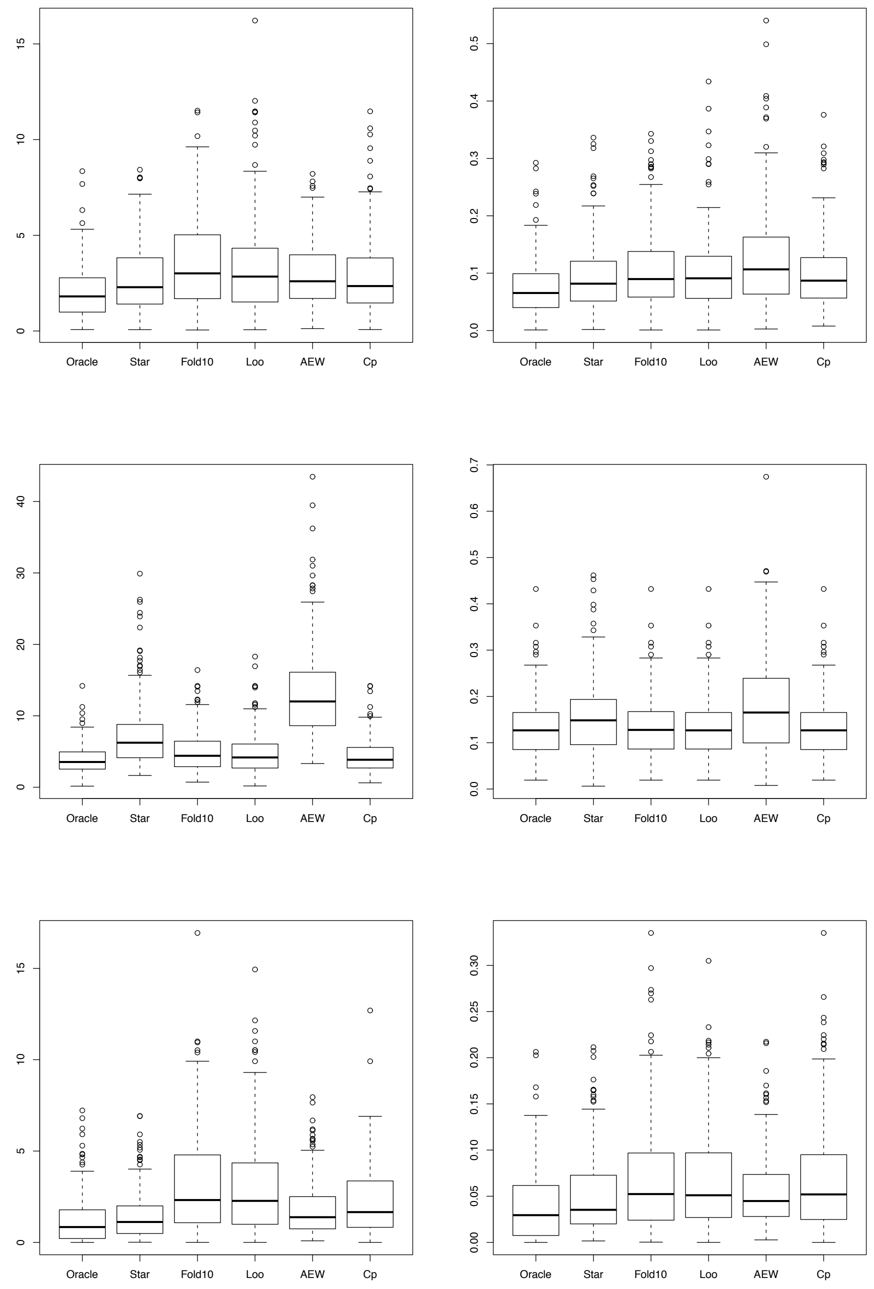


Figure 3: First line: prediction errors for Model 1, with $n = 20$, $\sigma = 3$ (left) and $n = 60$, $\sigma = 1$ (right) ; Second line : prediction errors for Model 2, with $n = 20, \sigma = 3$ (left) and $n = 60$, $\sigma = 1$ (right) ; thrid line: prediction errors for Model 3, with $n = 20$, $\sigma = 3$ (left) and $n = 60, \sigma = 1$ (right)

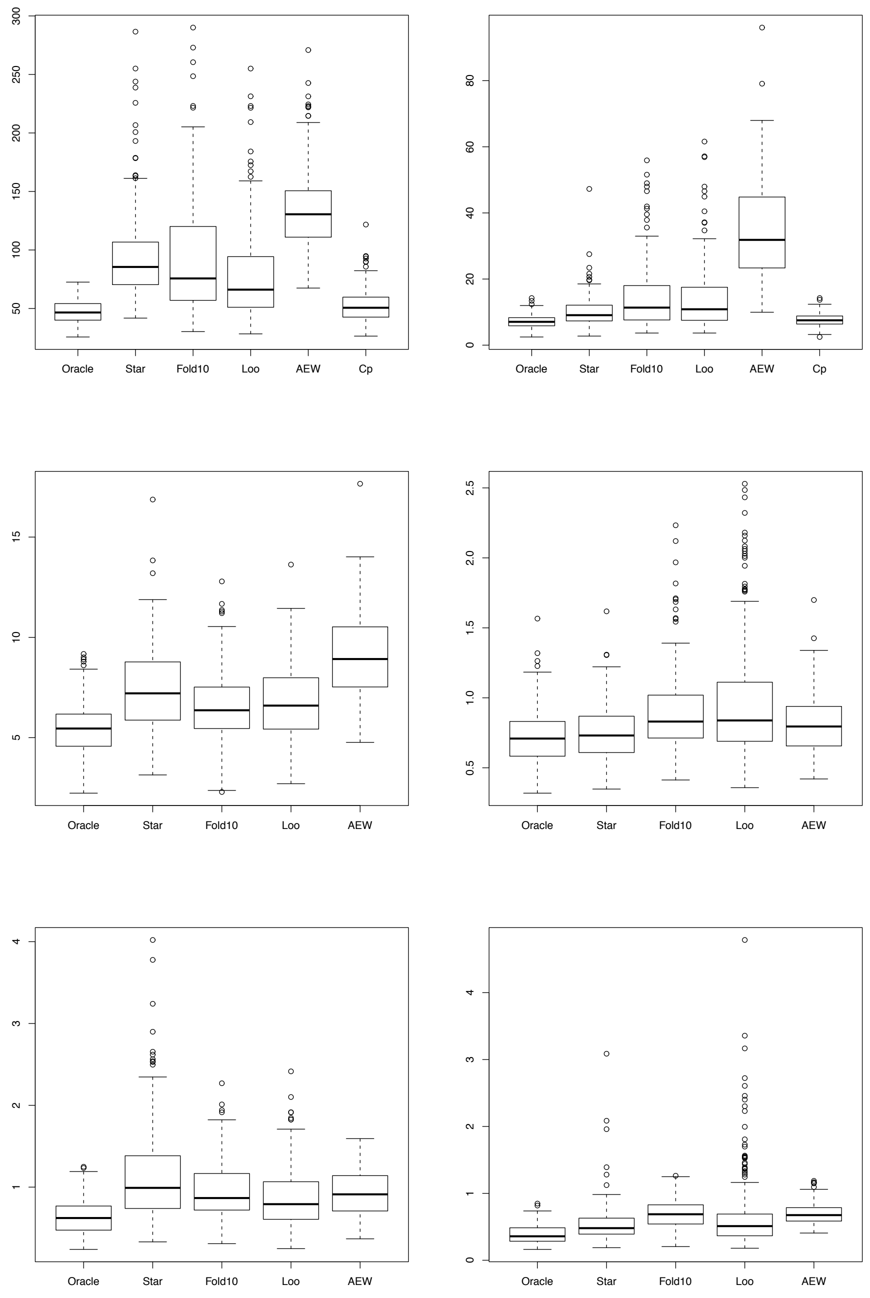


Figure 4: First line: prediction errors for Model 4, with $n = 100, \sigma = 15$ (left) and $n = 200, \sigma = 7$ (right) ; Second line: Prediction errors for Model 5, with $n = 50, \sigma = 3$ (left) and $n = 100$, $\sigma = 1.5$ (right) ; Third line: Prediction errors for Model 6, with $n = 50, \sigma = 1.5$ (left) and $n = 100, \sigma = 1.5$ (right)

	Model 1				**Model 2**			
	$n = 20, \sigma = 3$		$n = 60, \sigma = 1$		$n = 20, \sigma = 3$		$n = 60, \sigma = 1$	
	Selected	Noise	Selected	Noise	Selected	Noise	Selected	Noise
Truth	3	0	3	0	8	0	8	0
10-fold	3.870	1.410	5.260	2.260	7.190	0	8	0
Loo	3.965	1.465	5.055	2.055	7.235	0	8	0
Cp	4.165	1.645	4.710	1.710	7.085	0	8	0
Star	2.860	0.675	4.355	1.355	6.250	0	8	0
	Model 3				**Model 4**			
	$n = 20, \sigma = 3$		$n = 60, \sigma = 1$		$n = 100, \sigma = 15$		$n = 200, \sigma = 7$	
	Selected	Noise	Selected	Noise	Selected	Noise	Selected	Noise
Truth	1	0	1	0	20	0	20	0
10-fold	2.365	1.365	2.980	1.980	21.610	6.955	28.405	8.415
Loo	2.440	1.440	2.645	1.645	22.295	7.305	28.480	8.495
Cp	2.965	1.965	2.650	1.650	23.860	8.175	29.715	9.720
Star	1.655	0.655	1.855	0.855	18.065	4.910	27.850	7.855
	Model 5				**Model 6**			
	$n = 100, \sigma = 1.5$		$n = 200, \sigma = 0.5$		$n = 100, \sigma = 1.5$		$n = 200, \sigma = 0.5$	
	Selected	Noise	Selected	Noise	Selected	Noise	Selected	Noise
Truth	15	0	15	0	15	0	15	0
10-fold	47.375	32.550	14.035	0	39.150	25.830	7.560	0
Loo	44.030	29.215	10.455	0	24.370	10.990	2.425	0
Star	15.690	1.245	17.780	2.780	13.175	0.055	15.145	0.150

Table 1: Accuracy of variable prediction in Models 1 to 6 (Lars dictionary)

$\frac{1}{n}\sum_{i=n+1}^{2n}\mathcal{L}_{\widehat{\mathcal{F}}}(\tilde{f})(X_i,Y_i) \leq 0$, so, on this event (relative to $D_{n,2}$)

$$\begin{aligned}
R(\tilde{f}) &\leq R(f^{\widehat{\mathcal{F}}}) + \mathbb{E}\big(\mathcal{L}_{\widehat{\mathcal{F}}}(\tilde{f})|D_{n,1}\big) - \frac{1}{n}\sum_{i=n+1}^{2n}\mathcal{L}_{\widehat{\mathcal{F}}}(\tilde{f})(X_i,Y_i) \\
&\leq R(f^{\widehat{\mathcal{F}}}) + c(\sigma_\varepsilon + b)\max(d\phi, b\phi^2) \\
&= R(f^F) + \Big(c(\sigma_\varepsilon + b)\max(d\phi, b\phi^2) - \big(R(f^F) - R(f^{\widehat{\mathcal{F}}})\big)\Big) \\
&=: R(f^F) + \beta,
\end{aligned}$$

and it remains to show that

$$\beta \leq c_{b,\sigma_\varepsilon}\frac{(1+x)\log M \log n}{n}.$$

When $\widehat{\mathcal{F}}$ is given by (5) or (6), the geometrical configuration is the same as in Lecué and Mendelson (2009), so we skip the proof.

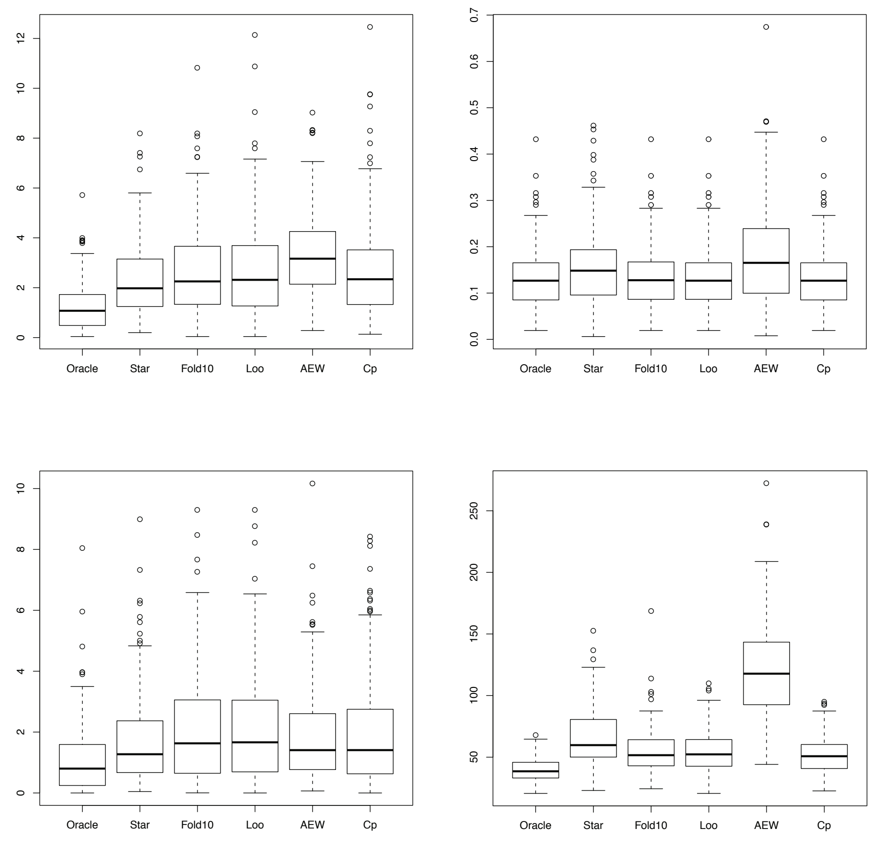


Figure 5: Prediction errors for Models 1 to 4 using the elastic-net dictionary (upper left: Model 1 with $\sigma = 3, n = 20$, upper right: Model 2 with $\sigma = 3, n = 20$, bottom left: Model 3 with $\sigma = 3, n = 20$ and bottom right: Model 4 with $n = 100, \sigma = 15$).

	Model 1		**Model 2**		**Model 3**		**Model 4**	
	$n = 20, \sigma = 3$		$n = 20, \sigma = 3$		$n = 20, \sigma = 3$		$n = 100, \sigma = 15$	
	Selected	Noise	Selected	Noise	Selected	Noise	Selected	Noise
Truth	3	0	8	0	1	0	20	0
10-fold	5.040	2.155	7.450	0	3.045	2.045	25.575	9.475
Loo	4.940	2.065	7.460	0	2.980	1.980	25.535	9.660
Cp	4.490	1.660	7.335	0	2.760	1.760	24.345	8.470
Star	4.355	1.475	7.485	0	2.080	1.080	24.090	8.755

Table 2: Accuracy of variable prediction in Models 1 to 4 (Elastic-Net dictionary)

Let us turn out to the situation where $\widehat{\mathcal{F}}$ is given by (7). Recall that $\widehat{f}_{n,1}$ is the ERM on $\widehat{F}_1$ using $D_{n,1}$. Consider f_1 such that $\|\widehat{f}_{n,1} - f_1\|_{L^2(\mu)} = \max_{f \in \widehat{F}_1} \|\widehat{f}_{n,1} - f\|_{L^2(\mu)}$, and note that

$$\|\widehat{f}_{n,1} - f_1\|_{L^2(\mu)} \leq d \leq 2\|\widehat{f}_{n,1} - f_1\|_{L^2(\mu)}.$$

The mid-point $f_2 := (\widehat{f}_{n,1} + f_1)/2$ belongs to $\mathrm{star}(\widehat{f}_{n,1}, \widehat{F}_1)$. Using the parallelogram identity, we have for any $u, v \in L_2(\nu)$:

$$\mathbb{E}_\nu\Big(\frac{u+v}{2}\Big)^2 \leq \frac{\mathbb{E}_\nu(u^2) + \mathbb{E}_\nu(v^2)}{2} - \frac{\|u-v\|_{L_2(\nu)}^2}{4},$$

where for every $h \in L_2(\nu)$, $\mathbb{E}_\nu(h) = \mathbb{E}h(X,Y)$. In particular, for $u(X,Y) = \widehat{f}_{n,1} - Y$ and $v(X,Y) = f_1(X) - Y$, the mid-point is $(u(X,Y) + v(X,Y))/2 = f_2(X) - Y$. Hence,

$$\begin{aligned}
R(f_2) = \mathbb{E}(f_2(X) - Y)^2 &= \mathbb{E}\Big(\frac{\widehat{f}_{n,1}(X) + f_1(X)}{2} - Y\Big)^2 \\
&\leq \frac{1}{2}\mathbb{E}(\widehat{f}_{n,1}(X) - Y)^2 + \frac{1}{2}\mathbb{E}(f_1(X) - Y)^2 - \frac{1}{4}\|f_{n,1} - f_1\|_{L_2(\mu)}^2 \\
&\leq \frac{1}{2}R(\widehat{f}_{n,1}) + \frac{1}{2}R(f_1) - \frac{d^2}{16},
\end{aligned}$$

where the expectations are taken conditioned on $D_{n,1}$. By Lemma 10 (see Appendix A.2 below), since $\widehat{f}_{n,1}, f_1 \in \widehat{F}_1$, we have

$$\frac{1}{2}R(\widehat{f}_{n,1}) + \frac{1}{2}R(f_1) \leq R(f^F) + c(\sigma_\varepsilon + b)\max(\phi d, b\phi^2),$$

and thus, since $f_2 \in \widehat{\mathcal{F}}$

$$R(f^{\widehat{\mathcal{F}}}) \leq R(f_2) \leq R(f^F) + c(\sigma_\varepsilon + b)\max(\phi d, b\phi^2) - cd^2.$$

Therefore,

$$\begin{aligned}
\beta &= c(\sigma_\varepsilon + b)\max(d\phi, b\phi^2) - \big(R(f^F) - R(f^{\widehat{\mathcal{F}}})\big) \\
&\leq c(\sigma_\varepsilon + b)\max(\phi d, b\phi^2) - cd^2.
\end{aligned}$$

Finally, if $d \geq c_{\sigma_\varepsilon, b}\phi$ then $\beta \leq 0$, otherwise $\beta \leq c_{\sigma_\varepsilon, b}\phi^2$. It concludes the proof of Theorem 2. $\square$

A.2 Tools from Empirical Process Theory and Technical Results

The following Theorem is a Talagrand's type concentration inequality (see Talagrand, 1996) for a class of unbounded functions.

Theorem 6 (Theorem 4, Adamczak, 2008) *Assume that $X, X_1, \ldots, X_n$ are independent random variables and F is a countable set of functions such that $\mathbb{E}f(X) = 0, \forall f \in F$ and $\|\sup_{f \in F} f(X)\|_{\psi_1} < +\infty$. Define*

$$Z := \sup_{f \in F}\Big|\frac{1}{n}\sum_{i=1}^n f(X_i)\Big|$$

and

$$\sigma^2 = \sup_{f \in F} \mathbb{E} f(X)^2 \text{ and } b := \| \max_{i=1,\dots,n} \sup_{f \in F} |f(X_i)| \|_{\psi_1}.$$

Then, for any $\eta \in (0,1)$ *and* $\delta > 0$*, there is* $c = c_{\eta,\delta}$ *such that for any* $x > 0$*:*

$$\mathbb{P}\Big[Z \geq (1+\eta)\mathbb{E}Z + \sigma\sqrt{2(1+\delta)\frac{x}{n}} + cb\Big(\frac{x}{n}\Big)\Big] \leq 4e^{-x}$$
$$\mathbb{P}\Big[Z \leq (1-\eta)\mathbb{E}Z - \sigma\sqrt{2(1+\delta)\frac{x}{n}} - cb\Big(\frac{x}{n}\Big)\Big] \leq 4e^{-x}.$$

Now we state some technical Lemmas, used in the proof of Theorem 2. Given a sample $(Z_i)_{i=1}^n$, we set the random empirical measure $P_n := n^{-1}\sum_{i=1}^n \delta_{Z_i}$. For any function f define $(P-P_n)(f) := n^{-1}\sum_{i=1}^n f(Z_i) - \mathbb{E} f(Z)$ and for a class of functions F, define $\|P-P_n\|_F := \sup_{f\in F} |(P-P_n)(f)|$. In all what follows, we denote by c an absolute positive constant, that can vary from place to place. Its dependence on the parameters of the setting is specified in place.

Lemma 7 *Define*

$$d(F) := \operatorname{diam}(F, L^2(\mu)), \quad \sigma^2(F) = \sup_{f\in F}\mathbb{E}[f(X)^2], \quad \mathcal{C} = \operatorname{conv}(F),$$

and $\mathcal{L}_{\mathcal{C}}(\mathcal{C}) = \{(Y - f(X))^2 - (Y - f^{\mathcal{C}}(X))^2 : f \in \mathcal{C}\}$*, where* $f^{\mathcal{C}} \in \operatorname{argmin}_{g\in\mathcal{C}} R(g)$*. If* (2) *holds, we have*

$$\mathbb{E}\Big[\sup_{f\in F}\frac{1}{n}\sum_{i=1}^n f^2(X_i)\Big] \leq c \max\Big(\sigma^2(F), \frac{b^2 \log M}{n}\Big), \text{ and}$$
$$\mathbb{E}\|P_n - P\|_{\mathcal{L}_{\mathcal{C}}(\mathcal{C})} \leq cb\sqrt{\frac{\log M}{n}} \max\Big(b\sqrt{\frac{\log M}{n}}, d(F)\Big).$$

If (3) *holds, we have*

$$\mathbb{E}\Big[\sup_{f\in F}\frac{1}{n}\sum_{i=1}^n f^2(X_i)\Big] \leq c \max\Big(\sigma^2(F), \frac{b^2 \log M}{n}\Big), \text{ and}$$
$$\mathbb{E}\|P_n - P\|_{\mathcal{L}_{\mathcal{C}}(\mathcal{C})} \leq cb\sqrt{\frac{\log M \log n}{n}} \max\Big(b\sqrt{\frac{\log M \log n}{n}}, d(F)\Big).$$

Proof First, consider the case (3). Define

$$r^2 = \sup_{f\in F}\frac{1}{n}\sum_{i=1}^n f(X_i)^2,$$

and note that $\mathbb{E}_X(r^2) \leq \mathbb{E}_X\|P - P_n\|_{F^2} + \sigma(F)^2$, where $F := \{f^2 : f \in F\}$. Using the Giné-Zinn symmetrization argument, see Giné and Zinn (1984), we have

$$\mathbb{E}_X\|P - P_n\|_{F^2} \leq \frac{c}{n}\mathbb{E}_X\mathbb{E}_g\Big[\sup_{f\in F}\Big|\sum_{i=1}^n g_i f^2(X_i)\Big|\Big],$$

where (g_i) are i.i.d. standard normal. The process $f \mapsto Z_{2,f} = \sum_{i=1}^n g_i f^2(X_i)$ is Gaussian, with intrinsic distance

$$\mathbb{E}_g|Z_{2,f} - Z_{2,f'}|^2 = \sum_{i=1}^n (f(X_i)^2 - f'(X_i)^2)^2 \leq d_{n,\infty}(f,f')^2 \times 4nr^2,$$

where $d_{n,\infty}(f,f') = \max_{i=1,\dots,n}|f(X_i) - f'(X_i)|$. Using (3) we have $d_{n,\infty}(f,f') \leq 2b$ for any $f, f' \in F$, so using Dudley's entropy integral, we have

$$\mathbb{E}_g\|P - P_n\|_{F^2} \leq \frac{c}{\sqrt{n}}\int_0^{2b}\sqrt{\log N(F, d_{n,\infty}, t)}dt \leq cr\sqrt{\frac{\log M}{n}}.$$

So, we get

$$\mathbb{E}_X\|P - P_n\|_{F^2} \leq cb\sqrt{\frac{\log M}{n}}\mathbb{E}_X[r] \leq cb\sqrt{\frac{\log M}{n}}\sqrt{\mathbb{E}_X[r^2]},$$

which entails that

$$\mathbb{E}_X(r^2) \leq c\max\Big(\frac{b^2\log M}{n} + \sigma(F)^2\Big).$$

Let us turn to the part of the Lemma concerning $\mathbb{E}\|P - P_n\|_{\mathcal{L}_C(C)}$. Recall that $C = \mathrm{conv}(F)$ and write for short $\mathcal{L}_f(X,Y) = \mathcal{L}_C(f)(X,Y) = (Y - f(X))^2 - (Y - f^C(X))^2$ for each $f \in C$, where we recall that $f^C \in \mathrm{argmin}_{g\in C}R(g)$. Using the same argument as before we have

$$\mathbb{E}\|P - P_n\|_{\mathcal{L}_C(C)} \leq \frac{c}{n}\mathbb{E}_{(X,Y)}\mathbb{E}_g\Big[\sup_{f\in C}\Big|\sum_{i=1}^n g_i\mathcal{L}_f(X_i,Y_i)\Big|\Big].$$

Consider the Gaussian process $f \in C \to Z_f := \sum_{i=1}^n g_i\mathcal{L}_f(X_i,Y_i)$ indexed by C. For every $f, f' \in C$, the intrinsic distance of $(Z_f)_{f\in C}$ satisfies

$$\begin{aligned}\mathbb{E}_g|Z_f - Z_{f'}|^2 &= \sum_{i=1}^n(\mathcal{L}_f(X_i,Y_i) - \mathcal{L}_{f'}(X_i,Y_i))^2\\ &\leq \max_{i=1,\dots,n}|2Y_i - f(X_i) - f'(X_i)|^2 \times \sum_{i=1}^n(f(X_i) - f'(X_i))^2\\ &= \max_{i=1,\dots,n}|2Y_i - f(X_i) - f'(X_i)|^2 \times \mathbb{E}_g|Z'_f - Z'_{f'}|^2,\end{aligned}$$

where $Z'_f := \sum_{i=1}^n g_i(f(X_i) - f^C(X_i))$. Therefore, by Slepian's Lemma, we have for every $(X_i,Y_i)_{i=1}^n$:

$$\mathbb{E}_g\Big[\sup_{f\in C}Z_f\Big] \leq \max_{i=1,\dots,n}\sup_{f,f'\in C}|2Y_i - f(X_i) - f'(X_i)| \times \mathbb{E}_g\Big[\sup_{f\in C}Z'_f\Big],$$

and since for every $f = \sum_{j=1}^M \alpha_j f_j \in C$, where $\alpha_j \geq 0, \forall j = 1,\dots,M$ and $\sum\alpha_j = 1$, $Z'_f = \sum_{j=1}^M \alpha_j Z_{f_j}$, we have

$$\mathbb{E}_g\Big[\sup_{f\in C}Z'_f\Big] \leq \mathbb{E}_g\Big[\sup_{f\in F}Z'_f\Big].$$

Moreover, we have, using Dudley's entropy integral argument,

$$\frac{1}{n}\mathbb{E}_g\Big[\sup_{f\in F}Z'_f\Big] \leq \frac{c}{\sqrt{n}}\int_0^{\Delta_n(F')}\sqrt{N(F,\|\cdot\|_n,t)}dt \leq c\sqrt{\frac{\log M}{n}}r',$$

where $F' := \{f - f^{\mathcal{C}} : f \in F\}$ and $\Delta_n(F') := \operatorname{diam}(F', \|\cdot\|_n)$ and

$$r'^2 := \sup_{f \in F'} \frac{1}{n} \sum_{i=1}^n f(X_i)^2.$$

Hence, we proved that

$$\mathbb{E}\|P - P_n\|_{\mathcal{L}_{\mathcal{C}}(\mathcal{C})} \leq c\sqrt{\frac{\log M}{n}} \sqrt{\mathbb{E}\Big[\max_{i=1,\ldots,n} |2Y_i - f(X_i) - f'(X_i)|^2\Big]} \sqrt{\mathbb{E}(r'^2)}.$$

Using Pisier's inequality for ψ_1 random variables and the fact that $\mathbb{E}(U^2) \leq 4\|U\|_{\psi_1}$ for any ψ_1-random variable U, together with (3), we obtain that

$$\mathbb{E}\Big[\max_{i=1,\ldots,n} \sup_{f,f' \in \mathcal{C}} |2Y_i - f(X_i) - f'(X_i)|^2\Big] \leq cb^2 \log(n). \tag{8}$$

So, we finally obtain

$$\mathbb{E}\|P - P_n\|_{\mathcal{L}_{\mathcal{C}}(\mathcal{C})} \leq c\sqrt{\frac{\log n \log M}{n}} \sqrt{\mathbb{E}(r'^2)},$$

and the conclusion follows from the first part of the Lemma, since $\sigma(F') \leq d(F)$. The case (2) is easier and follows from the fact that the left hand side of (8) is smaller than $4b$. ■

Lemma 7 combined with Theorem 6 leads to the following corollary.

Corollary 8 *Let $d(F) = \operatorname{diam}(F, L^2(\mu))$, $\mathcal{C} := \operatorname{conv}(F)$ and $\mathcal{L}_f(X,Y) = (Y - f(X))^2 - (Y - f^{\mathcal{C}}(X))^2$ for any $f \in \mathcal{C}$.*

If (3) holds, we have, with probability larger than $1 - 4e^{-x}$, that for every $f \in \mathcal{C}$:

$$\Big|\frac{1}{n}\sum_{i=1}^n \mathcal{L}_f(X_i, Y_i) - \mathbb{E}\mathcal{L}_f(X,Y)\Big| \leq c(\sigma_\varepsilon + b)\sqrt{\frac{(\log M + x)\log n}{n}} \max\Big(b\sqrt{\frac{(\log M + x)\log n}{n}}, d(F)\Big).$$

If (2) holds, we have, with probability larger than $1 - 2e^{-x}$, that for every $f \in \mathcal{C}$:

$$\Big|\frac{1}{n}\sum_{i=1}^n \mathcal{L}_f(X_i, Y_i) - \mathbb{E}\mathcal{L}_f(X,Y)\Big| \leq cb\sqrt{\frac{\log M + x}{n}} \max\Big(b\sqrt{\frac{\log M + x}{n}}, d(F)\Big).$$

Proof Applying Theorem 6 to

$$Z := \sup_{f \in \mathcal{C}} \Big|\frac{1}{n}\sum_{i=1}^n \mathcal{L}_f(X_i, Y_i) - \mathbb{E}\mathcal{L}_f(X,Y)\Big|,$$

we obtain that, with a probability larger than $1 - 4e^{-x}$:

$$Z \leq c\Big(\mathbb{E}Z + \sigma(\mathcal{C})\sqrt{\frac{x}{n}} + b_n(\mathcal{C})\frac{x}{n}\Big),$$

where

$$\sigma(\mathcal{C})^2 = \sup_{f \in \mathcal{C}} \mathbb{E}[\mathcal{L}_f(X,Y)^2], \quad \text{and}$$
$$b_n(\mathcal{C}) = \Big\| \max_{i=1,\dots,n} \sup_{f \in \mathcal{C}} |\mathcal{L}_f(X_i,Y_i) - \mathbb{E}[\mathcal{L}_f(X,Y)]| \Big\|_{\psi_1}.$$

Since

$$\mathcal{L}_f(X,Y) = 2\varepsilon(f^{\mathcal{C}}(X) - f(X)) + (f^{\mathcal{C}}(X) - f(X))(2f_0(X) - f(X) - f^{\mathcal{C}}(X)), \tag{9}$$

we have using Assumptions 1 and (3):

$$\mathbb{E}[\mathcal{L}_f(X,Y)^2] \le (4\sigma_\varepsilon^2 + 2b^2)\|f - f^{\mathcal{C}}\|_{L^2(\mu)}^2,$$

meaning that

$$\sigma(\mathcal{C})^2 \le (4\sigma_\varepsilon^2 + 2b^2)d(F).$$

Since $\mathbb{E}(|Z|) \le \|Z\|_{\psi_1}$, we have $b_n(\mathcal{C}) \le 2\log(n+1)\|\sup_{f\in\mathcal{C}}|\mathcal{L}_f(X,Y)|\|_{\psi_1}$. Moreover, using again (9), we obtain that

$$b_n(\mathcal{C}) \le 16\log(n+1)b^2.$$

Putting all this together, and using Lemma 7, we arrive at

$$Z \le c(\sigma_\varepsilon + b)\sqrt{\frac{(\log M + x)\log n}{n}} \max\Big(b\sqrt{\frac{(\log M + x)\log n}{n}}, d(F)\Big),$$

with probability larger than $1 - 4e^{-x}$ for any $x > 0$. In the bounded case (2) the proof is easier, and one can use the original Talagrand's concentration inequality. ■

Lemma 9 *Let $\mathcal{L}_f(X,Y) = (Y - f(X))^2 - (Y - f^F(X))^2$ for any $f \in F$.*

If (3) holds, we have with probability larger than $1 - 4e^{-x}$, that for every $f \in F$:

$$\Big|\frac{1}{n}\sum_{i=1}^n \mathcal{L}_f(X_i,Y_i) - \mathbb{E}\mathcal{L}_f(X,Y)\Big|$$
$$\le c(\sigma_\varepsilon + b)\sqrt{\frac{(\log M + x)\log n}{n}} \max\Big(b\sqrt{\frac{(\log M + x)\log n}{n}}, \|f - f^F\|\Big).$$

Also, with probability at least $1 - 4e^{-x}$, we have for every $f, g \in F$:

$$\big|\|f - g\|_n^2 - \|f - g\|^2\big|$$
$$\le cb\sqrt{\frac{(\log M + x)\log n}{n}} \max\Big(b\sqrt{\frac{(\log M + x)\log n}{n}}, \|f - g\|\Big).$$

If (2) holds, we have, with probability larger than $1 - 2e^{-x}$, that for every $f \in F$:

$$\Big|\frac{1}{n}\sum_{i=1}^n \mathcal{L}_f(X_i,Y_i) - \mathbb{E}\mathcal{L}_f(X,Y)\Big| \le cb\sqrt{\frac{\log M + x}{n}} \max\Big(b\sqrt{\frac{\log M + x}{n}}, \|f - f^F\|\Big),$$

and with probability at least $1 - 2e^{-x}$, that for every $f, g \in F$:

$$\big|\|f - g\|_n^2 - \|f - g\|^2\big| \le cb\sqrt{\frac{\log M + x}{n}} \max\Big(b\sqrt{\frac{\log M + x}{n}}, \|f - g\|\Big).$$

Proof [Proof of Lemma 9] The proof uses exactly the same arguments as that of Lemma 7 and Corollary 8, and thus is omitted. ■

Lemma 10 *Let $\widehat{F}_1$ be given by* (4) *and recall that $f^F \in \operatorname{argmin}_{f\in F} R(f)$ and let $d(\widehat{F}_1) = \operatorname{diam}(\widehat{F}_1, L_2(\mu))$.*

If (3) *holds, we have with probability at least $1-4\exp(-x)$ that $f^F \in \widehat{F}_1$, and any function $f \in \widehat{F}_1$ satisfies*

$$R(f) \le R(f^F) + c(\sigma_\varepsilon + b)\sqrt{\frac{(\log M + x)\log n}{n}} \max\Big(b\sqrt{\frac{(\log M + x)\log n}{n}}, d(\widehat{F}_1)\Big).$$

If (2) *holds, we have with probability at least $1-2\exp(-x)$ that $f^F \in \widehat{F}_1$, and any function $f \in \widehat{F}_1$ satisfies*

$$R(f) \le R(f^F) + cb\sqrt{\frac{\log M + x}{n}} \max\Big(b\sqrt{\frac{\log M + x}{n}}, d(\widehat{F}_1)\Big).$$

Proof The proof follows the lines of the proof of Lemma 4.4 in Lecué and Mendelson (2009), together with Lemma 9, so we don't reproduce it here. ■

References

Radosław Adamczak. A tail inequality for suprema of unbounded empirical processes with applications to Markov chains. *Electron. J. Probab.*, 13:no. 34, 1000–1034, 2008. ISSN 1083-6489.

Jean-Yves Audibert. Fast learning rates in statistical inference through aggregation. *Ann. Statist.*, 37:1591, 2009. URL `doi:10.1214/08-AOS623`.

Olivier Catoni. *Statistical Learning Theory and Stochastic Optimization*. Ecole d'été de Probabilités de Saint-Flour 2001, Lecture Notes in Mathematics. Springer, N.Y., 2001.

Arnak S. Dalalyan and Alexandre B. Tsybakov. Aggregation by exponential weighting and sharp oracle inequalities. In *COLT*, pages 97–111, 2007.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 2004. ISSN 0090-5364. With discussion, and a rejoinder by the authors.

Evarist Giné and Joel Zinn. Some limit theorems for empirical processes. *Ann. Probab.*, 12(4): 929–998, 1984. ISSN 0091-1798.

Anatoli Juditsky, Alexander V. Nazin, Alexandre B. Tsybakov, and Nicolas Vayatis. Recursive aggregation of estimators by the mirror descent method with averaging. *Problemy Peredachi Informatsii*, 41(4):78–96, 2005. ISSN 0555-2923.

Anatoli Juditsky, Philippe Rigollet, and Alexandre B. Tsybakov. Learning by mirror averaging. *Ann. Statist.*, 36(5):2183–2206, 2008. ISSN 0090-5364. doi: 10.1214/07-AOS546. URL `http://dx.doi.org/10.1214/07-AOS546`.

Guillaume Lecué and Shahar Mendelson. Aggregation via empirical risk minimization. *Probab. Theory Related Fields*, 145(3-4):591–613, 2009. ISSN 0178-8051. doi: 10.1007/s00440-008-0180-8. URL `https://dx.doi.org/10.1007/s00440-008-0180-8`.

Guillaume Lecué and Shahar Mendelson. On the optimality of the aggregate with exponential weights for small temperatures. *Submitted*, 2010.

Gilbert Leung and Andrew R. Barron. Information theory and mixing least-squares regressions. *IEEE Trans. Inform. Theory*, 52(8):3396–3410, 2006. ISSN 0018-9448.

Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

Michel Talagrand. New concentration inequalities in product spaces. *Invent. Math.*, 126(3):505–563, 1996. ISSN 0020-9910.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58 (1):267–288, 1996. ISSN 0035-9246.

Alexandre. B. Tsybakov. Optimal rates of aggregation. *Computational Learning Theory and Kernel Machines. B.Schölkopf and M.Warmuth, eds. Lecture Notes in Artificial Intelligence*, 2777:303–313, 2003. Springer, Heidelberg.

Yuhong Yang. Mixing strategies for density estimation. *Ann. Statist.*, 28(1):75–87, 2000. ISSN 0090-5364. doi: 10.1214/aos/1016120365. URL `http://dx.doi.org/10.1214/aos/1016120365`.

Yuhong Yang. Aggregating regression procedures to improve performance. *Bernoulli*, 10(1):25–47, 2004. ISSN 1350-7265.

Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(2):301–320, 2005. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2005.00503.x. URL `http://dx.doi.org/10.1111/j.1467-9868.2005.00503.x`.

Journal of Machine Learning Research 12 (2011) 1835-1863 Submitted 8/08; Revised 4/10; Published 6/11

A Refined Margin Analysis for Boosting Algorithms via Equilibrium Margin

Liwei Wang WANGLW@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception, MOE
School of Electronics Engineering and Computer Science
Peking University
Beijing, 100871, P.R.China

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

Zhaoxiang Jing JINGZX@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception, MOE
School of Electronics Engineering and Computer Science
Peking University
Beijing, 100871, P.R.China

Cheng Yang YANGCHENUG@GMAIL.COM
Beijing Aerospace Control Center
Beijing, 100094, P.R.China

Zhi-Hua Zhou ZHOUZH@NJU.EDU.CN
National Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210093, P.R. China

Jufu Feng FJF@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception, MOE
School of Electronics Engineering and Computer Science
Peking University
Beijing, 100871, P.R.China

Editor: Manfred Warmuth

Abstract

Much attention has been paid to the theoretical explanation of the empirical success of AdaBoost. The most influential work is the margin theory, which is essentially an upper bound for the generalization error of any voting classifier in terms of the margin distribution over the training data. However, important questions were raised about the margin explanation. Breiman (1999) proved a bound in terms of the minimum margin, which is sharper than the margin distribution bound. He argued that the minimum margin would be better in predicting the generalization error. Grove and Schuurmans (1998) developed an algorithm called LP-AdaBoost which maximizes the minimum margin while keeping all other factors the same as AdaBoost. In experiments however, LP-AdaBoost usually performs worse than AdaBoost, putting the margin explanation into serious doubt. In this paper, we make a refined analysis of the margin theory. We prove a bound in terms of a new margin measure called the *Equilibrium margin (Emargin)*. The Emargin bound is uniformly sharper than Breiman's minimum margin bound. Thus our result suggests that the minimum margin may be not crucial for the generalization error. We also show that a large Emargin and a small empirical error at Emargin imply a smaller bound of the generalization error. Experimental results on benchmark data sets demonstrate that AdaBoost usually has a larger Emargin and a smaller test error than LP-AdaBoost, which agrees well with our theory.

Keywords: boosting, margin bounds, voting classifier

1. Introduction

The AdaBoost algorithm (Freund and Schapire, 1996, 1997) has achieved great success in the past ten years. It has demonstrated excellent experimental performance both on benchmark data sets and real applications (Bauer and Kohavi, 1999; Dietterich, 2000; Viola and Jones, 2001; Wang et al., 2007). According to a recent evaluation (Caruana and Niculescu-Mizil, 2006), boosting with decision trees as base learners is the leading classification algorithm. An important property of boosting is its relative (although not complete) resistance to overfitting. On many data sets it is observed that the test error keeps decreasing even after thousands of base classifiers have been combined (Breiman, 1998; Quinlan, 1996). This fact, at first sight, obviously violates Occam's razor.

Considerable efforts have been made to explain the "mystery" of boosting. Friedman et al. (2000) related boosting to fitting an additive logistic regression model. From this statistical view they developed the LogitBoost algorithm. Jiang (2004), Lugosi and Vayatis (2004), Zhang (2004), Bartlett et al. (2006) and others proved that boosting is Bayes consistent if it is properly regularized. These works provide deep understanding of boosting. However, these explanations each focused on some aspects of boosting. The consistency assures that boosting is asymptotically optimal, but it does not explain boosting's effectiveness on small sample problems. The statistical view led to many new algorithms, but left boosting's relative resistance to overfitting not well explained. Boosting algorithms involve several factors such as the type of base classifiers, regularization methods and loss functions to minimize. Recently, Mease and Wyner (2008) studied the effects of these factors. They provided a number of examples that are contrary to previous theoretical explanations.

Schapire et al. (1998) tried to give a comprehensive explanation in terms of the margins of the training examples. Roughly speaking, the margin of an example with respect to a classifier is a measure of the confidence of the classification result. They proved an upper bound for the generalization error of a voting classifier that does not depend on how many classifiers were combined, but only on the margin distribution over the training set, the number of the training examples and the size (the VC dimension for example) of the set of base classifiers. They also demonstrated that AdaBoost has the ability to produce a "good" margin distribution. This theory suggests that producing a good margin distribution is the key to the success of AdaBoost and explains well its relative resistance to overfitting.

Soon after that however, there were serious doubt cast on this margin explanation. First Breiman (1999) and Grove and Schuurmans (1998) developed algorithms that maximize the *minimum margin*. (Minimum margin is the smallest margin over all training examples, see Section 2 for the formal definition). Breiman (1999) then gave an upper bound for the generalization error of a voting classifier in terms of the minimum margin, as well as the number of training examples and the size of the set of base classifiers. This bound is sharper than the bound based on the margin distribution given in Schapire et al. (1998). Breiman (1999) argued that if the bound of Schapire et al. implied that the margin distribution is important to the generalization error, his bound implied more strongly that the minimum margin is the key to the generalization error, and the minimum margin maximizing algorithms would achieve better performance than AdaBoost.

Grove and Schuurmans (1998) conducted a rigorous experimental comparison on the minimum margin. They developed an algorithm called LP-AdaBoost which first uses AdaBoost to train a series of base classifiers. Then by linear programming they obtained coefficients of the base classifiers, whose linear combination has the largest possible minimum margin. Thus LP-AdaBoost and AdaBoost have all relevant factors the same except the coefficients of the base classifiers. According to the minimum margin bound, LP-AdaBoost would have smaller generalization error than AdaBoost. In experiments, although LP-AdaBoost always achieves larger minimum margins, its test error is higher than AdaBoost on most data sets. This result puts the margin theory into serious doubt.

In this paper we provide a refined analysis of the margin theory. We propose a new upper bound for the generalization error of voting classifiers. This bound is uniformly sharper than Breiman's minimum margin bound. The key factor in this bound is a new margin notion which we refer to as the *Equilibrium margin (Emargin)*. The Emargin can be viewed as a measure of how good a margin distribution is. In fact, the Emargin depends, in a complicated way, on the margin distribution, and has little relation to the minimum margin. Experimental results show that AdaBoost usually produces a larger Emargin than LP-AdaBoost, which agrees with the Emargin explanation.

The margin theory has been studied and greatly improved by several authors. Especially Koltchinskii and Panchenko (2002, 2005) developed new tools for empirical processes and prove much sharper margin distribution bounds. However it is difficult to compare these bounds to the minimum margin bound of Breiman (1999), since they contain unspecified constants. Nevertheless, these results suggest that the margin distribution may be more important than the minimum margin for the generalization error of voting classifiers.

We also show that if a boosting algorithm returns a classifier that minimizes the Emargin bound or the margin distribution bound of Schapire et al. (1998) then the classifier learned converges to the best classifier in the hypothesis space as the number of training examples goes to infinity.

The rest of this paper is organized as follows: In Section 2 we briefly describe the background of the margin theory. Our main results—the Emargin bounds are given in Section 3. We provide further explanation of the main bound in Section 4 and the consistency results in Section 5. All the proofs can be found in Section 6. We provide experimental justification in Section 7 and conclude in Section 8.

2. Background

Consider binary classification problems. Examples are drawn independently according to an underlying distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1,+1\}$, where $\mathcal{X}$ is an instance space. Let $\mathcal{H}$ denote the space from which the base hypotheses are chosen. A base hypothesis $h \in \mathcal{H}$ is a mapping from $\mathcal{X}$ to $\{-1,+1\}$. A voting classifier $f(x)$ is of the form

$$f(x) = \sum_i \alpha_i h_i(x), \quad h_i \in \mathcal{H},$$

where

$$\sum \alpha_i = 1, \quad \alpha_i \geq 0.$$

An error occurs on an example (x,y) if and only if

$$yf(x) \leq 0.$$

We use $P_{\mathcal{D}}(A(x,y))$ to denote the probability of the event A when an example (x,y) is chosen randomly according to the distribution $\mathcal{D}$. Therefore, $P_{\mathcal{D}}(yf(x) \leq 0)$ is the generalization error of f which we want to bound. Let $\mathcal{S}$ be a training set containing n examples. We use $P_{\mathcal{S}}(A(x,y))$ to denote the probability with respect to choosing an example (x,y) uniformly at random from $\mathcal{S}$.

For an example (x,y), the value of $yf(x)$ reflects the confidence of the prediction. Since each base classifier outputs -1 or $+1$, one has

$$yf(x) = \sum_{i:y=h_i(x)} \alpha_i - \sum_{i:y\neq h_i(x)} \alpha_i.$$

Hence $yf(x)$ is the difference between the weights assigned to those base classifiers that correctly classify (x,y) and the weights assigned to those that misclassify the example. $yf(x)$ is called the *margin* for (x,y) with respect to f. If we consider the margins over the whole set of training examples, we can regard $P_{\mathcal{S}}(yf(x) \leq \theta)$ as a distribution over θ $(-1 \leq \theta \leq 1)$, since $P_{\mathcal{S}}(yf(x) \leq \theta)$ is the fraction of training examples whose margin is at most θ. This distribution is referred to as the *margin distribution*.

A description of AdaBoost is shown in Algorithm 1. In AdaBoost the linear coefficients α_t is set as

$$\alpha_t = \frac{1}{2} \log \frac{1+\gamma_t}{1-\gamma_t},$$

where γ_t is defined as:

$$\gamma_t = \sum_{i=1}^{n} D_t(i) y_i h_t(x_i).$$

γ_t is an affine transformation of the error rate of h_t with respect to the weight distribution D_t.

AdaBoost often does not overfit. Although it is known that boosting forever does overfit when there is high classification noise, on many data sets the performance of AdaBoost keeps improving even after a large number of rounds.

The first margin explanation (Schapire et al., 1998) of the AdaBoost algorithm is to upper bound the generalization error of voting classifiers in terms of the margin distribution, the number of training examples and the complexity of the set from which the base classifiers are chosen. The theory contains two bounds: one applies to the case that the base classifier set $\mathcal{H}$ is finite, and the other applies to the general case that $\mathcal{H}$ has a finite VC dimension.

Theorem 1 *(Schapire et al., 1998) For any $\delta > 0$, with probability at least $1-\delta$ over the random choice of the training set $\mathcal{S}$ of n examples, every voting classifier f satisfies the following bounds:*

$$P_{\mathcal{D}}\Big(yf(x) \leq 0\Big) \leq \inf_{\theta \in (0,1]} \left[P_{\mathcal{S}}\Big(yf(x) \leq \theta\Big) + O\left(\frac{1}{\sqrt{n}} \left(\frac{\log n \log |\mathcal{H}|}{\theta^2} + \log \frac{1}{\delta} \right)^{1/2} \right) \right],$$

if $|\mathcal{H}| < \infty$. And

$$P_{\mathcal{D}}\Big(yf(x) \leq 0\Big) \leq \inf_{\theta \in (0,1]} \left[P_{\mathcal{S}}\Big(yf(x) \leq \theta\Big) + O\left(\frac{1}{\sqrt{n}} \left(\frac{d \log^2(n/d)}{\theta^2} + \log \frac{1}{\delta} \right)^{1/2} \right) \right],$$

where d is the VC dimension of $\mathcal{H}$.

```
**Input**: $T,\ S = \{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$
        where $x_i \in \mathcal{X}, y_i \in \{-1,1\}$.
**Initialization:** $D_1(i) = 1/n$.
**for** $t = 1$ **to** $T$ **do**
    1. Train a base classifier $h_t \in \mathcal{H}$ using distribution $D_t$, where $h_t : \mathcal{X} \to \{-1,1\}$.
    2. Choose $\alpha_t$.
    3. Update:
        $$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$
    where $Z_t$ is the normalization factor chosen so that $D_{t+1}$ will be a distribution.
**end**
**Output**: The final classifier
        $$F(x) = \mathrm{sgn}\left(f(x)\right),$$
    where
        $$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x).$$
```

Algorithm 1: A description of AdaBoost.

The theorem states that if the voting classifier generates a good margin distribution, that is, most training examples have large margins so that $P_{\mathcal{S}}(yf(x) \le \theta)$ is small for not too small θ, then the upper bound of the generalization error is also small. In Schapire et al. (1998) it has also been shown that for the AdaBoost algorithm, $P_{\mathcal{S}}(yf(x) \le \theta)$ decreases to zero exponentially fast with respect to the number of boosting iterations if θ is not too large. These results suggest that the excellent performance of AdaBoost is due to its good margin distribution.

Another important notion is the minimum margin which is the smallest margin achieved on the training set. Formally, the minimum margin, denoted by θ_0, of a voting classifier f on a training set $\mathcal{S}$ is defined as

$$\theta_0 = \min\left\{yf(x) :\ (x,y) \in \mathcal{S}\right\}.$$

Breiman (1999) proved an upper bound for the generalization error of voting classifiers which depends only on the minimum margin, not on the entire margin distribution.

Theorem 2 *(Breiman, 1999) Assume that $|\mathcal{H}| < \infty$. Let θ_0 be a real number that satisfies $\theta_0 > 4\sqrt{\frac{2}{|\mathcal{H}|}}$ and*

$$R = \frac{32\log(2|\mathcal{H}|)}{n\theta_0^2} \le 2n.$$

Then for any $\delta > 0$, with probability at least $1-\delta$ over the random choice of the training set $\mathcal{S}$ of n examples, every voting classifier f whose minimum margin on $\mathcal{S}$ is at least θ_0 satisfies the following bound:

$$P_{\mathcal{D}}\Big(yf(x) \le 0\Big) \le R\left(\log(2n) + \log\frac{1}{R} + 1\right) + \frac{1}{n}\log\left(\frac{|\mathcal{H}|}{\delta}\right).$$

Breiman (1999) pointed out that his bound is sharper than the margin distribution bound of Schapire et al. If θ in Theorem 1 is taken to be the minimum margin θ_0, the bound in Theorem 2 is about the square of the bound in terms of the margin distribution, since the bound in Theorem 2 is $O\left(\frac{\log n}{n\theta_0^2}\right)$ and the bound in Theorem 1 is $O\left(\sqrt{\frac{\log n}{n\theta_0^2}}\right)$. Breiman then argued that compared to the margin distribution explanation, his bound implied more strongly that the minimum margin governs the generalization error.

Several authors developed algorithms to maximize the minimum margin. Among these, the most representative one is the LP-AdaBoost proposed by Grove and Schuurmans (1998). Let $h_1, \ldots, h_T$ be the base classifiers returned by AdaBoost on the training examples $\{(x_i, y_i),\ i = 1, \ldots, n\}$. Finding a voting classifier $g = \sum_{t=1}^{T} \beta_t h_t$ such that g maximizes the minimum margin can be formulated as a linear programming problem.

$$\begin{aligned}
&\max_{\beta, m} \ m \\
&s.t.\ \ y_i \sum_{t=1}^{T} \beta_t h_t(x_i) \geq m, \quad i = 1, 2, \ldots, n \\
&\qquad \beta_t \geq 0, \quad \sum_{t=1}^{T} \beta_t = 1,
\end{aligned}$$

where $\beta = (\beta_1, \cdots, \beta_T)$. Grove and Schuurmans called this algorithm LP-AdaBoost.

Comparing the performance of AdaBoost and LP-AdaBoost is a good test of significance of the minimum margin bound. Except the linear coefficients, the voting classifiers obtained by the two algorithms have all relevant factors the same. In experiments, although LP-AdaBoost always produces larger minimum margins, its test error is higher than AdaBoost more often than not. This result is different from what the minimum margin bound suggests and therefore puts the margin explanation into serious doubt.

Breiman (1999) and Meir and Rätsch (2003) developed arc-gv to maximize the minimum margin. Arc-gv can also be described by Algorithm 1. The only difference from AdaBoost is how to set α_t at each round. It can be shown that arc-gv converges to the maximum margin solution (Rätsch and Warmuth, 2005; Rudin et al., 2007) whereas AdaBoost does not always do this (Rudin et al., 2004). However on some data sets AdaBoost has larger minimum margin than arc-gv after a finite number of rounds. Also note that arc-gv and AdaBoost generate different base classifiers. Recently Reyzin and Schapire (2006) gained an important discovery that when Breiman (1999) tried to maximize the minimum margin by arc-gv, he had not make a good control of the complexity of the base classifiers, while comparing the margin is only meaningful when the complexity of base learners are the same.

3. Emargin Bounds

In this section we propose upper bounds in terms of the Emargin. The bounds are sharper than the minimum margin bound.

First let us introduce some notions. Consider the Bernoulli relative entropy function $D(q||p)$ defined as

$$D(q||p) = q \log \frac{q}{p} + (1-q) \log \frac{1-q}{1-p}, \qquad 0 \leq p, q \leq 1.$$

By convention, let $D(0||0) = 0$.

For a fixed q, $D(q||p)$ is a monotone increasing function of p for $q \le p \le 1$. It is easy to check that

$$D(q||p) = 0 \quad \text{when } p = q,$$

and

$$D(q||p) \to \infty \quad \text{as } p \to 1.$$

Thus one can define the inverse function of $D(q||p)$ for fixed q as $D^{-1}(q,u)$, such that

$$D\left(q||D^{-1}(q,u)\right) = u \quad \text{for all } u \ge 0 \text{ and } D^{-1}(q,u) \ge q.$$

See also Langford (2005).

The next theorem is our main result: the Emargin bound. Here we consider the case that the base classifier set $\mathcal{H}$ is finite. For the case that $\mathcal{H}$ is infinite but has a finite VC dimension, the bound is more complicated and will be given in Theorem 7. All the proofs can be found in Section 6.

Theorem 3 *If $8 < |\mathcal{H}| < \infty$, then for any $\delta > 0$, with probability at least $1-\delta$ over the random choice of the training set $\mathcal{S}$ of n examples ($n > 1$), every voting classifier f such that*

$$q_0 = P_{\mathcal{S}}\left(yf(x) \le \sqrt{\frac{8}{|\mathcal{H}|}}\right) < 1.$$

satisfies the following bound:

$$P_{\mathcal{D}}\Big(yf(x) \le 0\Big) \le \frac{\log|\mathcal{H}|}{n} + \inf_{q \in \{q_0, q_0+\frac{1}{n}, \ldots, \frac{n-1}{n}\}} D^{-1}\left(q, u\left[\hat{\theta}(q)\right]\right), \tag{1}$$

where

$$\hat{\theta}(q) = \sup\left\{\theta \in (0,1] : P_{\mathcal{S}}\Big(yf(x) \le \theta\Big) \le q\right\}, \tag{2}$$

$$u(\theta) = \frac{1}{n}\left(\frac{8}{\theta^2}\log\left(\frac{2n^2}{\log|\mathcal{H}|}\right)\log(2|\mathcal{H}|) + 2\log|\mathcal{H}| + \log\frac{n}{\delta}\right).$$

Note that the assumption $q_0 < 1$ in the theorem is very mild since it implies that at least one training example has a large margin (larger than $8/|\mathcal{H}|$), or equivalently the *largest* margin is not too small.[1] This contrasts with the fact that the minimum margin bound applies when the *minimum* margin is not too small.

Clearly the key factors in this bound are the optimal q and the corresponding $\hat{\theta}(q)$.

Definition 4 *Let q^* be the optimal q in Equation (1), and denote*

$$\theta^* = \hat{\theta}(q^*).$$

*We call θ^* the Equilibrium margin (**Emargin**). It can be seen that q^* is the empirical error at margin θ^*, that is,*

$$q^* = P_{\mathcal{S}}(yf(x) < \theta^*).$$

q^* will be referred to as the Emargin error.

1. This observation is due to a reviewer.

With Definition 4, the Emargin bound (1) can be simply written as

$$P_{\mathcal{D}}\Big(yf(x)\le 0\Big)\le \frac{\log|\mathcal{H}|}{n}+D^{-1}\Big(q^*,u(\theta^*)\Big).$$

Theorem 3 provides an upper bound of the generalization error of a voting classifier that depends on its Emargin and the Emargin error.

Our Emargin bound has a similar flavor to Theorem 1. Note that the Emargin depends, in a complicated way, on the whole margin distribution. Roughly, if most training examples have large margins, then θ^* is large and q^* is small. The minimum margin is only a special case of the Emargin. From (2) one can see that $\hat{\theta}(0)$ is the minimum margin. Hence the Emargin is equal to the minimum margin if and only if the optimal q^* is zero.

We next compare our Emargin bound to the minimum margin bound. We show that the Emargin bound is sharper than the minimum margin bound. Since the minimum margin bound applies only to the separable case, that is, $\theta_0 > 0$, we assume that the conditions in Theorem 2 are satisfied.

Theorem 5 *Assume that the minimum margin* θ_0 *is larger than* 0. *Then the bound given in Theorem 3 is uniformly sharper than the minimum margin bound in Theorem 2. That is, if*

$$R=\frac{32\log(2|\mathcal{H}|)}{n\theta_0^2}\le 2n,$$

then

$$\frac{\log|\mathcal{H}|}{n}+D^{-1}\Big(q^*,u(\theta^*)\Big)\le R\left(\log(2n)+\log\frac{1}{R}+1\right)+\frac{1}{n}\log\frac{|\mathcal{H}|}{\delta}.$$

This theorem suggests that the Emargin and Emargin error may be more relevant to the generalization error than the minimum margin. The following theorem describes how the Emargin θ^* and the Emargin error q^* affect the upper bound of the generalization ability. It states that a larger Emargin and a smaller Emargin error result in a lower generalization error bound.

Theorem 6 *Let* f_1, f_2 *be two voting classifiers. Denote by* θ_1, θ_2 *the Emargins and by* q_1, q_2 *the Emargin errors of* f_1, f_2 *respectively. Thus*

$$q_i=P_S\Big(yf_i(x)<\theta_i\Big),\qquad i=1,2.$$

Also denote by B_1, B_2 *the Emargin upper bounds of the generalization error of* f_1, f_2 *(i.e., the right-hand side of* (1)*). Then*

$$B_1\le B_2,$$

if

$$\theta_1\ge\theta_2 \text{ and } q_1\le q_2.$$

Theorem 6 suggests that the Emargin and the Emargin error can be used as measures of the quality of a margin distribution. A large Emargin and a small Emargin error indicate a good margin distribution. Experimental results in Section 7 show that AdaBoost often has larger Emargins and smaller Emargin errors than LP-AdaBoost.

The last theorem of this section is the Emargin bound for the case that the set of base classifiers has a finite VC dimension.

Theorem 7 *Suppose the set of base classifiers $\mathcal{H}$ has VC dimension d. Then for any $\delta > 0$, with probability at least $1-\delta$ over the random choice of the training set $\mathcal{S}$ of n examples, every voting classifier f satisfies the following bound:*

$$P_{\mathcal{D}}\Big(yf(x) \le 0\Big) \le \frac{d^2+1}{n} + \inf_{q \in \{q_0, q_0+\frac{1}{n}, \ldots, \frac{n-1}{n}\}} \frac{n-1}{n} \cdot D^{-1}\left(q, u\left[\hat{\theta}(q)\right]\right), \tag{3}$$

where

$$\hat{\theta}(q) = \sup\left\{\theta \in \big(0,1\big] : P_{\mathcal{S}}\Big(yf(x) \le \theta\Big) \le q\right\},$$

and

$$u(\theta) = \frac{1}{n}\left(\frac{16d}{\theta^2}\log\frac{n}{d}\log\frac{en^2}{d} + 3\log\left(\frac{16}{\theta^2}\log\frac{n}{d}+1\right) + \log\frac{2n}{\delta}\right),$$

provided $q_0 = P_{\mathcal{S}}(yf(x) \le 0) < 1$.

4. Explanation of the Emargin Bound

In Theorem 3, we adopted the partial inverse of the relative entropy to upper bound the generalization error. The key term in the Emargin bound is $\inf_q D^{-1}(q, u[\hat{\theta}(q)])$. To better understand the bound, we make use of three different upper bounds of $\inf_q D^{-1}(q, u[\hat{\theta}(q)])$ to obtain simpler forms and give explanations of the Emargin bound. We list in the following lemma the upper bounds of $\inf_q D^{-1}(q, u[\hat{\theta}(q)])$.

Lemma 8 *Let $u[\hat{\theta}(q)]$ be the one defined in Theorem 3. Let $\Gamma = \{q_0, q_0+\frac{1}{n}, \ldots, \frac{n-1}{n}\}$, where q_0 was defined in Theorem 3. Then the following bounds hold. (In the first bound we assume that $q_0 = 0$.)*

$$\inf_{q\in\Gamma} D^{-1}\left(q, u\left[\hat{\theta}(q)\right]\right) \le D^{-1}\left(0, u\left[\hat{\theta}(0)\right]\right) \le u\left[\hat{\theta}(0)\right].$$

$$\inf_{q\in\Gamma} D^{-1}\left(q, u\left[\hat{\theta}(q)\right]\right) \le \inf_{q\in\Gamma}\left(q + \left(\frac{u\left[\hat{\theta}(q)\right]}{2}\right)^{1/2}\right).$$

$$\inf_{q\in\Gamma} D^{-1}\left(q, u\left[\hat{\theta}(q)\right]\right) \le \inf_{q\in\Gamma,\ q\le Cu[\hat{\theta}(q)]} D^{-1}\left(q, u\left[\hat{\theta}(q)\right]\right) \le \inf_{q\in\Gamma,\ q\le Cu[\hat{\theta}(q)]} C' u[\hat{\theta}(q)],$$

where C is any constant such that there exists q such that $q \le Cu[\hat{\theta}(q)]$. Here $C' = \max(2C, 8)$.

Note from Theorem 3 that

$$u\left[\hat{\theta}(q)\right] = O\left(\frac{1}{n}\left(\frac{\log n \log|\mathcal{H}|}{\hat{\theta}(q)^2} + \log\frac{1}{\delta}\right)\right),$$

and

$$q = P_{\mathcal{S}}\left(yf(x) < \hat{\theta}(q)\right).$$

Thus we can derive the following three bounds of the generalization error from the Emargin bound by using the three inequalities in Lemma 8 respectively.

Corollary 9 *If* $8 < |\mathcal{H}| < \infty$*, then for any* $\delta > 0$*, with probability at least* $1-\delta$ *over the random choice of the training set* $\mathcal{S}$ *of n examples (*$n > 1$*), every voting classifier* $f \in C(\mathcal{H})$ *such that* $q_0 < 1$ *satisfies the following bounds:*

1.
$$P_{\mathcal{D}}(yf(x) \le 0) \le O\left(\frac{1}{n}\left(\frac{\log n \log|\mathcal{H}|}{\theta_0^2} + \log\frac{1}{\delta}\right)\right).$$

Here we assume $\theta_0 > \sqrt{8/|\mathcal{H}|}$ *is the minimum margin.*

2.
$$P_{\mathcal{D}}\Big(yf(x) \le 0\Big) \le \inf_{\theta \in [\frac{8}{|\mathcal{H}|},1]} \left[P_{\mathcal{S}}\Big(yf(x) \le \theta\Big) + O\left(\frac{1}{\sqrt{n}}\left(\frac{\log n \log|\mathcal{H}|}{\theta^2} + \log\frac{1}{\delta}\right)^{1/2}\right)\right].$$

3. *For any constant C and* $\theta \in [\sqrt{8/|\mathcal{H}|}, 1)$ *such that*
$$P_{\mathcal{S}}(yf(x) \le \theta) \le \frac{C}{n}\left(\frac{8}{\theta^2}\log\left(\frac{2n^2}{\log|\mathcal{H}|}\right)\log(2|\mathcal{H}| + \log|\mathcal{H}| + \log\frac{n}{\delta})\right), \tag{4}$$

we have
$$P_{\mathcal{D}}(yf(x) \le \theta) \le \frac{\log|\mathcal{H}|}{n} + \frac{C'}{n}\left(\frac{8}{\theta^2}\log\left(\frac{2n^2}{\log|\mathcal{H}|}\right)\log(2|\mathcal{H}| + \log|\mathcal{H}| + \log\frac{n}{\delta})\right),$$

where $C' = \max(2C, 8)$.

The first bound in the corollary has the same order as the minimum margin bound. The second bound is essentially the same as Theorem 1 except that θ cannot be too small. So previous bounds can be derived from the Emargin bound. The third bound states that the generalization error is $O\left(\frac{\log n \log|\mathcal{H}|}{n\theta^2}\right)$ even in the non-zero error case, provided the margin error $P_{\mathcal{S}}(yf(x) \le \theta)$ is small enough.

The third bound has a much simpler form than Theorem 1. If we use this bound to define Emargin, that is, the optimal θ in the bound, it can be greatly simplified. It is easy to see that the optimal θ is just the largest θ satisfying (4). The price however is that this approximate bound is not uniformly sharper than the minimum margin bound.

5. Consistency

So far the results are finite sample generalization error bounds. In this section we point out that the Emargin bound and the margin distribution bound in Theorem 1 imply statistical consistency. In particular we show that if a boosting algorithm minimizes the bound, then the classifier learned converges to the optimal classifier in the hypothesis space, that is, the convex hull of the base classifiers. Here we assume that the set of base classifiers $\mathcal{H}$ is symmetric. That is, if $h \in \mathcal{H}$ then $-h \in \mathcal{H}$. Therefore the best classifier in the convex hull of $\mathcal{H}$ is also the best classifier in the linear span of $\mathcal{H}$. An immediate consequence of this consistency is that margin bound optimization is Bayes consistent if the linear span of the base classifiers is dense in the space of all measurable functions. A typical example of such base classifiers is decision tree with the number of leaves larger than the dimension of the input space (Breiman, 2004).

Before stating the consistency theorem, we need some notions. Let $C(\mathcal{H})$ be the convex hull of the set of base classifiers. Also let

$$L^* = \inf_{f \in C(\mathcal{H})} P_{\mathcal{D}}\Big(yf(x) \le 0\Big).$$

That is, L^* is the minimal generalization error of the classifiers in $C(\mathcal{H})$.

We consider an algorithm that optimizes the Emargin: Given a training set $\mathcal{S}$ containing n examples, the learning algorithm returns a function $\hat{f}_n \in C(\mathcal{H})$ which minimizes the finite VC dimension Emargin bound (i.e., the right-hand side of (3)), or simply $D^{-1}\left(q^*, u(\theta^*)\right)$.

The next theorem states that margin bound optimization is consistent. With almost the same arguments one can show that minimizing the margin distribution bound in Theorem 1 is also consistent. But there is no such result for the minimum margin bound for the non-separable problems.

Theorem 10 *Let $C(\mathcal{H})$, L^* and $\hat{f}_n$ be defined as above. Then*

$$\lim_{n\to\infty} EP_{\mathcal{D}}\left(y\hat{f}_n(x) \le 0\right) = L^*,$$

where E is the expectation over the random draw of the training set $\mathcal{S}_n$.

6. Proofs

In this section, we give proofs of the theorems, lemmas and corollaries.

6.1 Proof of Theorem 3

The proof uses the tool developed in Schapire et al. (1998). The difference is that we do not bound the deviation of the generalization error from the empirical margin error directly, instead we consider the difference of the generalization error to a zero-one function of a certain empirical measure. This allows us to unify the zero-error and nonzero-error cases and it results in a sharper bound. For the sake of convenience, we follow the convention in Schapire et al. (1998).

Let $C(\mathcal{H})$ denote the convex hull of $\mathcal{H}$. Also let $C_N(\mathcal{H})$ denote the set of unweighted averages over N elements from the base classifier set $\mathcal{H}$. Formally,

$$C_N(\mathcal{H}) = \left\{ g : \ g = \frac{1}{N}\sum_{j=1}^{N} h_j, \ h_j \in \mathcal{H} \right\}.$$

Any voting classifier

$$f = \sum \beta_i h_i \in C(\mathcal{H}),$$

where

$$\sum \beta_i = 1, \ \ \beta_i \ge 0,$$

can be associated with a distribution over $\mathcal{H}$ by the coefficients $\{\beta_i\}$. We denote this distribution as $Q(f)$. By choosing N elements independently and randomly from $\mathcal{H}$ according to $Q(f)$, we can generate a classifier $g \in C_N(\mathcal{H})$. The distribution of g is denoted by $Q_N(f)$. For any fixed α $(0 < \alpha < 1)$

$$\begin{aligned} P_{\mathcal{D}}\Big(yf(x) \le 0\Big) &\le P_{\mathcal{D},g\sim Q_N(f)}\Big(yg(x) \le \alpha\Big) + P_{\mathcal{D},g\sim Q_N(f)}\Big(yg(x) > \alpha,\, yf(x) \le 0\Big) \\ &\le P_{\mathcal{D},g\sim Q_N(f)}\Big(yg(x) \le \alpha\Big) + \exp\left(-\frac{N\alpha^2}{2}\right). \end{aligned} \tag{5}$$

We next bound the first term on the right-hand side of the inequality. So far the argument is the same as Schapire et al. (1998). From now on we use some different techniques. For any fixed $g \in C_N(\mathcal{H})$, and for any positive number ε and nonnegative integer k such that $k \le n\varepsilon$, we consider the probability (over the random draw of n training examples) that the training error at margin α is less than k/n, while the true error of g at margin α is larger than ε:

$$\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(P_{\mathcal{S}}\left(yg(x) \le \alpha\right) \le \frac{k}{n},\, P_{\mathcal{D}}\left(yg(x) \le \alpha\right) > \varepsilon\right). \tag{6}$$

Here $\Pr_{\mathcal{S}\sim\mathcal{D}^n}$ denotes the probability over n training examples chosen independently at random according to $\mathcal{D}$. Note that the proof in Schapire et al. (1998) considers only the difference of $P_{\mathcal{D}}\left(yg(x) \le \alpha\right)$ and $P_{\mathcal{S}}\left(yg(x) \le \alpha\right)$, that is, $P_{\mathcal{D}}\left(yg(x) \le \alpha\right) - P_{\mathcal{S}}\left(yg(x) \le \alpha\right)$; While here we consider the values of $P_{\mathcal{D}}\left(yg(x) \le \alpha\right)$ and $P_{\mathcal{S}}\left(yg(x) \le \alpha\right)$ themselves. The benefit is that this allows us to use the tightest version of Chernoff bound—the relative entropy Chernoff bound—rather than the relatively looser additive Chernoff bound. To derive the bound, we write (6) in the following equivalent form.

$$\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(P_{\mathcal{D}}(yg(x) \le \alpha) > I\left[P_{\mathcal{S}}(yg(x) \le \alpha) > \frac{k}{n}\right] + \varepsilon\right), \tag{7}$$

where I is the indicator function. (7) is important in our proof. It bounds the difference of the true and empirical margin distributions as α and k vary over their ranges. But k and α can take essentially finite number of values, so we can use union bounds. It's easy to see that no matter $P_{\mathcal{D}}(yg(x) \le \alpha) > \varepsilon$ or $P_{\mathcal{D}}(yg(x) \le \alpha) \le \varepsilon$, we have the following inequality (In the former case, it is the tail bound for Bernoulli trials; and in the latter case the probability is actually zero).

$$\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(P_{\mathcal{D}}\Big(yg(x) \le \alpha\Big) > I\left[P_{\mathcal{S}}\Big(yg(x) \le \alpha\Big) > \frac{k}{n}\right] + \varepsilon\right) \le \quad \sum_{r=0}^{k}\binom{n}{r}\varepsilon^r(1-\varepsilon)^{n-r}.$$

Then applying the relative entropy Chernoff bound (Hoeffding, 1963) to the Bernoulli trials, we further have

$$\sum_{r=0}^{k}\binom{n}{r}\varepsilon^r(1-\varepsilon)^{n-r} \le \exp\left(-nD\left(\frac{k}{n}\Big\|\varepsilon\right)\right).$$

We thus obtain

$$\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(P_{\mathcal{D}}\Big(yg(x) \le \alpha\Big) > I\left[P_{\mathcal{S}}\Big(yg(x) \le \alpha\Big) > \frac{k}{n}\right] + \varepsilon\right) \le \exp\left(-nD\left(\frac{k}{n}\Big\|\varepsilon\right)\right). \tag{8}$$

We only consider α at the values in the set

$$U = \left\{\frac{1}{|\mathcal{H}|}, \frac{2}{|\mathcal{H}|}, \ldots, 1\right\}.$$

There are no more than $|\mathcal{H}|^N$ elements in $C_N(\mathcal{H})$. Using the union bound we get

$$\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(\exists g\in C_N(\mathcal{H}),\ \exists\alpha\in U,\ P_{\mathcal{D}}\Big(yg(x)\le\alpha\Big)>I\left[P_{\mathcal{S}}\Big(yg(x)\le\alpha\Big)>\frac{k}{n}\right]+\varepsilon\right)$$
$$\le|\mathcal{H}|^{(N+1)}\exp\left(-nD\left(\frac{k}{n}\Big\|\varepsilon\right)\right).$$

The above formula upper bounds the probability that "$\exists g\in C_N(\mathcal{H})$" certain inequality of g holds. The bound also applies to "$\exists$ a distribution of g over $C_N(\mathcal{H})$" such that the inequality of the *expectation* over g holds, since the latter implies the former. Note that

$$E_{g\sim Q_N(f)}P_{\mathcal{D}}\Big(yg(x)\le\alpha\Big)=P_{\mathcal{D},g\sim Q_N(f)}\Big(yg(x)\le\alpha\Big),$$
$$E_{g\sim Q_N(f)}I\left[P_{\mathcal{S}}\Big(yg(x)\le\alpha\Big)>\frac{k}{n}\right]=P_{g\sim Q_N(f)}\left(P_{\mathcal{S}}\Big(yg(x)\le\alpha\Big)>\frac{k}{n}\right).$$

We thus have

$$\Pr_{S\sim D^n}\left(\exists f\in C(\mathcal{H}),\exists\alpha\in U,P_{D,g\sim Q_N(f)}\Big(yg(x)\le\alpha\Big)>P_{g\sim Q_N(f)}\left(P_{\mathcal{S}}(yg(x)\le\alpha)>\frac{k}{n}\right)+\varepsilon\right)$$
$$\le|\mathcal{H}|^{(N+1)}\exp\left(-nD\left(\frac{k}{n}\Big\|\varepsilon\right)\right).$$

Let
$$\delta=|\mathcal{H}|^{(N+1)}\exp\left(-nD\left(\frac{k}{n}\Big\|\varepsilon\right)\right),$$
then
$$\varepsilon=D^{-1}\left(\frac{k}{n},\frac{1}{n}\left[(N+1)\log|\mathcal{H}|+\log\frac{1}{\delta}\right]\right).$$

We obtain that with probability at least $1-\delta$ over the draw of the training examples, for all $f\in C(\mathcal{H})$, all $\alpha\in U$, but fixed k,

$$P_{\mathcal{D},g\sim Q_N(f)}\Big(yg(x)\le\alpha\Big)\le P_{g\sim Q_N(f)}\left(P_{\mathcal{S}}\Big(yg(x)\le\alpha\Big)>\frac{k}{n}\right)+D^{-1}\left(\frac{k}{n},\frac{1}{n}\left[(N+1)\log|\mathcal{H}|+\log\frac{1}{\delta}\right]\right).\tag{9}$$

We next bound the first term in the right-hand side of (9). Using the same argument for deriving (5), we have for any fixed $f,\mathcal{S},\alpha,k$, any $\theta>\alpha$

$$P_{g\sim Q_N(f)}\left(P_{\mathcal{S}}\Big(yg(x)\le\alpha\Big)>\frac{k}{n}\right)\le I\left[P_{\mathcal{S}}\Big(yf(x)<\theta\Big)>\frac{k}{n}\right]+P_{g\sim Q_N(f)}\left(P_{\mathcal{S}}\Big(yg(x)\le\alpha\Big)>\frac{k}{n},P_{\mathcal{S}}\Big(yf(x)<\theta\Big)\le\frac{k}{n}\right).\tag{10}$$

Note that the last term in (10) can be written in the following equivalent form and further bounded by

$$P_{g\sim Q_N(f)}\left(\exists(x_i,y_i)\in\mathcal{S}:\ y_ig(x_i)\le\alpha,\ \ y_if(x_i)\ge\theta\right)\le n\exp\left(-\frac{N(\theta-\alpha)^2}{2}\right). \tag{11}$$

Combining (5), (9), (10) and (11), we have that with probability at least $1-\delta$ over the draw of training examples, for all $f\in C(\mathcal{H})$, all $\alpha\in U$, all $\theta>\alpha$, but fixed k and N

$$\begin{aligned}P_{\mathcal{D}}\Big(yf(x)\le 0\Big)\le&\exp\left(-\frac{N\alpha^2}{2}\right)+n\exp\left(-\frac{N(\theta-\alpha)^2}{2}\right)\\&+I\left[P_{\mathcal{S}}\Big(yf(x)<\theta\Big)>\frac{k}{n}\right]+D^{-1}\left(\frac{k}{n},\frac{1}{n}\left[(N+1)\log|\mathcal{H}|+\log\frac{1}{\delta}\right]\right).\end{aligned}$$

Since θ is arbitrary, we set $\theta=\hat{\theta}(\frac{k}{n})$. Now we construct α by rounding $\theta/2$ to the nearest neighbor of $1/|\mathcal{H}|$. Let

$$\alpha=\frac{\theta}{2}-\frac{\eta}{|\mathcal{H}|}\in U,$$

where $0\le\eta<1$. The goal is to let α takes only a finite number of values. (Recall that $U=\{\frac{1}{|\mathcal{H}|},\cdots,1\}$.) It is easy to check that the sum of the first two terms on the right-hand side of the above inequality can be bounded by the following.

$$\begin{aligned}&\exp\left(-\frac{N\alpha^2}{2}\right)+n\exp\left(-\frac{N(\theta-\alpha)^2}{2}\right)\\ \le\ &\exp\left(-\frac{N\theta^2}{8}\right)\exp\left(-\frac{N\eta^2}{2|\mathcal{H}|^2}\right)\left[\exp\left(\frac{N\theta\eta}{2|\mathcal{H}|}\right)+n\exp\left(-\frac{N\theta\eta}{2|\mathcal{H}|}\right)\right]\\ \le\ &\max\left(2n,\exp\left(\frac{N}{2|\mathcal{H}|}\right)+1\right)\exp\left(-\frac{N\theta^2}{8}\right).\end{aligned}$$

The last inequality holds since $0\le\theta,\eta\le 1$. Replacing δ by $\delta\cdot 2^{-N}$. we can get a union bound over all N by replacing $\log(\frac{n}{\delta})$ in all previous equations by $\log(\frac{n}{\delta\cdot 2^{-N}})=N\log 2+\log(\frac{n}{\delta})$. Put

$$N=\left\lceil\frac{8}{\theta^2}\log\left(\frac{2n^2}{\log|\mathcal{H}|}\right)\right\rceil.$$

Now for any sample $\mathcal{S}$ we only consider $f\in C(\mathcal{H})$ and k that satisfy $q_0<1$ and

$$\frac{k}{n}\ge q_0. \tag{12}$$

Note that by (12) and the assumption that $|\mathcal{H}|>8$, we have

$$\theta>\sqrt{\frac{8}{|\mathcal{H}|}}.$$

So by some numerical calculations one can show

$$2n>\exp\left(\frac{N}{2|\mathcal{H}|}\right)+1,\quad (n>1).$$

Recall that $\theta = \hat{\theta}(k/n)$, so $P_{\mathcal{S}}\Big(yf(x) < \theta\Big) \le \frac{k}{n}$. We thus obtain that for fixed k, with probability at least $1-\delta$ over the random choice of the training set $\mathcal{S}$ of n examples, every $f \in C(\mathcal{H})$ with $q_0 < 1$ satisfies

$$P_{\mathcal{D}}\Big(yf(x) \le 0\Big) \le \frac{\log|\mathcal{H}|}{n} + D^{-1}\left(\frac{k}{n}, u\right),$$

where

$$u = \frac{1}{n}\left(\frac{8}{\theta^2}\log\left(\frac{2n^2}{\log|\mathcal{H}|}\right)\log(2|\mathcal{H}|) + 2\log|\mathcal{H}| + \log\frac{1}{\delta}\right).$$

Finally using the union bound over $k \in \{nq_0, \ldots, n-1\}$ and replacing δ by δ/n, we have with probability at least $1-\delta$ over the random choice of the training set $\mathcal{S}$ of n examples, every $f \in C(\mathcal{H})$ with $q_0 < 1$ satisfies

$$P_{\mathcal{D}}\Big(yf(x) \le 0\Big) \le \frac{\log|\mathcal{H}|}{n} + \inf_{k \in \{nq_0,\ldots,n-1\}} D^{-1}\left(\frac{k}{n}, u'\right),$$

where

$$u' = \frac{1}{n}\left(\frac{8}{\theta^2}\log\left(\frac{2n^2}{\log|\mathcal{H}|}\right)\log(2|\mathcal{H}|) + 2\log|\mathcal{H}| + \log\frac{n}{\delta}\right).$$

The theorem follows. ■

6.2 Proof of Theorem 5

The following lemma will be used to prove Theorem 5.

Lemma 11 $D^{-1}(0, p) \le p$ *for* $p \ge 0$.

Proof of Lemma 11. We only need to show

$$D(0||p) \ge p,$$

since $D(q||p)$ is a monotonic increasing function of p for $p \ge q$. By Taylor expansion

$$D(0||p) = -\log(1-p) = p + \frac{p^2}{2} + \frac{p^3}{3} + \cdots \ge p.$$

■

Proof of Theorem 5. By the assumption of Theorem 2 we have $\theta_0 > 4\sqrt{\frac{2}{|\mathcal{H}|}}$. Then it is easy to see that the right-hand side of the Emargin bound (1) is the minimum over all $q \in \left\{0, \ldots, \frac{n-1}{n}\right\}$. Take $q = 0$, it is clear that $\hat{\theta}(0)$ is the minimum margin. By Lemma 11, the Emargin bound can be relaxed to

$$\begin{aligned} P_{\mathcal{D}}\Big(yf(x) \le 0\Big) &\le \frac{1}{n}\left(\frac{8}{\theta_0^2}\log\left(\frac{2n^2}{\log|\mathcal{H}|}\right)\log(2|\mathcal{H}|) + 3\log|\mathcal{H}| + \log\frac{n}{\delta}\right) \\ &\le \frac{16\log(2n)\log(2|\mathcal{H}|)}{n\theta_0^2} + \frac{\log n + 2\log|\mathcal{H}|}{n} + \frac{1}{n}\log\left(\frac{|\mathcal{H}|}{\delta}\right). \qquad (13) \end{aligned}$$

We only need to show that this relaxed bound is sharper than Theorem 2. For the minimum margin bound, we only consider the case that $R \le 1$, since otherwise the bound is larger than one. Simple calculations show that the right-hand side of (13) is smaller than the minimum margin bound. ■

6.3 Proof of Theorem 6

Remember that $q_i = P_{\mathcal{S}}(yf_i(x) < \theta_i)$ is the optimal q^* in the Emargin bound. Thus we only need to show

$$D^{-1}\Big(q_1, u(\theta_1)\Big) \le D^{-1}\Big(q_2, u(\theta_2)\Big).$$

Note that if $\theta_1 \ge \theta_2$, then $u(\theta_1) \le u(\theta_2)$. So

$$D^{-1}\Big(q_2, u(\theta_2)\Big) \ge D^{-1}\Big(q_2, u(\theta_1)\Big),$$

since $D^{-1}(q,u)$ is an increasing function of u for fixed q. Also $D^{-1}(q,u)$ is an increasing function of q for fixed u, we have

$$D^{-1}\Big(q_2, u(\theta_1)\Big) \ge D^{-1}\Big(q_1, u(\theta_1)\Big)$$

since $q_1 \le q_2$. This completes the proof. ■

6.4 Proof of Theorem 7

The next lemma is a modified version of the uniform convergence result (Vapnik and Chervonenkis, 1971; Vapnik, 1998) and its refinement (Devroye, 1982). It will be used for proving Theorem 7.

Lemma 12 *Let $\mathcal{A}$ be a class of subsets of a space Z. Let $z_i \in Z$, $i = 1, \ldots, n$. Let $N^{\mathcal{A}}(z_1, z_2, \ldots, z_n)$ be the number of different sets in*

$$\Big\{\{z_1, z_2, \ldots, z_n\} \bigcap A : \; A \in \mathcal{A}\Big\}.$$

Define

$$s(\mathcal{A}, n) = \max_{(z_1, z_2, \ldots, z_n) \in Z^n} N^{\mathcal{A}}(z_1, z_2, \ldots, z_n).$$

Assume $\varepsilon \ge \frac{1}{n}$. Let $\varepsilon' = \frac{n}{n-1}\varepsilon - \frac{1}{n}$. Then for any distribution $\mathcal{D}$ over Z and any nonnegative integer k such that $\frac{k}{n} \le \varepsilon'$

$$\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(\exists A \in \mathcal{A} : \; P_{\mathcal{D}}(A) > I\left[P_{\mathcal{S}}(A) > \frac{k}{n}\right] + \varepsilon\right) \le 2 \cdot s(\mathcal{A}, n^2) \exp\left(-nD\left(\frac{k}{n} \Big\| \varepsilon'\right)\right).$$

Proof of Lemma 12. The proof is the standard argument. We first show that for any $0 < \alpha < 1$, $\varepsilon > 0$, and any integer n'

$$\begin{aligned}
&\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(\exists A \in \mathcal{A} : \; P_{\mathcal{D}}(A) > I\left[P_{\mathcal{S}}(A) > \frac{k}{n}\right] + \varepsilon\right) \\
&\qquad \le \left(\frac{1}{1 - e^{-2n'\alpha^2\varepsilon^2}}\right) \Pr_{\mathcal{S}\sim\mathcal{D}^n,\, \mathcal{S}'\sim\mathcal{D}^{n'}}\left(\exists A \in \mathcal{A} : \; P_{\mathcal{S}'}(A) > I\left[P_{\mathcal{S}}(A) > \frac{k}{n}\right] + (1-\alpha)\varepsilon\right).
\end{aligned}$$

Or equivalently,

$$\begin{aligned}
&\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(\sup_{A\in\mathcal{A}}\left(P_{\mathcal{D}}(A) - I\left[P_{\mathcal{S}}(A) > \frac{k}{n}\right]\right) > \varepsilon\right) \\
&\qquad \le \left(\frac{1}{1 - e^{-2n'\alpha^2\varepsilon^2}}\right) \Pr_{\mathcal{S}\sim\mathcal{D}^n,\, \mathcal{S}'\sim\mathcal{D}^{n'}}\left(\sup_{A\in\mathcal{A}}\left(P_{\mathcal{S}'}(A) - I\left[P_{\mathcal{S}}(A) > \frac{k}{n}\right]\right) > (1-\alpha)\varepsilon\right). \qquad (14)
\end{aligned}$$

Let V denote the event

$$\sup_{A\in\mathcal{A}}\left(P_{\mathcal{D}}(A)-I\left[P_{\mathcal{S}}(A)>\frac{k}{n}\right]\right)>\varepsilon.$$

If the above event occurs, let A^* be any $A\in\mathcal{A}$ so that $P_{\mathcal{D}}(A)-I\left[P_{\mathcal{S}}(A)>\frac{k}{n}\right]>\varepsilon$. Otherwise let A^* be any $A\in\mathcal{A}$. Note that the following two events

$$P_{\mathcal{S}'}(A^*)\geq P_{\mathcal{D}}(A^*)-\alpha\varepsilon$$

and

$$P_{\mathcal{D}}(A^*)-I\left[P_{\mathcal{S}}(A^*)>\frac{k}{n}\right]>\varepsilon$$

imply that

$$P_{\mathcal{S}'}(A^*)-I\left[P_{\mathcal{S}}(A^*)>\frac{k}{n}\right]>(1-\alpha)\varepsilon.$$

Then

$$\begin{aligned}
&\Pr_{\mathcal{S}\sim\mathcal{D}^n,\ \mathcal{S}'\sim\mathcal{D}^{n'}}\left(\sup_{A\in\mathcal{A}}\left(P_{\mathcal{S}'}(A)-I\left[P_{\mathcal{S}}(A)>\frac{k}{n}\right]\right)>(1-\alpha)\varepsilon\right)\\
&=\int d\mathcal{D}^n\int I\left[\sup_{A\in\mathcal{A}}\left(P_{\mathcal{S}'}(A)-I\left[P_{\mathcal{S}}(A)>\frac{k}{n}\right]\right)>(1-\alpha)\varepsilon\right]dD^{n'}\\
&\geq\int_V d\mathcal{D}^n\int I\left[\sup_{A\in\mathcal{A}}\left(P_{\mathcal{S}'}(A)-I\left[P_{\mathcal{S}}(A)>\frac{k}{n}\right]\right)>(1-\alpha)\varepsilon\right]d\mathcal{D}^{n'}\\
&\geq\int_V d\mathcal{D}^n\int I\left[P_{\mathcal{S}'}(A^*)-I\left[P_{\mathcal{S}}(A^*)>\frac{k}{n}\right]>(1-\alpha)\varepsilon\right]d\mathcal{D}^{n'}\\
&\geq\int_V d\mathcal{D}^n\int I\left[P_{\mathcal{S}'}(A^*)\geq P_{\mathcal{D}}(A^*)-\alpha\varepsilon\right]d\mathcal{D}^{n'}\\
&\geq\left(1-e^{-2n'\alpha^2\varepsilon^2}\right)\int_V d\mathcal{D}^n\\
&=\left(1-e^{-2n'\alpha^2\varepsilon^2}\right)\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(\sup_{A\in\mathcal{A}}\left(P_{\mathcal{D}}(A)-I\left[P_{\mathcal{S}}(A)>\frac{k}{n}\right]\right)>\varepsilon\right).
\end{aligned}$$

This completes the proof of (14).

Take

$$n'=n^2-n,$$
$$\alpha=\frac{1}{(n-1)\varepsilon},$$

we have

$$\begin{aligned}
&\Pr_{\mathcal{S}\sim\mathcal{D}^n}\left(\exists A\in\mathcal{A}:\ P_{\mathcal{D}}(A)>I\left[P_{\mathcal{S}}(A)>\frac{k}{n}\right]+\varepsilon\right)\\
&\quad\leq 2\Pr_{\mathcal{S}\sim\mathcal{D}^n,\ \mathcal{S}'\sim\mathcal{D}^{n'}}\left(\exists A\in\mathcal{A}:\ P_{\mathcal{S}'}(A)>I\left[P_{\mathcal{S}}(A)>\frac{k}{n}\right]+(\varepsilon-\frac{1}{n-1})\right).
\end{aligned}$$

Proceeding as Devroye (1982) and using the relative entropy Hoeffding inequality, the lemma follows. ■

Proof of Theorem 7. The proof is the same as Theorem 3 until we have (8). Let $\alpha = \frac{\theta}{2}$, we need to bound

$$\Pr_{S\sim\mathcal{D}^n}\left(\exists g \in C_N(\mathcal{H}),\ \exists\theta > 0,\ P_{\mathcal{D}}\Big(yg(x) \leq \frac{\theta}{2}\Big) > I\left[P_S\Big(yg(x) \leq \frac{\theta}{2}\Big) > \frac{k}{n}\right] + \varepsilon\right).$$

Note that for fixed N, in order to derive a bound uniformly over all $0 < \theta \leq 1$ it suffices to show the bound holds for $\theta = \frac{1}{N}, \frac{2}{N}, \ldots, 1$. Let

$$A(g) = \left\{(x,y) \in \mathcal{X} \times \{-1,1\} :\ yg(x) \leq \frac{\theta}{2}\right\},$$

and

$$\mathcal{A} = \left\{A(g) :\ g \in C_N(\mathcal{H})\right\}.$$

By Sauer's lemma (Sauer, 1972) it is easy to see that

$$s(\mathcal{A}, n) \leq \left(\frac{en}{d}\right)^{Nd},$$

where d is the VC dimension of $\mathcal{H}$. By Lemma 12, we have

$$\begin{aligned}&\Pr_{S\sim D^n}\left(\exists g \in C_N(\mathcal{H}),\ \exists\theta > 0,\ P_{\mathcal{D}}\Big(yg(x) \leq \frac{\theta}{2}\Big) > I\left[P_S\Big(yg(x) \leq \frac{\theta}{2}\Big) > \frac{k}{n}\right] + \varepsilon\right)\\ &\leq 2(N+1)\left(\frac{en^2}{d}\right)^{Nd}\exp\left(-nD\left(\frac{k}{n}\Big\|\varepsilon'\right)\right),\end{aligned}$$

where

$$\varepsilon' = \frac{n}{n-1}\varepsilon - \frac{1}{n}.$$

Proceeding as the proof of Theorem 3, we have that with probability at least $1-\delta$ the following holds for every $f \in C(\mathcal{H})$, every $\theta > 0$ but fixed k, where $0 \leq k \leq n\varepsilon$.

$$P_{\mathcal{D}, g\sim Q_N(f)}\left(yg(x) \leq \frac{\theta}{2}\right) \leq P_{g\sim Q_N(f)}\left(P_S\left(yg(x) \leq \frac{\theta}{2}\right) > \frac{k}{n}\right) + \frac{1}{n} + \frac{n-1}{n}D^{-1}\left(\frac{k}{n}, \tau\right), \tag{15}$$

where

$$\tau = \frac{1}{n}\left[Nd\left(\log\frac{n^2}{d} + 1\right) + \log(2(N+1)) + \log\frac{1}{\delta}\right].$$

Similar to the proof of Theorem 3, we can bound the first term of (15) as

$$\begin{aligned}P_{g\sim Q_N(f)}\left(P_S\left(yg(x) \leq \frac{\theta}{2}\right) > \frac{k}{n}\right) &\leq I\left[P_S\left(yf(x) < \theta\right) > \frac{k}{n}\right]\\ &\quad + P_{g\sim Q_N(f)}\left(P_S\left(yg(x) \leq \frac{\theta}{2}\right) > \frac{k}{n},\ P_S\left(yf(x) < \theta\right) \leq \frac{k}{n}\right)\\ &\leq I\left[P_S\left(yf(x) < \theta\right) > \frac{k}{n}\right] + n\exp\left(-\frac{N\theta^2}{8}\right).\end{aligned} \tag{16}$$

Setting $\theta = \hat{\theta}(\frac{k}{n})$ and combining (15), (16) and (5); recalling $\alpha = \theta/2$ we have with probability at least $1-\delta$ for all $f \in C(\mathcal{H})$, all $0 < \theta \leq 1$, but fixed k and N

$$P_{\mathcal{D}}(yf(x) \leq 0) \leq \frac{1}{n} + (n+1)\exp\left(-\frac{N\theta^2}{8}\right) + \frac{n-1}{n} D^{-1}\left(\frac{k}{n}, \tau\right).$$

Use the union bound over N; put $N = \frac{16}{\theta^2}\log\frac{n}{d}$ and use the union bound over k as in the proof of Theorem 3 we obtain the theorem. ■

6.5 Proof of Lemma 8

The first inequality has already been proved in Lemma 11.

For the second inequality, we only need to show

$$D^{-1}(q,u) \leq q + \sqrt{u/2},$$

or equivalently

$$D(q, q+\sqrt{u/2}) \geq u,$$

since D is an increasing function in the second parameter. But this is immediate by a well known result (Hoeffding, 1963):

$$D(q, q+\delta) \geq 2\delta^2.$$

For the third inequality we first show that for all $0 < q < 1$

$$D^{-1}\left(\frac{q}{2}, \frac{q}{8}\right) \leq q, \tag{17}$$

which is equivalent to

$$D\left(\frac{q}{2}\middle\|q\right) \geq \frac{q}{8}.$$

For fixed q, let $\phi(x) = D(qx||q), 0 < x \leq 1$. Note that

$$\phi(1) = \phi'(1) = 0,$$

and

$$\phi''(x) = \frac{q}{x(1-qx)} \geq q,$$

we have

$$D\left(\frac{q}{2}\middle\|q\right) = \phi\left(\frac{1}{2}\right) \geq \frac{q}{8}.$$

This completes the proof of (17).

Now if $q \leq Cu[\hat{\theta}(q)]$, recall that $C' = \max(2C, 8)$, and note D^{-1} is increasing function on its first and second parameter respectively. If $C'u[\hat{\theta}(q)] < 1$ we have

$$\begin{aligned} D^{-1}\left(q, u\left[\hat{\theta}(q)\right]\right) &\leq D^{-1}\left(\frac{C'}{2}u\left[\hat{\theta}(q)\right], u\left[\hat{\theta}(q)\right]\right) \\ &\leq D^{-1}\left(\frac{C'}{2}u\left[\hat{\theta}(q)\right], \frac{C'}{8}u\left[\hat{\theta}(q)\right]\right) \\ &\leq C'u\left[\hat{\theta}(q)\right]. \end{aligned}$$

The lemma follows. ■

6.6 Proof of Corollary 9

The first and third bounds are straightforward from lemma 8. We only prove the second bound.

Let $\Phi(\theta)$ be the right hand side of the bound (without taking the infimum) we want to prove, that is,

$$\Phi(\theta) = P_{\mathcal{S}}\Big(yf(x) \le \theta\Big) + O\left(\frac{1}{\sqrt{n}}\left(\frac{\log n \log|\mathcal{H}|}{\theta^2} + \log\frac{1}{\delta}\right)^{1/2}\right).$$

It is not difficult to see that there is no θ that can achieve $\inf_{\theta\in[8/|\mathcal{H}|,1]}\Phi(\theta)$. To see this, first note that for any θ, either $P_{\mathcal{S}}(yf(x) < \theta) = P_{\mathcal{S}}(yf(x) \le \theta)$ (a continuous point), or $P_{\mathcal{S}}(yf(x) < \theta) < P_{\mathcal{S}}(yf(x) \le \theta)$ (a jump point). In the former case, increasing θ decreases $\Phi(\theta)$ since $P_{\mathcal{S}}(yf(x) \le \theta)$ does not change but $u(\theta)$ is decreasing. In the latter case, decreasing θ also decreases $\Phi(\theta)$, since $P_{\mathcal{S}}(yf(x) \le \theta)$ decreases discontinuously while $u(\theta)$ increases continuously.

Let $\theta_1, \theta_2, \ldots$, be a sequence so that $\Phi(\theta_i)$ converges to $\inf_\theta \Phi(\theta)$. Let $\bar{\theta}$ be the limiting point of $\theta_1, \theta_2, \ldots$. It is not difficult to see from the above argument that for sufficiently large i, $\theta_i < \bar{\theta}$, since there is a jump of $\Phi(\theta)$ at those θ such that $P_{\mathcal{S}}(yf(x) \le \theta)$ is discontinuous. Take any θ_i that is sufficiently close to $\bar{\theta}$. Let $q_i = P_{\mathcal{S}}(yf(x) \le \theta_i)$, we must have $\hat{\theta}(q_i) = \bar{\theta}$ (recall that $\hat{\theta}(q_i) = \sup\{\theta \in (0,1] : P_{\mathcal{S}}(yf(x) \le \theta) \le q_i\}$). Therefore $u[\hat{\theta}(q_i)] < u(\theta_i)$ and hence $q_i + (u[\hat{\theta}(q_i)])^{1/2} < \Phi(\theta_i)$. Thus $\inf_q(q + (u[\hat{\theta}(q)])^{1/2}) \le \inf_\theta \Phi(\theta)$. The corollary follows. ■

6.7 Proof of Theorem 10

We first give a simple lemma.

Lemma 13 *Let ξ be a random variable and κ a positive constant. If for any $t > 0$ we have $P(\xi > \kappa t) < \exp(-t^2)$, then $E\xi \le \frac{\sqrt{\pi}}{2}\kappa$.*

Proof of Lemma 13.

$$E\xi = \int_{-\infty}^{\infty} u\, d(-P(\xi > u)) \le \int_0^{\infty} u\, d(-P(\xi > u)).$$

By the assumption, we have

$$E\xi \le \int_0^{\infty} \kappa t\, d(-e^{-t^2}) = \frac{\sqrt{\pi}}{2}\kappa.$$

■

Proof of Theorem 10.

Let $B(f)$ be the right-hand-side of the Emargin bound in Theorem 7. Then for any training set $\mathcal{S}$, $\hat{f}_n$ is the function f in $C(\mathcal{H})$ so that $B(f)$ is minimized, that is, $\hat{f}_n = \arg\min_{f\in C(\mathcal{H})} B(f)$. According to the Emargin bound, with probability $1-\delta$

$$P_{\mathcal{D}}(y\hat{f}_n(x) \le 0) \le B(\hat{f}_n).$$

Since $\hat{f}_n = \arg\min_{f\in C(\mathcal{H})} B(f)$, then for any $f \in C(\mathcal{H})$, we have $B(\hat{f}_n) \le B(f)$. Therefore for all $f \in C(\mathcal{H})$, with probability $1-\delta$

$$P_{\mathcal{D}}(y\hat{f}_n(x) \le 0) \le B(f) = \frac{d^2+1}{n} + \inf_{q\in\{q_0,\ldots,\frac{n-1}{n}\}} \frac{n-1}{n} D^{-1}(q, u[\hat{\theta}(q)]).$$

For any fixed $f \in C(\mathcal{H})$, let $q = P_{\mathcal{S}}(yf(x) \le n^{-1/4})$. It is easy to see that $\hat{\theta}(q) \ge n^{-1/4}$ and $u[\hat{\theta}(q)] \le u[n^{-1/4}]$, where $u[\hat{\theta}(q)]$ is defined in Theorem 7. By the second inequality of D^{-1} in lemma 8, we have

$$\begin{aligned} P_{\mathcal{D}}(y\hat{f}_n(x) \le 0) &\le \frac{d^2+1}{n} + \frac{n-1}{n}\left(q + (u\left[\hat{\theta}(q)\right])^{1/2}\right), \\ &\le \frac{d^2+1}{n} + \frac{n-1}{n}\left(P_{\mathcal{S}}(yf(x) \le n^{-1/4}) + (u\left[n^{-1/4}\right])^{1/2}\right). \end{aligned}$$

It is easy to see that there is a constant c (independent of f) such that the right-hand-side of the above inequality can be further bounded by

$$\frac{n-1}{n} P_{\mathcal{S}}(yf(x) \le n^{-1/4}) + c\frac{d \log \frac{n}{d}}{n^{1/4}} + c\sqrt{\frac{\log n}{n} \log(\frac{1}{\delta})}.$$

Let $t = \sqrt{\log(\frac{1}{\delta})}$, we have that for any $t > 0$ with probability at most $\exp(-t^2)$

$$P_{\mathcal{D}}(y\hat{f}_n(x) \le 0) - \frac{n-1}{n} P_{\mathcal{S}}(yf(x) \le n^{-1/4}) - c\frac{d \log \frac{n}{d}}{n^{1/4}} > c\sqrt{\frac{\log n}{n}}t.$$

According to lemma 13, we obtain

$$EP_{\mathcal{D}}(y\hat{f}_n(x) \le 0) - \frac{n-1}{n} EP_{\mathcal{S}}(yf(x) \le n^{-1/4}) - c\frac{d \log \frac{n}{d}}{n^{1/4}} \le \frac{c\sqrt{\pi}}{2}\sqrt{\frac{\log n}{n}},$$

where the expectation is over the random choice of the training set. Note that

$$EP_{\mathcal{S}}(yf(x) \le n^{-1/4}) = P_{\mathcal{D}}(yf(x) \le n^{-1/4}),$$

we have

$$EP_{\mathcal{D}}(y\hat{f}_n(x) \le 0) \le \frac{n-1}{n} P_{\mathcal{D}}(yf(x) \le n^{-1/4}) + c\frac{d \log \frac{n}{d}}{n^{1/4}} + \frac{c\sqrt{\pi}}{2}\sqrt{\frac{\log n}{n}}.$$

Let $n \to \infty$, we obtain

$$\lim_{n\to\infty} EP_{\mathcal{D}}(y\hat{f}_n(x) \le 0) \le \lim_{n\to\infty} P_{\mathcal{D}}(yf(x) \le n^{-1/4}) = P_{\mathcal{D}}(yf(x) \le 0).$$

The last equality holds because $P_{\mathcal{D}}(yf(x) \le \theta)$ is a right continuous function of θ. Since the above inequality is true for every $f \in C(\mathcal{H})$, we have

$$\lim_{n\to\infty} EP_{\mathcal{D}}(y\hat{f}_n(x) \le 0) \le \inf_{f \in C(\mathcal{H})} P_{\mathcal{D}}(yf(x) \le 0) = L^*.$$

■

7. Experiments

In this section we provide experimental results to verify our theory. We compare AdaBoost and LP-AdaBoost in terms of their Emargin, Emargin error and the generalization error. Theorem 6 suggests that if a voting classifier f_1 has a larger Emargin and a smaller Emargin error than another classifier f_2, then f_1 has a smaller bound of the generalization error than f_2. Thus we expect f_1 will

Data Set	# Examples	# Features	Data Set	# Examples	# Features
Image	2310	16	Page-block	5473	10
Isolet	7797	617	Pendigits	10992	16
Letter	20000	16	Satimage	6435	36
Magic04	19022	10	Shuttle	58000	9
Mfeat-fac	2000	216	Spambase	4601	57
Optdigits	5620	64	Waveform	5000	30

Table 1: Description of the large data sets

Data Set	# Examples	# Features
Breast	683	9
Diabetes	768	8
German	1000	24
Vehicle	845	18
Wdbc	569	30

Table 2: Description of the small data sets

have better performance on the test data. The goal of the experiment is to see whether the empirical results agree with the theoretical prediction.

The experiments are conducted on 17 benchmark data sets all from the UCI repository (Asuncion and Newman, 2007). The data sets are grouped into two categories. Table 1 lists 12 "large" data sets, each containing at least 1000 data points. Table 7 lists 5 "small" data sets, each has at most 1000 examples. (We distinguish large and small data sets because we found they demonstrate somewhat different results, see below for discussions.) If the data is multiclass, we group them into two classes since we study binary classification problems. For instance, the "letter" data set has 26 classes, we use the first 13 as the positive and the others as the negative. In the preprocessing stage, each feature is normalized to $[0,1]$. For all data sets we use 5-fold cross validation, and average the results over 10 runs (for a total of 50 runs on each data set).

In order to study the effect of the margins, we need to control and calculate the complexity of the base classifiers. We conduct two sets of experiments using different base classifiers. For one set of experiments, we use decision stumps. For the other, we use three-layer eight-leaf (complete) binary decision trees (Therefore the shape of the trees are fixed). We consider a finite set of base classifiers. Specifically, for each feature we consider 100 thresholds uniformly distributed on $[0,1]$. Therefore the size of the set of decision stumps is $2\times 100\times k$, and for the three-layer eight-leaf trees is $(2\times 100\times k)^7$, where k denotes the number of features.

We run AdaBoost 100 rounds, and use the obtained base classifiers to train the LP-AdaBoost voting classifier. We then calculate the Emargin, Emargin error, test error as well as the minimum margin of them respectively. The calculation of the Emargin involves solving the inverse relative entropy $D^{-1}(q,u)$. Since D is a monotone function on the second parameter, one can adopt the Newton method to find the root of $D(q||\cdot)-u=0$ on $[q,1]$. Another simple way to solve $D^{-1}(q,u)$ is just applying binary search on $[q,1]$: Let $p_1=q, p_2=1$. We have $D(q,p_1)=0\le u$ and $D(q,p_2)=\infty>u$. Then let $p_3=\frac{p_1+p_2}{2}$, compute $D(q,p_3)$ and see if $D(q,p_3)>u$ or not, etc.

		Emargin	Emargin Error	Test Error	Min margin
Image	Ada	0.461 ± 0.024	0.799 ± 0.016	0.032 ± 0.009	-0.076 ± 0.010
	LP	**0.751 ± 0.238**	**0.664 ± 0.075**	**0.029 ± 0.009**	0.000 ± 0.001
Isolet	Ada	0.172 ± 0.057	**0.714 ± 0.040**	**0.163 ± 0.045**	-0.195 ± 0.063
	LP	0.145 ± 0.031	0.763 ± 0.021	0.180 ± 0.053	-0.069 ± 0.015
Letter	Ada	**0.199 ± 0.010**	**0.804 ± 0.017**	**0.190 ± 0.005**	-0.309 ± 0.009
	LP	0.000 ± 0.000	0.905 ± 0.021	0.202 ± 0.012	0.000 ± 0.000
Magic04	Ada	**0.190 ± 0.007**	**0.716 ± 0.017**	**0.230 ± 0.006**	-0.412 ± 0.034
	LP	0.000 ± 0.000	0.859 ± 0.063	0.265 ± 0.017	0.000 ± 0.000
Mfeat-fac	Ada	**0.184 ± 0.008**	**0.538 ± 0.033**	**0.040 ± 0.009**	-0.018 ± 0.007
	LP	0.171 ± 0.009	0.558 ± 0.038	0.045 ± 0.010	0.033 ± 0.003
Optdigits	Ada	**0.173 ± 0.009**	**0.654 ± 0.022**	**0.111 ± 0.013**	-0.231 ± 0.016
	LP	0.017 ± 0.046	0.708 ± 0.027	0.127 ± 0.019	-0.010 ± 0.027
Page-block	Ada	0.278 ±0.014	**0.458 ± 0.037**	**0.048 ± 0.005**	-0.213 ± 0.023
	LP	0.232 ± 0.374	0.686 ± 0.218	0.055 ± 0.008	0.000 ± 0.000
Pendigits	Ada	**0.176 ± 0.006**	**0.634 ± 0.020**	**0.091 ± 0.006**	-0.243 ± 0.015
	LP	0.135 ± 0.046	0.711 ± 0.028	0.131 ± 0.010	-0.085 ± 0.029
Satimage	Ada	**0.262 ± 0.008**	**0594 ± 0.018**	**0.057 ± 0.005**	-0.161 ± 0.014
	LP	0.092 ± 0.280	0.771 ± 0.036	0.066 ± 0.007	0.000 ± 0.000
Shuttle	Ada	0.173 ± 0.017	**0.062 ± 0.038**	0.001 ± 0.000	-0.087 ± 0.026
	LP	**0.204 ± 0.032**	0.251 ± 0.065	0.001 ± 0.000	0.000 ± 0.000
Spambase	Ada	**0.315 ± 0.217**	**0.591 ± 0.201**	**0.055 ± 0.020**	-0.126 ± 0.365
	LP	0.116 ± 0.316	0.737 ± 0.257	0.080 ± 0.028	0.096 ± 0.291
Waveform	Ada	**0.371 ± 0.014**	**0.721 ± 0.013**	**0.096 ± 0.008**	-0.185 ± 0.014
	LP	0.000 ± 0.000	0.780 ±0.014	0.104 ± 0.011	0.000 ± 0.000

Table 3: Margin measures and performances of AdaBoost and LP-AdaBoost on the **large** data sets and using the **stump** base classifiers.

		Emargin	Emargin Error	Test Error	Min margin
Breast	Ada	0.312 ± 0.045	**0.425 ± 0.082**	**0.044 ± 0.016**	-0.048 ± 0.017
	LP	0.299 ± 0.068	0.556 ± 0.135	0.053 ± 0.017	0.022 ± 0.012
Diabetes	Ada	0.216 ± 0.017	**0.753 ± 0.033**	**0.228 ± 0.026**	-0.199 ± 0.018
	LP	0.149 ± 0.294	0.821 ± 0.071	0.271 ± 0.040	-0.008 ± 0.015
German	Ada	**0.221 ± 0.015**	**0.769 ± 0.029**	**0.240 ± 0.026**	-0.246 ± 0.018
	LP	0.059 ± 0.173	0.818 ± 0.073	0.272 ± 0.030	0.000 ± 0.000
Vehicle	Ada	0.196 ± 0.012	**0.688 ± 0.035**	0.223 ± 0.026	-0.102 ± 0.011
	LP	0.273 ± 0.285	0.790 ± 0.075	0.231 ± 0.029	-0.018 ± 0.008
Wdbc	Ada	**0.400 ± 0.032**	0.537 ± 0.048	**0.028 ± 0.014**	0.096 ± 0.012
	LP	0.376 ± 0.032	0.546 ± 0.050	0.033 ± 0.015	0.139 ± 0.008

Table 4: Margin measures and performances of AdaBoost and LP-AdaBoost on the **small** data sets and using the **stump** base classifiers.

		Emargin	Emargin Error	Test Error	Min margin
Image	Ada	0.370 ± 0.016	0.375 ± 0.034	0.010 ± 0.004	0.184 ± 0.008
	LP	0.374 ± 0.023	0.374 ± 0.054	0.010 ± 0.004	0.232 ± 0.007
Isolet	Ada	0.252 ± 0.076	0.589 ± 0.028	0.074 ± 0.067	0.020 ± 0.144
	LP	0.240 ± 0.010	0.591 ± 0.040	0.074 ± 0.056	0.063 ± 0.071
Letter	Ada	**0.246 ± 0.017**	**0.714 ± 0.034**	**0.077 ± 0.006**	-0.144 ± 0.012
	LP	0.236 ± 0.019	0.775 ± 0.031	0.086 ± 0.006	0.061 ± 0.004
Magic04	Ada	**0.312 ± 0.018**	**0.805 ± 0.018**	**0.156 ± 0.006**	-0.212 ± 0.012
	LP	0.282 ± 0.038	0.879 ± 0.028	0.225 ± 0.013	-0.085 ± 0.003
Mfeat-fac	Ada	**0.377 ± 0.029**	0.293 ± 0.104	0.017 ± 0.005	0.285 ± 0.006
	LP	0.350 ± 0.044	**0.146 ± 0.174**	0.018 ± 0.006	0.314 ± 0.005
Optdigits	Ada	0.288 ± 0.009	0.460 ± 0.025	0.018 ± 0.003	0.090 ± 0.006
	LP	0.288 ± 0.010	0.466 ± 0.022	0.018 ± 0.003	0.124 ± 0.004
Page-block	Ada	0.392 ± 0.024	**0.465 ± 0.038**	**0.030 ± 0.005**	-0.068 ± 0.009
	LP	**0.508 ± 0.041**	0.518 ± 0.057	0.033 ± 0.005	0.000 ± 0.000
Pendigits	Ada	**0.305 ± 0.008**	0.337 ± 0.017	0.005 ± 0.001	0.101 ± 0.008
	LP	0.301 ± 0.010	0.345 ± 0.022	0.005 ± 0.001	0.137 ± 0.005
Satimage	Ada	**0.319 ± 0.013**	0.484 ± 0.026	**0.044 ± 0.006**	0.012 ± 0.008
	LP	0.284 ± 0.014	0.496 ± 0.039	0.046 ± 0.006	0.055 ± 0.004
Shuttle	Ada	0.503 ± 0.037	**0.034 ± 0.020**	0.001 ± 0.000	-0.049 ± 0.013
	LP	**0.541 ± 0.066**	0.071 ± 0.042	0.001 ± 0.000	0.000 ± 0.000
Spambase	Ada	0.294 ± 0.014	**0.601 ± 0.034**	**0.052 ± 0.006**	-0.092 ± 0.008
	LP	0.309 ± 0.181	0.681 ± 0.077	0.067 ± 0.008	-0.002 ± 0.002
Waveform	Ada	**0.494 ± 0.023**	0.709 ± 0.011	**0.100 ± 0.009**	0.001 ± 0.006
	LP	0.473 ± 0.033	0.714 ± 0.018	0.103 ± 0.008	0.041 ± 0.003

Table 5: Margin measures and performances of AdaBoost and LP-AdaBoost on the **large** data sets and using the **Tree** base classifiers.

		Emargin	Emargin Error	Test Error	Min margin
Breast	Ada	0.591 ± 0.057	0.392 ± 0.051	**0.030 ± 0.014**	0.317 ± 0.030
	LP	**0.667 ± 0.059**	0.404 ± 0.053	0.033 ± 0.014	0.385 ± 0.033
Diabetes	Ada	0.230 ± 0.032	0.706 ± 0.062	**0.272 ± 0.027**	0.035 ± 0.007
	LP	0.222 ± 0.026	0.709 ± 0.058	0.284 ± 0.030	0.082 ± 0.004
German	Ada	**0.202 ± 0.015**	0.704 ± 0.041	**0.242 ± 0.027**	-0.010 ± 0.010
	LP	0.192 ± 0.017	0.703 ± 0.050	0.259 ± 0.028	0.046 ± 0.004
Vehicle	Ada	**0.271 ± 0.018**	0.644 ± 0.038	0.216 ± 0.029	0.087 ± 0.007
	LP	0.256 ± 0.020	0.633 ± 0.046	0.216 ± 0.027	0.127 ± 0.004
Wdbc	Ada	0.539 ± 0.018	0.015 ± 0.010	0.028 ± 0.013	0.527 ± 0.019
	LP	**0.582 ± 0.020**	**0.002 ± 0.000**	0.030 ± 0.014	0.582 ± 0.020

Table 6: Margin measures and performances of AdaBoost and LP-AdaBoost on the **small** data sets and using the **tree** base classifiers.

The results are described in Tables 3, 4, 5 and 6 respectively according to the type of base classifiers used and the size of the data sets. To highlight the results we use boldface in the following manner: By a t-test with significant level 0.01, **larger Emargin**, **smaller Emargin error**, and **smaller test error** are denoted in boldface. If on a data set, the empirical result agrees with the theory, the **name of the data set** is marked in boldface. For example, if one algorithm has larger Emargin, smaller or equal Emargin error, and smaller test error, then the data set is marked in boldface. Similarly, if one algorithm has smaller Emargin error, larger or equal Emargin, and smaller test error, then the data set is marked in boldface. Also if the two algorithms have (statistically) the same Emargin, Emargin error and test error, it agrees with the theory.

In Table 3 we use decision stump base classifiers on large data sets. We see that only one data set is not marked in boldface. On this "Shuttle" data set, LP-AdaBoost has a larger Emargin and also a larger Emargin error. In this case, the comparison theorem (Theorem 6) does not apply. We mark such data sets by italic font. Note that AdaBoost does not always have larger Emargin than LP-AdaBoost. On the "Image" data set, LP-AdaBoost achieves larger Emargin, smaller Emargin error and, as the bound predicts, a smaller test error.

In Table 4 we use decision stump base classifiers on small data sets. Four data sets agree with the theory. On the "Vehicle" data set, although the bound predicts that AdaBoost would have a smaller generalization error, the test error of AdaBoost is not significantly smaller than LP-AdaBoost.

In Table 5 we use eight-leave decision tree base classifiers on large data sets. Eight data sets agree with the theory. For the "Mfeat-fac", "Page-block" and "Shuttle" data sets, our comparison theorem does not apply. Only the "Pendigits" data set differs from the theoretical prediction: The test errors are the same while the theory predicts AdaBoost would perform better.

The last set of experiments, listed in Table 6, in which we use eight-leave decision tree base classifiers on small data sets, behaves different from all the previous results. Only one data set agrees with the theory. On the "Breast" data set, the test error is contrary to what the bound predicts.

To summarize, on large data sets, the Emargin theory usually agrees with empirical observations. AdaBoost has better performances because it has a larger Emargin and a smaller Emargin error. Note there are also cases that LP-AdaBoost achieves a larger Emargin and a smaller Emargin error and a smaller test error. However, on small data sets and with more complex base classifiers, the theory does not often give the correct predictions. We think the reason is that the bound is still loose, especially when the data set contains only a few hundred of points. Also the number of classifiers is a loose bound for the complexity of complex decision trees.

Finally we plot in Figure 1 some margin distribution graphs and the corresponding Emargin and Emargin errors to give an illustration. AdaBoost often has intuitively "better" margin distributions.

8. Conclusions

In this paper we provided a refined analysis on the margin theory for boosting algorithms, which extended our preliminary study (Wang et al., 2008). We proposed a bound in terms of a new margin measure called the Emargin, which depends on the whole margin distribution. This bound is uniformly sharper than the minimum margin bound whose prediction is different from the empirical observations. Our theory suggests that a boosting classifier may not be necessarily achieve better performance even though it generates a larger minimum margin.

Our bound suggests that the Emargin and the Emargin error play important roles to guarantee a smaller bound of the generalization error of a voting classifier—a larger Emargin and a smaller

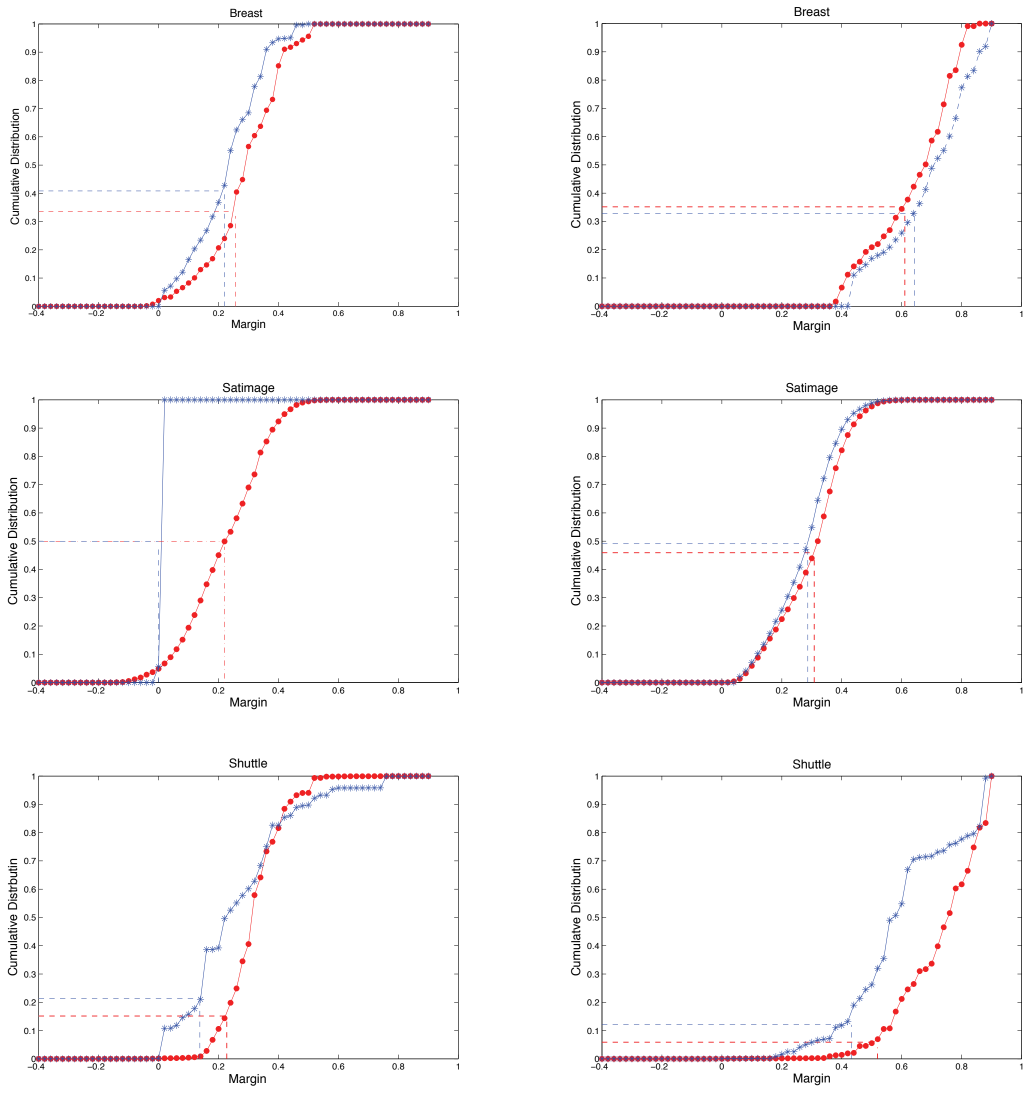


Figure 1: Margin distribution graphs with Emargin and Emargin errors. The lines marked with stars are the margin distributions of LP-AdaBoost. The lines marked with circles are of AdaBoost. Emargin and Emargin errors are plotted by lines parallel to the axes. The left column uses decision stump base classifiers, the right column uses decision tree classifiers. The three rows are from the data sets of Breast, Satimage and Shuttle respectively.

Emargin error result in better generalization ability. Experimental results on (not-too-small) benchmark data sets agree well with our theory.

From a practical point of view, the Emargin bound is still too loose to give useful quantitative predictions. For most data sets, the bound is larger than $1/2$. On the other hand we can employ the bound to "compare" voting classifiers with the help of Emargin and Emargin error. This provides some guidance to choose classifiers. To calculate the Emargin, one needs to know the complexity (e.g., VC dimension) of the base classifiers. This can be difficult for some base learners like C4.5 decision trees.

A future work is to develop algorithms that generate voting classifiers with good margin distributions, that is, large Emargin and small Emargin error. Directly optimizing Emargin and Emargin error would be computationally difficult. On the other hand, given a voting classifier $\sum \alpha_t h_t$, it might be possible to improve its margin distribution. One way is to solve the following linear optimization problem to obtain $\sum \beta_t h_t$.

$$\begin{aligned} \max_{\beta,\xi} \quad & \sum \xi_i \\ s.t. \quad & y_i \sum \beta_t h_t(x_i) \geq y_i \sum \alpha_t h_t(x_i) + \xi_i, \quad i = 1,2,\ldots \\ & \beta_t \geq 0, \quad \sum \beta_t = 1, \\ & \xi_i \geq 0, \end{aligned} \tag{18}$$

where $\alpha = (\alpha_1, \ldots, \alpha_T)$, $\beta = (\beta_1, \ldots, \beta_T)$, $\xi = (\xi_1, \ldots, \xi_n)$. If there is a nontrivial solution (i.e., $\beta \neq \alpha$), $\sum \beta_t h_t$ would have a uniformly better margin distribution than $\sum \alpha_t h_t$ and therefore we expect it has a smaller generalization error. However, there is usually no nontrivial solutions when $\sum \alpha_t h_t$ is an AdaBoost classifier—it already has a good margin distribution. An open problem is to modify and relax (18) and obtain a solution with larger Emargin and smaller Emargin error. Then it would be a good test to see if such a classifier achieves better performance as our theory predicts.

Acknowledgments

We thank the referees for their useful and insightful comments. It greatly improves the quality of the paper. The first author would like to thank Phil Long, Gilles Blanchard and Lev Reyzin for helpful discussions. This work was supported by NSFC(61075003, 61073097), Global COE Program of Tokyo Institute of Technology and JiangsuSF(BK2008018) and the National Fundamental Research Program of China(2010CB327903, 2011CB302400). Part of the work was done when the first author was visiting Tokyo Institute of Technology.

References

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

P. Bartlett, M. Jordan, and J.D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101:138–156, 2006.

E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36:105–139, 1999.

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26:801–849, 1998.

L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11:1493–1517, 1999.

L. Breiman. Population theory for boosting ensembles. *Annals of Statistics*, 32:1–11, 2004.

R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *23th International Conference on Machine Learning*, 2006.

L. Devroye. Bounds for the uniform deviation of empirical measures. *Journal of Multivariate Analysis*, 12:72–79, 1982.

T. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40:139–157, 2000.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, 1996.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.

A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *National Conference on Artificial Intelligence*, 1998.

W. Hoeffding. Probability inequalities for sum of bounded random variables. *Journal of American Statistical Society*, 58:13–30, 1963.

W. Jiang. Process consistency for adaboost. *The Annals of Statistics*, 32:13–29, 2004.

V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30:1–50, 2002.

V. Koltchinskii and D. Panchenko. Complexities of convex combinations and bounding the generalization error in classification. *Annals of Statistics*, 33:1455–1496, 2005.

J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005.

G. Lugosi and Nicolas Vayatis. On the bayes-risk consistency of regularized boosting methods. *The Annals of Statistics*, 32:30–55, 2004.

D. Mease and A. Wyner. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156, 2008.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning*, pages 118–183, 2003.

J. R. Quinlan. Bagging, boosting, and c4.5. In *13th International Conference on Artificial Intelligence*, 1996.

G. Rätsch and M. Warmuth. Efficient margin maximization with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.

L. Reyzin and R. E. Schapire. How boosting the margin can also boost classifier complexity. In *International Conference on Machine Learning*, 2006.

C. Rudin, I. Daubechies, and R. Schapire. The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, Dec 2004.

C. Rudin, I. Daubechies, and R. Schapire. Analysis of boosting algorithms using the smooth margin function. *Annals of Statistics*, 35:2723–2768, 2007.

N. Sauer. On the density of family of sets. *Journal of Combinatorial Theory, Series A*, 13:145–147, 1972.

R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26:1651–1686, 1998.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons Inc., 1998.

V. N. Vapnik and A. YA. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

L. Wang, C. Yang, and J. Feng. On learning with dissimilarity functions. In *24th International Conference on Machine Learning*, 2007.

L. Wang, M. Sugiyama, C. Yang, Z. Zhou, and J. Feng. On the margin explanation of boosting algorithms. In *21th Annual Conference on Learning Theory*, 2008.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85, 2004.

Journal of Machine Learning Research 12 (2011) 1865-1892 Submitted 12/10; Revised 5/11; Published 6/11

Stochastic Methods for ℓ_1-regularized Loss Minimization

Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Givat Ram, Jerusalem 91904, Israel

Ambuj Tewari AMBUJ@CS.UTEXAS.EDU
Computer Science Department
The University of Texas at Austin
Austin, TX 78701, USA

Editor: Leon Bottou

Abstract

We describe and analyze two stochastic methods for ℓ_1 regularized loss minimization problems, such as the Lasso. The first method updates the weight of a single feature at each iteration while the second method updates the entire weight vector but only uses a single training example at each iteration. In both methods, the choice of feature or example is uniformly at random. Our theoretical runtime analysis suggests that the stochastic methods should outperform state-of-the-art deterministic approaches, including their deterministic counterparts, when the size of the problem is large. We demonstrate the advantage of stochastic methods by experimenting with synthetic and natural data sets.[1]

Keywords: L1 regularization, optimization, coordinate descent, mirror descent, sparsity

1. Introduction

We present optimization procedures for solving problems of the form:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w},\mathbf{x}_i\rangle, y_i) + \lambda\|\mathbf{w}\|_1 \ , \tag{1}$$

where $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_m,y_m) \in ([-1,+1]^d \times \mathcal{Y})^m$ is a sequence of training examples, $L:\mathbb{R}^d\times\mathcal{Y}\to[0,\infty)$ is a non-negative loss function, and $\lambda>0$ is a regularization parameter. This generic problem includes as special cases the Lasso (Tibshirani, 1996), in which $L(a,y)=\frac{1}{2}(a-y)^2$, and logistic regression, in which $L(a,y)=\log(1+\exp(-ya))$.

Our methods can also be adapted to deal with additional boxed constraints of the form $w_i \in [a_i,b_i]$, which enables us to use them for solving the dual problem of Support Vector Machine (Cristianini and Shawe-Taylor, 2000). For concreteness, we focus on the formulation given in (1).

Throughout the paper, we assume that L is convex in its first argument. This implies that (1) is a convex optimization problem, and therefore can be solved using standard optimization techniques, such as interior point methods. However, standard methods scale poorly with the size of the problem (i.e., m and d). In recent years, machine learning methods are proliferating in data-laden domains such as text and web processing in which data sets of millions of training examples or features are

1. An initial version of this work (Shalev-Shwartz and Tewari, 2009) appeared in ICML 2009.

not uncommon. Since traditional methods for solving (1) generally scale very poorly with the size of the problem, their usage is inappropriate for data-laden domains. In this paper, we discuss how to overcome this difficulty using stochastic methods. We describe and analyze two practical methods for solving (1) even when the size of the problem is very large.

The first method we propose is a stochastic version of the familiar coordinate descent approach. The coordinate descent approach for solving ℓ_1 regularized problems is not new (as we survey below in Section 1.1). At each iteration of coordinate descent, a single element of $\mathbf{w}$ is updated. The only twist we propose here regarding the way one should choose the next feature to update. We suggest to choose features uniformly at random from the set $[d] = \{1, \ldots, d\}$. This simple modification enables us to show that the runtime required to achieve ε (expected) accuracy is upper bounded by

$$\frac{m d \beta \, \|\mathbf{w}^\star\|_2^2}{\varepsilon} , \tag{2}$$

where β is a constant which only depends on the loss function (e.g., $\beta = 1$ for the quadratic loss function) and $\mathbf{w}^\star$ is the optimal solution. This bound tells us that the runtime grows only linearly with the size of the problem. Furthermore, the stochastic method we propose is parameter free and very simple to implement.

Another well known stochastic method that has been successfully applied for loss minimization problems, is stochastic gradient descent (e.g., Bottou and LeCunn, 2005; Shalev-Shwartz et al., 2007). In stochastic gradient descent, at each iteration, we pick one example from the training set, uniformly at random, and update the weight vector based on the chosen example. The attractiveness of stochastic gradient descent methods is that their runtime do not depend at all on the number of examples, and can even sometime decrease with the number of examples (see Bottou and Bousquet, 2008; Shalev-Shwartz and Srebro, 2008). Unfortunately, the stochastic gradient descent method fails to produce sparse solutions, which makes the algorithm both slower and less attractive as sparsity is one of the major reasons to use ℓ_1 regularization. To overcome this problem, two variants were recently proposed. First, Duchi et al. (2008) suggested to replace the ℓ_1 regularization term with a constraint of the form $\|\mathbf{w}\|_1 \leq B$, and then to use stochastic gradient projection procedure. Another solution, which uses the regularization form given in (1), has been proposed by Langford et al. (2009) and is called truncated gradient descent. In this approach, the elements of $\mathbf{w}$ that cross 0 after the stochastic gradient step are truncated to 0, hence sparsity is achieved. The disadvantage of both Duchi et al. (2008) and Langford et al. (2009) methods is that, in some situations, their runtime might grow quadratically with the dimension d, even if the optimal predictor $\mathbf{w}^\star$ is very sparse (see Section 1.1 below for details). This quadratic dependence on d can be avoided if one uses mirror descent updates (Beck and Teboulle, 2003) such as the exponentiated gradient approach (Littlestone, 1988; Kivinen and Warmuth, 1997; Beck and Teboulle, 2003). However, this approach again fails to produce sparse solutions. In this paper, we combine the idea of truncating the gradient (Langford et al., 2009) with another variant of stochastic mirror descent, which is based on p-norm updates (Grove et al., 2001; Gentile, 2003). The resulting algorithm both produces sparse solutions and has $\tilde{O}(d)$ dependence on the dimension. We call the algorithm SMIDAS for "Stochastic MIrror Descent Algorithm made Sparse".

We provide runtime guarantees for SMIDAS as well. In particular, for the logistic-loss and the squared-loss we obtain the following upper bound on the runtime to achieving ε expected accuracy:

$$O\left(\frac{d \log(d) \, \|\mathbf{w}^\star\|_1^2}{\varepsilon^2}\right) . \tag{3}$$

Comparing the above with the runtime bound of the stochastic coordinate descent method given in (2) we note three major differences. First, while the bound in (2) depends on the number of examples, m, the runtime of SMIDAS does not depend on m at all. On the flip side, the dependence of stochastic coordinate descent on the dimension is better both because the lack of the term $\log(d)$ and because $\|\mathbf{w}^\star\|_2^2$ is always smaller than $\|\mathbf{w}^\star\|_1^2$ (the ratio is at most d). Last, the dependence on $\frac{1}{\varepsilon}$ is linear in (2) and quadratic in (3). If ε is the same order as the objective value at $\mathbf{w}^\star$, it is possible to improve the dependence on $1/\varepsilon$ (Proposition 4). Finally, we would like to point out that while the stochastic coordinate descent method is parameter free, the success of SMIDAS and of the method of Langford et al. (2009), depends on a careful tuning of a learning rate parameter.

1.1 Related Work

We now survey several existing methods and in particular show how our stochastic twist enables us to give superior runtime guarantees.

1.1.1 COORDINATE DESCENT METHODS FOR ℓ_1 REGULARIZATION

Following the Gauss-Siedel approach of Zhang and Oles (2001), Genkin et al. (2007) described a coordinate descent method (called BBR) for minimizing ℓ_1 regularized objectives. This approach is similar to our method, with three main differences. First, and most important, at each iteration we choose a coordinate uniformly at random. This allows us to provide theoretical runtime guarantees. We note that no theoretical guarantees are provided by Zhang and Oles (2001) and Genkin et al. (2007). Second, we solely use gradient information which makes our algorithm parameters-free and extremely simple to implement. In contrast, the Gauss-Siedel approach is more complicated and involves second order information, or a line search procedure, or a trusted region Newton step. Last, the generality of our derivation allows us to tackle a more general problem. For example, it is easy to deal with additional boxed constraints. Friedman et al. (2010) generalized the approach of Genkin et al. (2007) to include the case of elastic-net regularization. In a series of experiments, they observed that cyclic coordinate descent outperforms many alternative popular methods such as LARS (Efron et al., 2004), an interior point method called `l1logreg` (Koh et al., 2007), and the Lasso Penalized Logistic (LPL) program (Wu and Lange, 2008). However, no theoretical guarantees are provided in Friedman et al. (2010) as well. Our analysis can partially explain the experimental result of Friedman et al. (2010) since updating the coordinates in a cyclic order can in practice be very similar to stochastic updates.

Luo and Tseng (1992) established a linear convergence result for coordinate descent algorithms. This convergence result tells us that after an unspecified number of iterations, the algorithm converges very fast to the optimal solution. However, this analysis is useless in data laden domains as it can be shown that the initial unspecified number of iterations depends at least quadratically on the number of training examples. In an attempt to improve the dependence on the size of the problem, Tseng and Yun (2009) recently studied other variants of block coordinate descent for optimizing 'smooth plus separable' objectives. In particular, ℓ_1 regularized loss minimization (1) is of this form, provided that the loss function is smooth. The algorithm proposed by Tseng and Yun (2009) is not stochastic. Translated to our notation, the runtime bound given in Tseng and Yun (2009) is of order[2] $\frac{md^2\beta\|\mathbf{w}^\star\|_2^2}{\varepsilon}$. This bound is inferior to our runtime bound for stochastic coordinate descent given in (2) by a factor of the dimension d.

1.1.2 COORDINATE DESCENT METHODS FOR ℓ_1 DOMAIN CONSTRAINTS

A different, but related, optimization problem is to minimize the loss, $\frac{1}{m}\sum_i L(\langle\mathbf{w},\mathbf{x}_i\rangle, y_i)$, subject to a domain constraint of the form $\|\mathbf{w}\|_1 \le B$. Many authors presented a forward greedy selection algorithm (a.k.a. Boosting) for this problem. We refer the reader to Frank and Wolfe (1956), Zhang (2003), Clarkson (2008) and Shalev-Shwartz et al. (2010). These authors derived the upper bound $O\left(\beta\|\mathbf{w}^\star\|_1^2/\varepsilon\right)$ on the number of iterations required by this algorithm to find an ε-accurate solution. Since at each iteration of the algorithm, one needs to calculate the gradient of the loss at $\mathbf{w}$, the runtime of each iteration is md. Therefore, the total runtime becomes $O\left(md\beta\|\mathbf{w}^\star\|_1^2/\varepsilon\right)$. Note that this bound is better than the bound given by Tseng and Yun (2009), since for any vector in $\mathbb{R}^d$ we have $\|\mathbf{w}\|_1 \le \sqrt{d}\|\mathbf{w}\|_2$. However, the boosting bound given above is still inferior to our bound given in (2) since $\|\mathbf{w}^\star\|_1 \ge \|\mathbf{w}^\star\|_2$. Furthermore, in the extreme case we have $\|\mathbf{w}^\star\|_1^2 = d\|\mathbf{w}^\star\|_2^2$, thus our bound can be better than the boosting bound by a factor of d. Lemma 5 in Appendix A shows that the *iteration* bound (not runtime) of *any* algorithm cannot be smaller than $\Omega(\|\mathbf{w}^\star\|_1^2/\varepsilon)$ (see also the lower bounds in Shalev-Shwartz et al., 2010). This seems to imply that *any* deterministic method, which goes over the entire data at each iteration, will induce a runtime which is inferior to the runtime we derive for stochastic coordinate descent.

1.1.3 STOCHASTIC GRADIENT DESCENT AND MIRROR DESCENT

Stochastic gradient descent (SGD) is considered to be one of the best methods for large scale loss minimization, when we measure how fast a method achieves a certain generalization error. This has been observed in experiments (Bottou, Web Page) and also has been analyzed theoretically by Bottou and Bousquet (2008) and Shalev-Shwartz and Srebro (2008).

As mentioned before, one can apply SGD for solving (1). However, SGD fails to produce sparse solutions. Langford et al. (2009) proposed an elegant simple modification of the SGD update rule that yields a variant of SGD with sparse intermediate solutions. They also provide bounds on the runtime of the resulting algorithm. In the general case (i.e., without assuming low objective relative to ε), their analysis implies the following runtime bound

$$O\left(\frac{d\|\mathbf{w}^\star\|_2^2 X_2^2}{\varepsilon^2}\right), \tag{4}$$

where $X_2^2 = \frac{1}{m}\sum_i \|\mathbf{x}_i\|_2^2$ is the average squared norm of an instance. Comparing this bound with our bound in (3), we observe that none of the bounds dominates the other, and their relative performance depends on properties of the training set and the optimal solution $\mathbf{w}^\star$. Specifically, if $\mathbf{w}^\star$ has only $k \ll d$ non-zero elements and each $\mathbf{x}_i$ is dense (say $\mathbf{x}_i \in \{-1,+1\}^d$), then the ratio between the above bound of SGD and the bound in (3) becomes $\frac{d}{k\log(d)} \gg 1$. On the other hand, if $\mathbf{x}_i$ has only k non-zeros while $\mathbf{w}^\star$ is dense, then the ratio between the bounds can be $\frac{k}{d\log(d)} \ll 1$. Although the relative performance is data dependent, in most applications if one prefers ℓ_1 regularization over ℓ_2

2. To see this, note that the iterations bound in Equation (21) of Tseng and Yun (2009) is: $\frac{\beta\|\mathbf{w}^\star\|_2^2}{\varepsilon\nu}$, and using Equation (25) in Section 6, we can set the value of ν to be $\nu = 1/d$ (since in our case there are no linear constraints). The complexity bound now follows from the fact that the cost of each iteration is $O(dm)$.

regularization, he should also believe that $\mathbf{w}^\star$ is sparse, and thus our runtime bound in (3) is likely to be superior.[3]

The reader familiar with the online learning and mirror descent literature will not be surprised by the above discussion. Bounds that involved $\|\mathbf{w}^\star\|_1$ and $\|\mathbf{x}_i\|_\infty$, as in (3), are well known and the relative performance discussed above was pointed out in the context of additive vs. multiplicative updates (see, e.g., Kivinen and Warmuth, 1997). However, the most popular algorithm for obtaining bounds of the form given in (3) is the EG approach (Kivinen and Warmuth, 1997), which involves the exponential potential, and this algorithm cannot yield intermediate sparse solutions. One of the contributions of this paper is to show that with a different potential, which is called the p-norm potential, one can obtain the bound given in (3) while still enjoying sparse intermediate solutions.

1.1.4 RECENT WORKS DEALING WITH STOCHASTIC METHODS FOR LARGE SCALE REGULARIZED LOSS MINIMIZATION

Since the publication of the conference version (Shalev-Shwartz and Tewari, 2009) of this paper, several papers proposing stochastic algorithms for regularized loss minimization have appeared. Of these, we would like to mention a few that are especially connected to the themes pursued in the present paper. Regularized Dual Averaging (RDA) of Xiao (2010) uses a running average of all the past subgradients of the loss function and the regularization term to generate its iterates. He develops a p-norm RDA method that is closely related to SMIDAS. The theoretical bounds for SMIDAS and p-norm RDA are similar but the latter employs a more aggressive truncation schedule that can potentially lead to sparser iterates.

SMIDAS deals with ℓ_1 regularization. The Composite Objective MIrror Descent (COMID) algorithm of Duchi et al. (2010) generalizes the idea behind SMIDAS to deal with general regularizers provided a certain minimization problem involving a Bregman divergence and the regularizer is efficiently solvable. Viewing the average loss in (1) leads to interesting connections with the area of Stochastic Convex Optimization that deals with minimizing a convex function given access to an oracle that can return unbiased estimates of the gradient of the convex function at any query point. For various classes of convex functions, one can ask: What is the optimal number of queries needed to achieve a certain accuracy (in expectation)? For developments along these lines, please see Lan (2010) and Ghadimi and Lan (2011), especially the latter since it deals with functions that are the sum of a smooth and a non-smooth but "simple" (like ℓ_1-norm) part. Finally, Nesterov (2010) has analyzed randomized versions of coordinate descent for unconstrained and constrained minimization of smooth convex functions.

2. Stochastic Coordinate Descent

To simplify the notation throughout this section, we rewrite the problem in (1) using the notation

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \overbrace{\underbrace{\frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w},\mathbf{x}_i\rangle, y_i)}_{\equiv C(\mathbf{w})} + \lambda\|\mathbf{w}\|_1}^{\equiv P(\mathbf{w})} \ . \tag{5}$$

3. One important exception is the large scale text processing application described in Langford et al. (2009) where the dimension is so large and ℓ_1 is used simply because we cannot store a dense weight vector in memory.

We are now ready to present the stochastic coordinate descent algorithm. The algorithm initializes $\mathbf{w}$ to be $\mathbf{0}$. At each iteration, we pick a coordinate j uniformly at random from $[d]$. Then, the derivative of $C(\mathbf{w})$ w.r.t. the jth element of $\mathbf{w}$, $g_j = (\nabla C(\mathbf{w}))_j$, is calculated. That is, $g_j = \frac{1}{m}\sum_{i=1}^m L'(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)x_{i,j}$, where L' is the derivative of the loss function with respect to its first argument. Simple calculus yields

$$L'(a,y) = \begin{cases} (a-y) & \text{for squared-loss} \\ \frac{-y}{1+\exp(ay)} & \text{for logistic-loss} \end{cases} . \tag{6}$$

Next, a step size is determined based on the value of g_j and a parameter of the loss function denoted by β. This parameter is an upper bound on the second derivative of the loss. Again, for our running examples we have

$$\beta = \begin{cases} 1 & \text{for squared-loss} \\ 1/4 & \text{for logistic-loss} \end{cases} . \tag{7}$$

If there was no regularization, we would just subtract the step size g_j/β from the current value of w_j. However, to take into account the regularization term, we further add/subtract λ/β from w_j provided we do not cross 0 in the process. If we do, we let the new value of w_j be exactly 0. This is crucial for maintaining sparsity of $\mathbf{w}$. To describe the entire update succinctly, it is convenient to define the following simple "thresholding" operation:

$$s_\tau(w) = \operatorname{sign}(w)(|w|-\tau)_+ = \begin{cases} 0 & w \in [-\tau,\tau] \\ w-\tau & w > \tau \\ w+\tau & w < -\tau \end{cases} .$$

Algorithm 1 Stochastic Coordinate Descent (SCD)

```
let $\mathbf{w} = \mathbf{0}$
**for** $t = 1,2,\dots$ **do**
   sample $j$ uniformly at random from $\{1,\dots,d\}$
   let $g_j = (\nabla C(\mathbf{w}))_j$
   $\mathbf{w}_j \leftarrow s_{\lambda/\beta}(w_j - g_j/\beta)$
**end for**
```

2.1 Efficient Implementation

We now present an efficient implementation of Algorithm 1. The simple idea is to maintain a vector $\mathbf{z} \in \mathbb{R}^m$ such that $z_i = \langle \mathbf{w}, \mathbf{x}_i\rangle$. Once we have this vector, calculating g_j on average requires $O(sm)$ iterations, where

$$s = \frac{|\{(i,j) : x_{i,j} \neq 0\}|}{md} \tag{8}$$

is the average number of non-zeros in our training set. Concretely, we obtain Algorithm 2 for logistic-loss and squared-loss.

Algorithm 2 SCD for logistic-loss and squared-loss

```
let $\mathbf{w} = \mathbf{0} \in \mathbb{R}^d$, $\mathbf{z} = \mathbf{0} \in \mathbb{R}^m$
for $t = 1, 2, \ldots$ do
  sample $j$ uniformly at random from $\{1, \ldots, d\}$
  let $L'$ and $\beta$ be as defined in (6) and (7)
  let $g_j = \frac{1}{m} \sum_{i: x_{i,j} \neq 0} L'(z_i, y_i) x_{i,j}$
  if $w_j - g_j/\beta > \lambda/\beta$ then
    $w_j \leftarrow w_j - g_j/\beta - \lambda/\beta$
  else if $w_j - g_j/\beta < -\lambda/\beta$ then
    $w_j \leftarrow w_j - g_j/\beta + \lambda/\beta$
  else
    $w_j \leftarrow 0$
  end if
  $\forall i$ s.t. $x_{i,j} \neq 0$ let $z_i = z_i + \eta x_{i,j}$
end for
```

2.2 Runtime Guarantee

The following theorem establishes runtime guarantee for SCD.

Theorem 1 *Let* $\mathbf{w}^\star$ *be a minimizer of* (5) *where the function* $C(\mathbf{w})$ *is differentiable and satisfies,*

$$\forall \mathbf{w}, \eta, j,\ C(\mathbf{w} + \eta \mathbf{e}^j) \leq C(\mathbf{w}) + \eta [\nabla C(\mathbf{w})]_j + \tfrac{\beta}{2} \eta^2 \ . \tag{9}$$

Let $\mathbf{w}_T$ *denote the weight vector* $\mathbf{w}$ *at the end of iteration* T *of Algorithm 1. Then,*

$$\mathbb{E}[P(\mathbf{w}_T)] - P(\mathbf{w}^\star) \ \leq \ \frac{d\,\Psi(0)}{T+1} \ ,$$

where

$$\Psi(\mathbf{w}) = \frac{\beta}{2} \|\mathbf{w}^\star - \mathbf{w}\|_2^2 + P(\mathbf{w})$$

and the expectation is over the algorithm's own randomization.

Proof To simplify the proof, let us rewrite the update as $w_j \leftarrow w_j + \eta_j$ where $\eta_j = s_{\lambda/\beta}(w_j - g_j/\beta) - w_j$. We first show that

$$\eta_j = \operatorname*{argmin}_{\eta} \left(\eta g_j + \tfrac{\beta}{2} \eta^2 + \lambda |w_{t-1,j} + \eta| \right) \ . \tag{10}$$

Indeed, if η is a solution of the above then by optimality conditions, we must have,

$$0 = g_j + \beta \eta + \lambda \rho_j \ ,$$

where $\rho_j \in \partial |w_{t-1,j} + \eta|$, the sub-differential of the absolute value function at $w_{t-1,j} + \eta$. Since $\rho_j = \operatorname{sign}(w_{t-1,j} + \eta)$ if $w_{t-1,j} + \eta \neq 0$ and otherwise $\rho_j \in [-1, 1]$, we obtain that:

$$\text{If } \eta > -w_{t-1,j} \ \Rightarrow\ \rho_j = 1 \ \Rightarrow\ \eta = \tfrac{-g_j - \lambda}{\beta} > -w_{t-1,j}$$

$$\text{If } \eta < -w_{t-1,j} \ \Rightarrow\ \rho_j = -1 \ \Rightarrow\ \eta = \tfrac{-g_j + \lambda}{\beta} < -w_{t-1,j}$$

$$\text{Else } \eta = -w_{t-1,j} \ .$$

But, this is equivalent to the definition of η_j and therefore (10) holds.

Define the potential,

$$\Phi(\mathbf{w}_t) = \tfrac{1}{2}\|\mathbf{w}^\star - \mathbf{w}_t\|_2^2 \ ,$$

and let $\Delta_{t,j} = \Phi(\mathbf{w}_{t-1}) - \Phi(\mathbf{w}_{t-1} + \eta_j \mathbf{e}^j)$ be the change in the potential assuming we update $\mathbf{w}_{t-1}$ using coordinate j. Since $0 = g_j + \beta\eta_j + \lambda\rho_j$, we have that,

$$\begin{aligned}\Delta_{t,j} &= \tfrac{1}{2}\|\mathbf{w}^\star - \mathbf{w}_{t-1}\|_2^2 - \tfrac{1}{2}\|\mathbf{w}^\star - \mathbf{w}_{t-1} - \eta_j\mathbf{e}^j\|_2^2 \\ &= \tfrac{1}{2}(w_j^\star - w_{t-1,j})^2 - \tfrac{1}{2}(w_j^\star - w_{t-1,j} - \eta_j)^2 \\ &= \tfrac{1}{2}\eta_j^2 - \eta_j(w_{t-1,j} + \eta_j - w_j^\star) \\ &= \tfrac{1}{2}\eta_j^2 + \frac{g_j}{\beta}(w_{t-1,j} + \eta_j - w_j^\star) + \frac{\lambda\rho_j}{\beta}(w_{t-1,j} + \eta_j - w_j^\star).\end{aligned}$$

Next, we note that

$$\rho_j(w_{t-1,j} + \eta_j - w_j^\star) \geq |w_{t-1,j} + \eta_j| - |w_j^\star| \ ,$$

which yields

$$\Delta_{t,j} \geq \tfrac{1}{2}\eta_j^2 + \frac{g_j}{\beta}(w_{t-1,j} + \eta_j - w_j^\star) + \frac{\lambda}{\beta}(|w_{t-1,j} + \eta_j| - |w_j^\star|) \ .$$

By (9), we have,

$$C(\mathbf{w}_{t-1} + \eta_j\mathbf{e}^j) - C(\mathbf{w}_{t-1}) \leq g_j\eta_j + \frac{\beta}{2}\eta_j^2 \ ,$$

and thus

$$\Delta_{t,j} \geq \frac{1}{\beta}(C(\mathbf{w}_{t-1} + \eta_j\mathbf{e}^j) - C(\mathbf{w}_{t-1})) + \frac{g_j}{\beta}(w_{t-1,j} - w_j^\star) + \frac{\lambda}{\beta}(|w_{t-1,j} + \eta_j| - |w_j^\star|) \ .$$

Taking expectations (with respect to the choice of j and conditional on $\mathbf{w}_{t-1}$) on both sides, we get,

$$\begin{aligned}&\mathbb{E}[\Phi(\mathbf{w}_{t-1}) - \Phi(\mathbf{w}_t) \,|\, \mathbf{w}_{t-1}] \ = \ \frac{1}{d}\sum_{k=1}^d \Delta_{t,k} \\ &\geq \frac{1}{\beta d}\left[\sum_{k=1}^d (C(\mathbf{w}_{t-1} + \eta_k\mathbf{e}^k) - C(\mathbf{w}_{t-1})) + \sum_{k=1}^d g_k(w_{t-1,k} - w_k^\star) + \lambda\sum_{k=1}^d (|w_{t-1,k} + \eta_k| - |w_k^\star|)\right] \\ &= \frac{1}{\beta d}\left[\sum_{k=1}^d (C(\mathbf{w}_{t-1} + \eta_k\mathbf{e}^k) - C(\mathbf{w}_{t-1})) + \langle \nabla C(\mathbf{w}_{t-1}), \mathbf{w}_{t-1} - \mathbf{w}^\star\rangle + \lambda\sum_{k=1}^d (|w_{t-1,k} + \eta_k| - |w_k^\star|)\right] \\ &\geq \frac{1}{\beta d}\left[\sum_{k=1}^d (C(\mathbf{w}_{t-1} + \eta_k\mathbf{e}^k) - C(\mathbf{w}_{t-1})) + C(\mathbf{w}_{t-1}) - C(\mathbf{w}^\star) + \lambda\sum_{k=1}^d (|w_{t-1,k} + \eta_k| - |w_k^\star|)\right] \\ &= \frac{1}{\beta}\left[\mathbb{E}[C(\mathbf{w}_t) \,|\, \mathbf{w}_{t-1}] - C(\mathbf{w}_{t-1}) + \frac{C(\mathbf{w}_{t-1}) - C(\mathbf{w}^\star)}{d} + \frac{\lambda}{d}\sum_{k=1}^d |w_{t-1,k} + \eta_k| - \frac{\lambda\|\mathbf{w}^\star\|_1}{d}\right] \ ,\end{aligned}$$

where the second inequality follows from the convexity of C. Note that, we have,

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_t\|_1 \,|\, \mathbf{w}_{t-1}] &= \frac{1}{d}\sum_{k=1}^{d} \|\mathbf{w}_{t-1} + \eta_k \mathbf{e}^k\|_1 \\ &= \frac{1}{d}\sum_{k=1}^{d} (\|\mathbf{w}_{t-1}\|_1 - |w_{t-1,k}| + |w_{t-1,k} + \eta_k|) \\ &= \|\mathbf{w}_{t-1}\|_1 - \frac{1}{d}\|\mathbf{w}_{t-1}\|_1 + \frac{1}{d}\sum_{k=1}^{d} |w_{t-1,k} + \eta_k| \ .\end{aligned}$$

Plugging this above gives us,

$$\begin{aligned}&\beta\mathbb{E}[\Phi(\mathbf{w}_{t-1}) - \Phi(\mathbf{w}_t) \,|\, \mathbf{w}_{t-1}] \\ &\geq \mathbb{E}[C(\mathbf{w}_t) + \lambda\|\mathbf{w}_t\|_1 \,|\, \mathbf{w}_{t-1}] - C(\mathbf{w}_{t-1}) - \lambda\|\mathbf{w}_{t-1}\|_1 + \frac{C(\mathbf{w}_{t-1}) + \lambda\|\mathbf{w}_{t-1}\|_1 - C(\mathbf{w}^\star) - \lambda\|\mathbf{w}^\star\|_1}{d} \\ &= \mathbb{E}[P(\mathbf{w}_t) \,|\, \mathbf{w}_{t-1}] - P(\mathbf{w}_{t-1}) + \frac{P(\mathbf{w}_{t-1}) - P(\mathbf{w}^\star)}{d} \ .\end{aligned}$$

This is equivalent to,

$$\mathbb{E}[\beta\Phi(\mathbf{w}_{t-1}) + P(\mathbf{w}_{t-1}) - \beta\Phi(\mathbf{w}_t) - P(\mathbf{w}_t) \,|\, \mathbf{w}_{t-1}] \geq \frac{P(\mathbf{w}_{t-1}) - P(\mathbf{w}^\star)}{d} \ .$$

Thus, defining the composite potential,

$$\Psi(\mathbf{w}) = \beta\Phi(\mathbf{w}) + P(\mathbf{w}) \ ,$$

and taking full expectations, we get,

$$\mathbb{E}[\Psi(\mathbf{w}_{t-1}) - \Psi(\mathbf{w}_t)] \geq \frac{1}{d}\mathbb{E}[P(\mathbf{w}_{t-1}) - P(\mathbf{w}^\star)] \ .$$

Summing over $t = 1, \ldots, T+1$ and realizing that $P(\mathbf{w}_t)$ monotonically decreases gives,

$$\begin{aligned}\mathbb{E}\left[\tfrac{T+1}{d}(P(\mathbf{w}_T) - P(\mathbf{w}^\star))\right] &\leq \mathbb{E}\left[\tfrac{1}{d}\sum_{t=1}^{T+1}(P(\mathbf{w}_{t-1}) - P(\mathbf{w}^\star))\right] \\ &\leq \mathbb{E}\left[\sum_{t=1}^{T+1}(\Psi(\mathbf{w}_{t-1}) - \Psi(\mathbf{w}_t))\right] \\ &= \mathbb{E}\left[\Psi(\mathbf{w}_0) - \Psi(\mathbf{w}_{T+1})\right] \ \leq \ \mathbb{E}\left[\Psi(\mathbf{w}_0)\right] \ = \ \Psi(0) \ .\end{aligned}$$

■

The above theorem bounds the expected performance of SCD. We next give bounds that hold with high probability.

Theorem 2 *Assume that the conditions of Theorem 1 holds. Then, with probability of at least* $1/2$ *we have that*

$$P(\mathbf{w}_T) - P(\mathbf{w}^\star) \ \leq \ \frac{2d\,\Psi(0)}{T+1} \ .$$

Furthermore, for any $\delta \in (0,1)$*, suppose we run SCD* $r = \lceil \log_2(1/\delta) \rceil$ *times, each time T iterations, and let* $\mathbf{w}$ *be the best solution out of the r obtained solutions, then with probability of at least* $1-\delta$*,*

$$P(\mathbf{w}) - P(\mathbf{w}^\star) \leq \frac{2d\Psi(0)}{T+1} .$$

Proof The random variable $P(\mathbf{w}_T) - P(\mathbf{w}^\star)$ is non-negative and therefore the first inequality follows from Markov's inequality using Theorem 1. To prove the second result, note that the probability that on all r rounds it holds that $P(\mathbf{w}_T) - P(\mathbf{w}^\star) > \frac{2d\Psi(0)}{T+1}$ is at most $2^{-r} \leq \delta$, which concludes our proof. ■

Next, we specify the runtime bound for the case of ℓ_1 regularized logistic-regression and squared-loss. First, Lemma 6 in Appendix B shows that for C as defined in (5), if the second derivative of L is bounded by β then the condition on C given in Theorem 1 holds. Additionally, for the logistic-loss we have $C(0) \leq 1$. Therefore, for logistic-loss, after performing

$$\frac{d\,(\frac{1}{4}\,\|\mathbf{w}^\star\|_2^2 + 2)}{\varepsilon}$$

iterations of Algorithm 2 we achieve (expected) ε-accuracy in the objective P. Since the average cost of each iteration is sm, where s is as defined in (8), we end up with the total runtime

$$\frac{smd\,(\frac{1}{4}\,\|\mathbf{w}^\star\|_2^2 + 2)}{\varepsilon} .$$

The above is the runtime required to achieve expected ε-accuracy. Using Theorem 2 the required runtime to achieve ε-accuracy with a probability of at least $1-\delta$ is

$$smd\left(\frac{(\frac{1}{2}\,\|\mathbf{w}^\star\|_2^2 + 4)}{\varepsilon} + \lceil \log(1/\delta) \rceil\right) .$$

For the squared-loss we have $C(0) = \frac{1}{m}\sum_i y_i^2$. Assuming that the targets are normalized so that $C(0) \leq 1$, and using similar derivation we obtain the total runtime bound

$$smd\left(\frac{(2\|\mathbf{w}^\star\|_2^2 + 4)}{\varepsilon} + \lceil \log(1/\delta) \rceil\right) .$$

3. Stochastic Mirror Descent Made Sparse

In this section, we describe our mirror descent approach for ℓ_1 regularized loss minimization that maintains intermediate sparse solutions. Recall that we rewrite the problem in (1) using the notation

$$\min_{\mathbf{w}\in\mathbb{R}^d} \overbrace{\underbrace{\frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)}_{\equiv C(\mathbf{w})} + \lambda\|\mathbf{w}\|_1}^{\equiv P(\mathbf{w})} . \tag{11}$$

Mirror descent algorithms (Nemirovski and Yudin, 1978, Chapter 3) maintain two weight vectors: primal $\mathbf{w}$ and dual θ. The connection between the two vectors is via a link function $\theta = f(\mathbf{w})$, where $f : \mathbb{R}^d \to \mathbb{R}^d$. The link function is always taken to be the gradient map ∇F of some strictly convex function F and is therefore invertible. We can thus also write $\mathbf{w} = f^{-1}(\theta)$. In our mirror descent variant, we use the p-norm link function. That is, the jth element of f is

$$f_j(\mathbf{w}) \;=\; \frac{\operatorname{sign}(\mathbf{w}_j)\,|\mathbf{w}_j|^{q-1}}{\|\mathbf{w}\|_q^{q-2}} ,$$

where $\|\mathbf{w}\|_q = (\sum_j |w_j|^q)^{1/q}$. Note that f is simply the gradient of the function $\frac{1}{2}\|\mathbf{w}\|_q^2$. The inverse function is (see, e.g., Gentile, 2003)

$$f_j^{-1}(\theta) \;=\; \frac{\operatorname{sign}(\theta_j)\,|\theta_j|^{p-1}}{\|\theta\|_p^{p-2}} , \tag{12}$$

where $p = q/(q-1)$.

We first describe how mirror descent algorithms can be applied to the objective $C(\mathbf{w})$ without the ℓ_1 regularization term. At each iteration of the algorithm, we first sample a training example i uniformly at random from $\{1,\ldots,m\}$. We then estimate the gradient of $C(\mathbf{w})$ by calculating the vector $\mathbf{v} \;=\; L'(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)\,\mathbf{x}_i$. Note that the expectation of $\mathbf{v}$ over the random choice of i is $\mathbb{E}[\mathbf{v}] \;=\; \nabla C(\mathbf{w})$. That is, $\mathbf{v}$ is an unbiased estimator of the gradient of $C(\mathbf{w})$. Next, we update the dual vector according to $\theta = \theta - \eta\,\mathbf{v}$. If the link function is the identity mapping, this step is identical to the update of stochastic gradient descent. However, in our case f is not the identity function and it is important to distinguish between θ and $\mathbf{w}$. The above update of θ translates to an update of $\mathbf{w}$ by applying the link function $\mathbf{w} = f^{-1}(\theta)$. So far, we ignored the additional ℓ_1 regularization term. The simplest way to take this term into account is by also subtracting from θ the gradient of the term $\lambda\,\|\mathbf{w}\|_1$. (More precisely, since the ℓ_1 norm is not differentiable, we will use any subgradient of $\|\mathbf{w}\|_1$ instead, for example, the vector whose jth element is $\operatorname{sign}(w_j)$, where we interpret $\operatorname{sign}(0) = 0$.) Therefore, we could have redefined the update of θ to be $\theta_j = \theta_j - \eta(v_j + \lambda \operatorname{sign}(w_j))$. Unfortunately, as noted in Langford et al. (2009), this update leads to a dense vector θ, which in turn leads to a dense vector $\mathbf{w}$. The solution proposed in Langford et al. (2009) breaks the update into three phases. First, we let $\tilde{\theta} = \theta - \eta\,\mathbf{v}$. Second, we let $\hat{\theta} = \tilde{\theta} - \eta\,\lambda \operatorname{sign}(\tilde{\theta})$. Last, if in the second step we crossed the zero value, that is, $\operatorname{sign}(\hat{\theta}_j) \neq \operatorname{sign}(\tilde{\theta}_j)$, then we truncate the jth element to be zero. Intuitively, the goal of the first step is to decrease the value of $C(\mathbf{w})$ and this is done by a (mirror) gradient step, while the goal of the second and third steps is to decrease the value of $\lambda\,\|\mathbf{w}\|_1$. So, by truncating θ at zero we make the value of $\lambda\,\|\mathbf{w}\|_1$ even smaller.

3.1 Runtime Guarantee

We now provide runtime guarantees for Algorithm 3. We introduce two types of assumptions on the loss function:

$$|L'(a,y)| \quad\leq\quad \rho\,, \tag{13}$$
$$|L'(a,y)|^2 \quad\leq\quad \rho L(a,y)\,. \tag{14}$$

In the above, L' is the derivative w.r.t. the first argument and can also be a sub-gradient of L if L is not differentiable. It is easy to verify that (14) holds for the squared-loss with $\rho = 4$ and that (13)

Algorithm 3 Stochastic Mirror Descent Algorithm mAde Sparse (SMIDAS)

```
parameter: $\eta > 0$
let $p = 2\ln(d)$ and let $f^{-1}$ be as in (12)
let $\theta = 0, \mathbf{w} = 0$
for $t = 1, 2, \ldots$ do
  sample $i$ uniformly at random from $\{1, \ldots, m\}$
  let $\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i)\, \mathbf{x}_i$
  ($L'$ is the derivative of $L$. See, for example, (6))
  let $\tilde{\theta} = \theta - \eta\, \mathbf{v}$
  let $\forall j, \theta_j = \operatorname{sign}(\tilde{\theta}_j) \max\{0, |\tilde{\theta}_j| - \eta\lambda\}$
  let $\mathbf{w} = f^{-1}(\theta)$
end for
```

holds for the hinge-loss, $L(a,y) = \max\{0, 1-ya\}$, with $\rho = 1$. Interestingly, for the logistic-loss, both (13) holds with $\rho = 1$ and (14) holds with $\rho = 1/2$.

Theorem 3 *Let $\mathbf{w}^\star$ be a minimizer of* (11). *Suppose Algorithm 3 is run for $T-1$ iterations. Denote the value of $\mathbf{w}$ at the end of iteration t by $\mathbf{w}_t$ (with $\mathbf{w}_0 = 0$) and set $\mathbf{w}_o = \mathbf{w}_r$ for r chosen uniformly at random from $0, ..., T-1$.*

1. *If L satisfies* (13) *then,*

$$\mathbb{E}[P(\mathbf{w}_o)] - P(\mathbf{w}^\star) \le \tfrac{\eta(p-1)\rho^2 e}{2} + \tfrac{1}{\eta T}\|\mathbf{w}^\star\|_1^2 .$$

In particular, if we set

$$\eta = \frac{\|\mathbf{w}^\star\|_1}{\rho}\sqrt{\frac{2}{(p-1)\,e\,T}} ,$$

then we have,

$$\mathbb{E}[P(\mathbf{w}_o)] - P(\mathbf{w}^\star) \le \rho\,\|\mathbf{w}^\star\|_1 \sqrt{\tfrac{12\log(d)}{T}} .$$

2. *If L satisfies* (14) *then,*

$$\mathbb{E}[P(\mathbf{w}_o)] - P(\mathbf{w}^\star) \le \left(\frac{1}{1 - \frac{\eta\,(p-1)\,\rho\, e}{2}} - 1\right) P(0) + \frac{\|\mathbf{w}^\star\|_1^2}{\eta\, T\,(1 - \frac{\eta\,(p-1)\,\rho\, e}{2})} .$$

In particular, if we set

$$\eta = \frac{\|\mathbf{w}^\star\|_1^2}{P(0)\,T}\left(\sqrt{1 + \frac{2\,P(0)\,T}{(p-1)\,\rho\, e\,\|\mathbf{w}^\star\|_1^2}} - 1\right) ,$$

then we have,

$$\mathbb{E}[P(\mathbf{w}_o)] - P(\mathbf{w}^\star) \le 4\|\mathbf{w}^\star\|_1\sqrt{\frac{6\,\rho\,\log(d)\,P(0)}{2T}} + \frac{12\,\rho\,\log(d)\,\|\mathbf{w}^\star\|_1^2}{T} .$$

In both cases, the expectation is with respect to the algorithm's own randomization.

Proof We first give the proof for the case when (13) holds. Let θ_t be the value of θ at the beginning of iteration t of the algorithm, let $\mathbf{v}_t$ be the value of $\mathbf{v}$, and let $\tilde{\theta}_t = \theta_t - \eta \mathbf{v}_t$. Let $\mathbf{w}_t = f^{-1}(\theta_t)$ and $\tilde{\mathbf{w}}_t = f^{-1}(\tilde{\theta}_t)$ where f^{-1} is as defined in (12). Recall that $f(\mathbf{w}) = \nabla F(\mathbf{w})$ where $F(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_q^2$. Consider the Bregman divergence,

$$\begin{aligned}\Delta_F(\mathbf{w},\mathbf{w}') &= F(\mathbf{w}) - F(\mathbf{w}') - \langle \nabla F(\mathbf{w}'), \mathbf{w}-\mathbf{w}'\rangle \\ &= F(\mathbf{w}) - F(\mathbf{w}') - \langle f(\mathbf{w}'), \mathbf{w}-\mathbf{w}'\rangle ,\end{aligned}$$

and define the potential,

$$\Psi(\mathbf{w}) = \Delta_F(\mathbf{w}^\star, \mathbf{w}) .$$

We first rewrite the change in potential as

$$\Psi(\mathbf{w}_t) - \Psi(\mathbf{w}_{t+1}) = (\Psi(\mathbf{w}_t) - \Psi(\tilde{\mathbf{w}}_t)) + (\Psi(\tilde{\mathbf{w}}_t) - \Psi(\mathbf{w}_{t+1})) \ , \tag{15}$$

and bound each of the two summands separately.

Definitions of Δ_F, Ψ and simple algebra yield,

$$\begin{aligned}\Psi(\mathbf{w}_t) - \Psi(\tilde{\mathbf{w}}_t) &= \Delta_F(\mathbf{w}^\star,\mathbf{w}_t) - \Delta_F(\mathbf{w}^\star,\tilde{\mathbf{w}}_t) \\ &= F(\tilde{\mathbf{w}}_t) - F(\mathbf{w}_t) - \langle f(\mathbf{w}_t) - f(\tilde{\mathbf{w}}_t), \mathbf{w}^\star\rangle + \langle f(\mathbf{w}_t), \mathbf{w}_t\rangle - \langle f(\tilde{\mathbf{w}}_t), \tilde{\mathbf{w}}_t\rangle \\ &= \Delta_F(\tilde{\mathbf{w}}_t,\mathbf{w}_t) + \langle f(\mathbf{w}_t) - f(\tilde{\mathbf{w}}_t), \tilde{\mathbf{w}}_t - \mathbf{w}^\star\rangle \qquad (16) \\ &= \Delta_F(\tilde{\mathbf{w}}_t,\mathbf{w}_t) + \langle \theta_t - \tilde{\theta}_t, \tilde{\mathbf{w}}_t - \mathbf{w}^\star\rangle \\ &= \Delta_F(\tilde{\mathbf{w}}_t,\mathbf{w}_t) + \langle \eta\mathbf{v}_t, \tilde{\mathbf{w}}_t - \mathbf{w}^\star\rangle \\ &= \Delta_F(\tilde{\mathbf{w}}_t,\mathbf{w}_t) + \langle \eta\mathbf{v}_t, \mathbf{w}_t - \mathbf{w}^\star\rangle + \langle \eta\mathbf{v}_t, \tilde{\mathbf{w}}_t - \mathbf{w}_t\rangle \ . \qquad (17)\end{aligned}$$

By strong convexity of F with respect to the q-norm (see, e.g., Section A.4 of Shalev-Shwartz, 2007), we have

$$\Delta_F(\tilde{\mathbf{w}}_t,\mathbf{w}_t) \geq \tfrac{q-1}{2}\|\tilde{\mathbf{w}}_t - \mathbf{w}_t\|_q^2 \ .$$

Moreover, using Fenchel-Young inequality with the conjugate functions $g(\mathbf{x}) = \frac{q-1}{2}\|\mathbf{x}\|_q^2$ and $g^\star(\mathbf{x}) = \frac{1}{2(q-1)}\|\mathbf{x}\|_p^2$ we have

$$|\langle \eta\mathbf{v}_t, \tilde{\mathbf{w}}_t - \mathbf{w}^\star\rangle| \leq \tfrac{\eta^2}{2(q-1)}\|\mathbf{v}_t\|_p^2 + \tfrac{q-1}{2}\|\tilde{\mathbf{w}}_t - \mathbf{w}^\star\|_q^2 \ .$$

Plugging these into (17), we get

$$\begin{aligned}\Psi(\mathbf{w}_t) - \Psi(\tilde{\mathbf{w}}_t) &\geq \eta\langle \mathbf{v}_t, \mathbf{w}_t - \mathbf{w}^\star\rangle - \tfrac{\eta^2}{2(q-1)}\|\mathbf{v}_t\|_p^2 \\ &= \eta\langle \mathbf{v}_t, \mathbf{w}_t - \mathbf{w}^\star\rangle - \tfrac{\eta^2(p-1)}{2}\|\mathbf{v}_t\|_p^2 \ .\end{aligned}$$

By convexity of L, we have,

$$\langle \mathbf{v}_t, \mathbf{w}_t - \mathbf{w}^\star\rangle \geq L(\langle \mathbf{w}_t, \mathbf{x}_i\rangle, y_i) - L(\langle \mathbf{w}^\star, \mathbf{x}_i\rangle, y_i) \ ,$$

and therefore

$$\Psi(\mathbf{w}_t) - \Psi(\tilde{\mathbf{w}}_t) \geq \eta(L(\langle \mathbf{w}_t, \mathbf{x}_i\rangle, y_i) - L(\langle \mathbf{w}^\star, \mathbf{x}_i\rangle, y_i)) - \tfrac{\eta^2(p-1)}{2}\|\mathbf{v}_t\|_p^2 \ .$$

From (13), we obtain that

$$\|\mathbf{v}_t\|_p^2 \le \left(\|\mathbf{v}_t\|_\infty d^{1/p}\right)^2 \le \rho^2 d^{2/p} = \rho^2 e \ . \tag{18}$$

Thus,

$$\Psi(\mathbf{w}_t) - \Psi(\tilde{\mathbf{w}}_t) \ge \eta(L(\langle \mathbf{w}_t, \mathbf{x}_i \rangle, y_i) - L(\langle \mathbf{w}^\star, \mathbf{x}_i \rangle, y_i)) - \tfrac{\eta^2 (p-1)\rho^2 e}{2} \ . \tag{19}$$

So far, our analysis has followed the standard analysis of mirror descent (see, e.g., Beck and Teboulle, 2003). It is a bit more tricky to show that

$$\Psi(\tilde{\mathbf{w}}_t) - \Psi(\mathbf{w}_{t+1}) \ge \eta\lambda(\|\mathbf{w}_{t+1}\|_1 - \|\mathbf{w}^\star\|_1) \ . \tag{20}$$

To show this, we begin the same way as we did to obtain (16),

$$\begin{aligned}\Psi(\tilde{\mathbf{w}}_t) - \Psi(\mathbf{w}_{t+1}) &= \Delta_F(\mathbf{w}_{t+1}, \tilde{\mathbf{w}}_t) + \langle f(\tilde{\mathbf{w}}_t) - f(\mathbf{w}_{t+1}), \mathbf{w}_{t+1} - \mathbf{w}^\star \rangle \\ &= \Delta_F(\mathbf{w}_{t+1}, \tilde{\mathbf{w}}_t) + \langle \tilde{\theta}_t - \theta_{t+1}, \mathbf{w}_{t+1} - \mathbf{w}^\star \rangle \\ &\ge \langle \tilde{\theta}_t - \theta_{t+1}, \mathbf{w}_{t+1} - \mathbf{w}^\star \rangle \\ &= \langle \tilde{\theta}_t - \theta_{t+1}, \mathbf{w}_{t+1} \rangle - \langle \tilde{\theta}_t - \theta_{t+1}, \mathbf{w}^\star \rangle \ . \end{aligned} \tag{21}$$

Note that $\mathrm{sign}(w_{t+1,j}) = \mathrm{sign}(\theta_{t+1,j})$. Moreover, when $\theta_{t+1,j} \ne 0$ then,

$$\tilde{\theta}_{t,j} - \theta_{t+1,j} = \eta\lambda\, \mathrm{sign}(\theta_{t+1,j}) \ .$$

Thus, we have,

$$\begin{aligned}\langle \tilde{\theta}_t - \theta_{t+1}, \mathbf{w}_{t+1} \rangle &= \sum_{j: w_{t+1,j} \ne 0} (\tilde{\theta}_{t,j} - \theta_{t+1,j}) w_{t+1,j} \\ &= \sum_{j: w_{t+1,j} \ne 0} \eta\lambda\, \mathrm{sign}(\theta_{t+1,j}) w_{t+1,j} \\ &= \eta\lambda \sum_{j: w_{t+1,j} \ne 0} \mathrm{sign}(w_{t+1,j}) w_{t+1,j} \\ &= \eta\lambda \|\mathbf{w}_{t+1}\|_1 \ . \end{aligned}$$

Note that this equality is crucial and does not hold for the Bregman potential corresponding to the exponentiated gradient algorithm. Plugging the above equality, along with the inequality,

$$|\langle \tilde{\theta}_t - \theta_{t+1}, \mathbf{w}^\star \rangle| \le \|\tilde{\theta}_t - \theta_{t+1}\|_\infty \|\mathbf{w}^\star\|_1 = \eta\,\lambda\, \|\mathbf{w}^\star\|_1$$

into (21), we get (20).

Combining the lower bounds (19) and (20) and plugging them into (15), we get,

$$\begin{aligned}\Psi(\mathbf{w}_t) - \Psi(\mathbf{w}_{t+1}) \ge\ & \eta(L(\langle \mathbf{w}_t, \mathbf{x}_i \rangle, y_i) - L(\langle \mathbf{w}^\star, \mathbf{x}_i \rangle, y_i)) \\ & - \tfrac{\eta^2 (p-1)\rho^2 e}{2} + \eta\lambda(\|\mathbf{w}_{t+1}\|_1 - \|\mathbf{w}^\star\|_1) \ . \end{aligned}$$

Taking expectation with respect to i drawn uniformly at random from $\{1, \ldots, m\}$, we get,

$$\begin{aligned}\mathbb{E}[\Psi(\mathbf{w}_t) - \Psi(\mathbf{w}_{t+1})] &\ge \eta\mathbb{E}[C(\mathbf{w}_t) - C(\mathbf{w}^\star)] - \tfrac{\eta^2 (p-1)\rho^2 e}{2} + \eta\lambda\mathbb{E}[\|\mathbf{w}_{t+1}\|_1 - \|\mathbf{w}^\star\|_1] \\ &= \eta\mathbb{E}[P(\mathbf{w}_t) - P(\mathbf{w}^\star)] - \tfrac{\eta^2 (p-1)\rho^2 e}{2} + \eta\lambda\mathbb{E}[\|\mathbf{w}_{t+1}\|_1 - \|\mathbf{w}_t\|_1] \ . \end{aligned}$$

Summing over $t = 0, \ldots, T-1$, dividing by ηT, and rearranging gives,

$$\begin{aligned}
\tfrac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[P(\mathbf{w}_t)] - P(\mathbf{w}^\star) &\le \tfrac{\eta(p-1)\rho^2 e}{2} + \tfrac{\lambda}{T}\,\mathbb{E}\left[\|\mathbf{w}_0\|_1 - \|\mathbf{w}_T\|_1\right] + \tfrac{1}{\eta T}\,\mathbb{E}\left[\Psi(\mathbf{w}_0) - \Psi(\mathbf{w}_T)\right] \\
&\le \tfrac{\eta(p-1)\rho^2 e}{2} + 0 + \tfrac{1}{\eta T}\Delta_F(\mathbf{w}^\star, 0) \\
&= \tfrac{\eta(p-1)\rho^2 e}{2} + \tfrac{1}{\eta T}\|\mathbf{w}^\star\|_q^2 \\
&\le \tfrac{\eta(p-1)\rho^2 e}{2} + \tfrac{1}{\eta T}\|\mathbf{w}^\star\|_1^2 \,. \qquad (22)
\end{aligned}$$

Now, optimizing over η gives

$$\tfrac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[P(\mathbf{w}_t)] - P(\mathbf{w}^\star) \le \rho\|\mathbf{w}^\star\|_1\sqrt{\frac{2(p-1)e}{T}}$$

and this concludes our proof for the case when (13) holds, since for a random r we have $\mathbb{E}[P(\mathbf{w}_r)] = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[P(\mathbf{w}_t)]$.

When (14) holds, instead of the bound (18), we have,

$$\|\mathbf{v}_t\|_p^2 \le \left(\|\mathbf{v}_t\|_\infty d^{1/p}\right)^2 \le \rho L(\langle \mathbf{w}_t, \mathbf{x}_i\rangle, y_i)\, d^{2/p} = \rho L(\langle \mathbf{w}_t, \mathbf{x}_i\rangle, y_i)\, e \,.$$

As a result, the final bound (22) now becomes,

$$\begin{aligned}
\tfrac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[P(\mathbf{w}_t)] - P(\mathbf{w}^\star) &\le \tfrac{\eta(p-1)\rho e}{2T}\sum_{t=0}^{T-1}\mathbb{E}[C(\mathbf{w}_t)] + \tfrac{1}{\eta T}\|\mathbf{w}^\star\|_1^2 \\
&\le \tfrac{\eta(p-1)\rho e}{2T}\sum_{t=0}^{T-1}\mathbb{E}[P(\mathbf{w}_t)] + \tfrac{1}{\eta T}\|\mathbf{w}^\star\|_1^2 \,.
\end{aligned}$$

For the sake of brevity, let $a = (p-1)\rho e/2$ and $b = \|\mathbf{w}^\star\|_1^2$, so that the above bound can be written as,

$$\begin{aligned}
\tfrac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[P(\mathbf{w}_t)] - P(\mathbf{w}^\star) &\le \left(\frac{1}{1-a\eta} - 1\right)P(\mathbf{w}^\star) + \frac{b/(\eta T)}{1-a\eta} \qquad (23) \\
&\le \left(\frac{1}{1-a\eta} - 1\right)P(0) + \frac{b/(\eta T)}{1-a\eta} \,.
\end{aligned}$$

At this stage, we need to minimize the expression on the right hand side as a function of η. This somewhat tedious but straightforward minimization is done in Lemma 7 in Appendix B. Using Lemma 7 (with $P = P(0)$), we see that the right hand side is minimized by setting

$$\eta = \frac{\|\mathbf{w}^\star\|_1^2}{P(0)\,T}\left(\sqrt{1 + \frac{2P(0)\,T}{(p-1)\,\rho\, e\,\|\mathbf{w}^\star\|_1^2}} - 1\right) ,$$

and the minimum value is upper bounded by

$$4\|\mathbf{w}^\star\|_1\sqrt{\frac{(p-1)\,\rho\, e P(0)}{2T}} + \frac{2\,(p-1)\,\rho\, e\,\|\mathbf{w}^\star\|_1^2}{T} \,.$$

This concludes the proof for the case when (14) holds. ■

The bound in the above theorem can be improved if (14) holds and the desired accuracy is the same order as $P(\mathbf{w}^\star)$. This is the content of the next proposition.

Proposition 4 *Let* $\mathbf{w}^\star$ *be a minimizer of* (11). *Suppose Algorithm 3 is run for* $T-1$ *iterations. Denote the value of* $\mathbf{w}$ *at the beginning of iteration* t *by* $\mathbf{w}_t$ *and set* $\mathbf{w}_o = \mathbf{w}_r$ *for* r *chosen uniformly at random from* $0,\dots,T-1$. *If* L *satisfies* (14) *and we set*

$$\eta = \frac{2}{(p-1)\rho e}\cdot\frac{K}{1+K} \ ,$$

for some arbitrary $K>0$*, then we have,*

$$\mathbb{E}[P(\mathbf{w}_o)] \le (1+K)P(\mathbf{w}^\star) + \frac{(1+K)^2}{K}\cdot\frac{3\rho\log(d)\,\|\mathbf{w}^\star\|_1^2}{T} \ .$$

Proof Plugging $\eta = K/a(1+K)$ in (23) gives,

$$\tfrac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[P(\mathbf{w}_t)] - P(\mathbf{w}^\star) \le K\,P(\mathbf{w}^\star) + \frac{(1+K)^2}{K}\cdot\frac{ab}{T} \ .$$

Recalling that $p = 2\log(d)$, $a = (p-1)\rho e/2$ and $b = \|\mathbf{w}^\star\|_1^2$ concludes our proof. ■

4. Experiments

In this section, we provide experimental results for our algorithms on 4 data sets. We begin with a description of the data sets following by a description of the algorithms we ran on them.

4.1 Data Sets

We consider 4 binary classification data sets for our experiments: DUKE, ARCENE, MAGIC04S, and MAGIC04D.

DUKE is a breast cancer data set from West et al. (2001). It has 44 examples with 7,129 features with a density level of 100%. ARCENE is a data set from the UCI Machine Learning repository where the task is to distinguish cancer patterns from normal ones based on 10,000 mass-spectrometric features. Out of these, 3,000 features are synthetic features as this data set was designed for the NIPS 2003 variable selection workshop. There are 100 examples in this data set and the example matrix contains 5.4×10^5 non-zero entries corresponding to a density level of 54%. The data sets MAGIC04S and MAGIC04D were obtained by adding 1,000 random features to the MAGIC Gamma Telescope data set from the UCI Machine Learning repository. The original data set has 19,020 examples with 10 features. This is also a binary classification data set and the task is to distinguish high-energy gamma particles from background using a gamma telescope. Following the experimental setup of Langford et al. (2009), we added 1,000 random features, each of which takes value 0 with probability 0.95 or 1 with probability 0.05, to create a sparse data set, MAGIC04S. We also created a dense data set, MAGIC04D, in which the random features took value -1 or $+1$, each with probability 0.5. MAGIC04S and MAGIC04D have density levels of 5.81% and 100% respectively.

4.2 Algorithms

We ran 4 algorithms on these data sets: SCD, GCD, SMIDAS, and TRUNCGRAD. SCD is the stochastic coordinate descent algorithm given in Section 2 above. GCD is the corresponding deterministic and "greedy" version of the same algorithm. The coordinate to be updated at each iteration is *greedily* chosen in a deterministic manner to maximize a lower bound on the guaranteed decrease in the objective function. This type of deterministic criterion for choosing features is common in Boosting approaches. Since choosing a coordinate (or feature in our case) in a deterministic manner involves significant computation in case of large data sets, we expect that the deterministic algorithm will converge much slower than the stochastic algorithm. We also tried the *cyclic* version of coordinate descent that just cycles through the coordinates. We found its performance to be indistinguishable from that of SCD and hence we do not report it here. SMIDAS is the mirror descent algorithm given in Section 3 above. TRUNCGRAD is the truncated gradient algorithm of Langford et al. (2009) (In fact, Langford et al., 2009 suggests another way to truncate the gradient. Here, we refer to the variant corresponding to SMIDAS.) Of these 4, the first two are parameter-free algorithms while the latter two require a parameter η. In our experiments, we ran SMIDAS and TRUNCGRAD for a range of different values of η and chose the one that yielded the minimum value of the objective function (i.e., the regularized loss). We chose to minimize the (regularized) logistic loss in all our experiments.

4.3 Results

For each data set, we show two plots. One plot shows the regularized objective function plotted against the number of *data accesses*, that is, the number of times the algorithm accesses the data matrix $(x_{i,j})$. We choose to use this as opposed to, say CPU time, as this is an implementation independent quantity. Moreover, the actual time taken by these algorithms will be roughly proportional to this quantity provided computing features is time consuming. The second plot shows the density (or ℓ_0-norm, the number of non-zero entries) of the iterate plotted against the number of data accesses. In the next subsection, we use mild regularization ($\lambda = 10^{-6}$). Later on, we will show results for stronger regularization ($\lambda = 10^{-2}$).

4.3.1 LESS REGULARIZATION

Figure 1 is for the DUKE data set. It is clear that GCD does much worse than the other three algorithms. GCD is much slower because, as we mentioned above, it spends a lot of time in finding the best coordinate to update. The two algorithms having a tunable parameter η have roughly the same performance as SCD. However, SCD has a definite edge if we add up the time to perform several runs of these algorithms for tuning η. Note, however, that SMIDAS has better sparsity properties as compared to TRUNCGRAD and SCD even though their performance measured in the objective is similar.

Figure 2 is for the ARCENE data set. The results are quite similar to those for the DUKE data set. SMIDAS is slow for a short while early on but quickly catches up. Again, it displays good sparsity properties.

For the MAGIC data sets, SMIDAS does much better than TRUNCGRAD for the MAGIC04D data set (where the example vectors are dense). TRUNCGRAD is slightly better for the MAGIC04S data set (where the example vectors are sparse). This is illustrated in Figure 3. Note that this behavior is consistent with the bounds (3) and (4) given above. These bounds suggest that if the

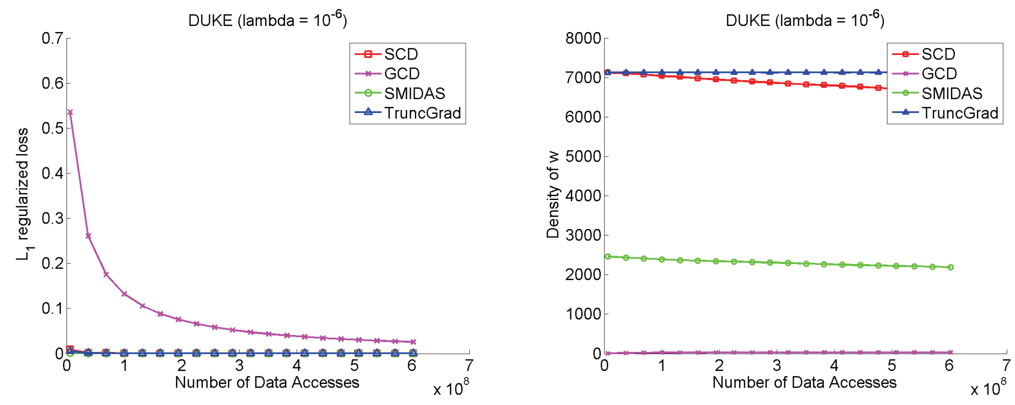


Figure 1: DUKE data set; less regularization

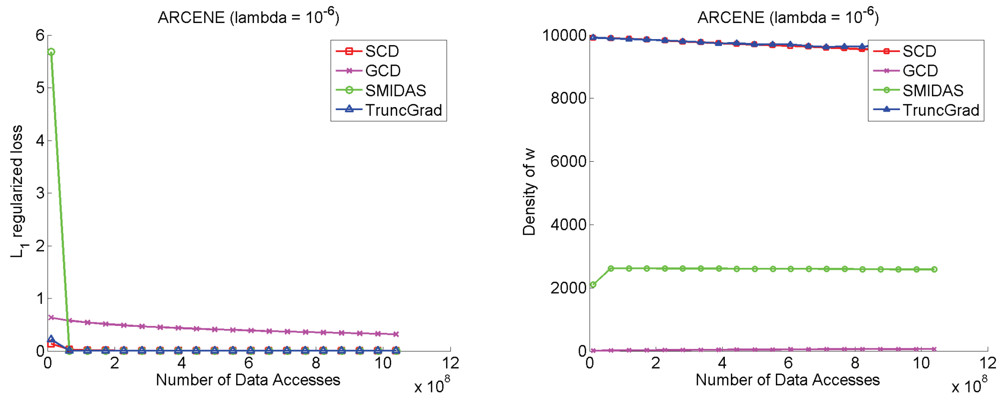


Figure 2: ARCENE data set; less regularization

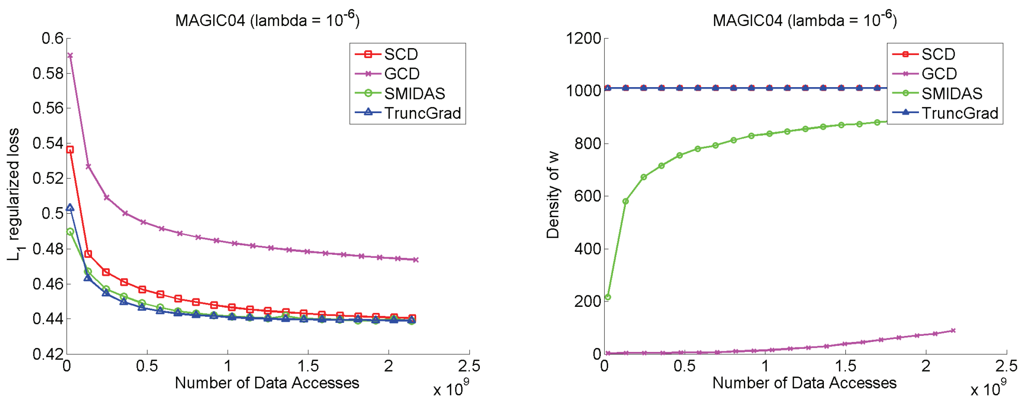


Figure 3: MAGIC04S data set; less regularization

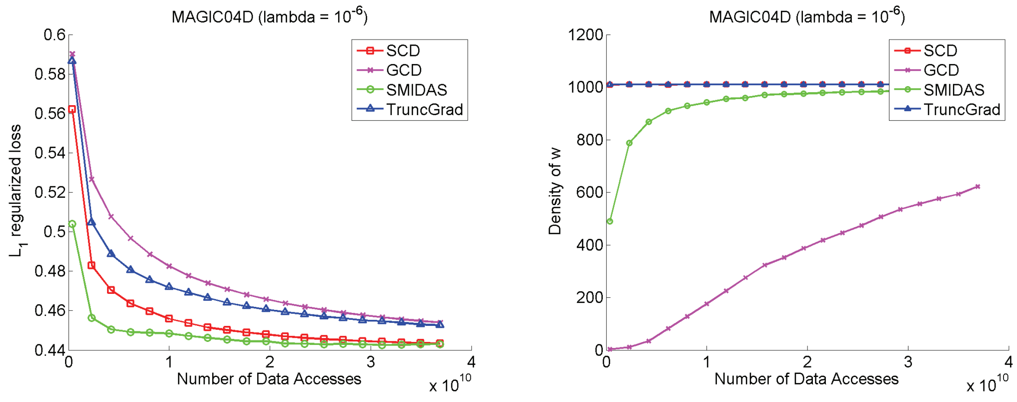


Figure 4: MAGIC04D data set; less regularization

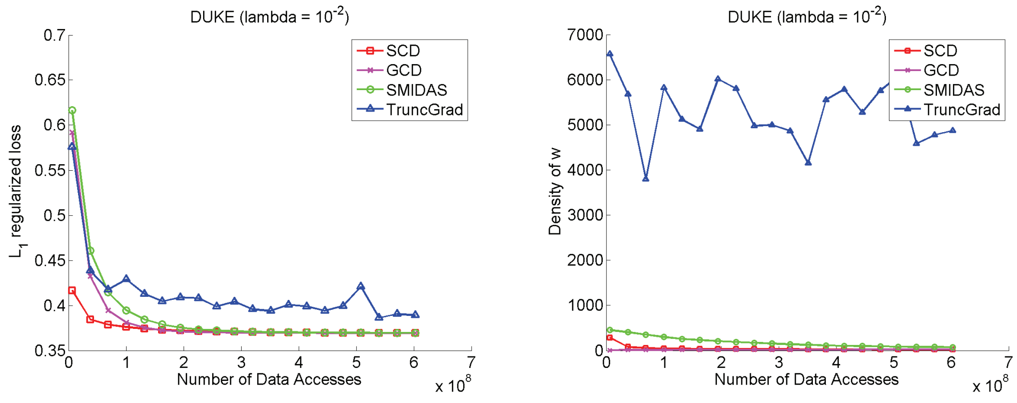


Figure 5: DUKE data set; more regularization

true solution has low ℓ_1 norm, SMIDAS will require fewer iterations than TRUNCGRAD when the examples are dense. The parameter-free SCD algorithm is slightly worse than the parameter-based algorithms TRUNCGRAD and SMIDAS on MAGIC04S. For MAGIC04D, its performance is better than TRUNCGRAD, but slightly worse than SMIDAS. On both data sets, SMIDAS seems to be doing quite well in terms of sparsity.

4.3.2 MORE REGULARIZATION

As mentioned before, we now present results with a large value of the regularization parameter ($\lambda = 10^{-2}$).

From Figures 5 and 6, we see that SCD outperforms all other algorithms on the DUKE and ARCENE data sets. Not only does it get to the minimum objective faster, but it also gets best sparsity. On both data sets, SMIDAS does better than TRUNCGRAD.

For the MAGIC data sets (Figures 7 and 8), we see a previous phenomenon repeated: SMIDAS does better when the features are dense. The coordinate descent algorithms SCD and GCD are quicker to reach the minimum on MAGIC04S. The edge, however, seems to be lost on the

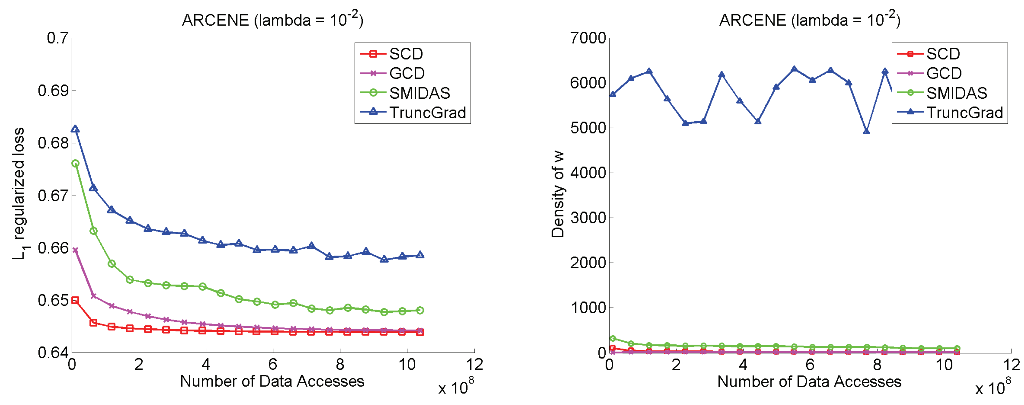


Figure 6: ARCENE data set; more regularization

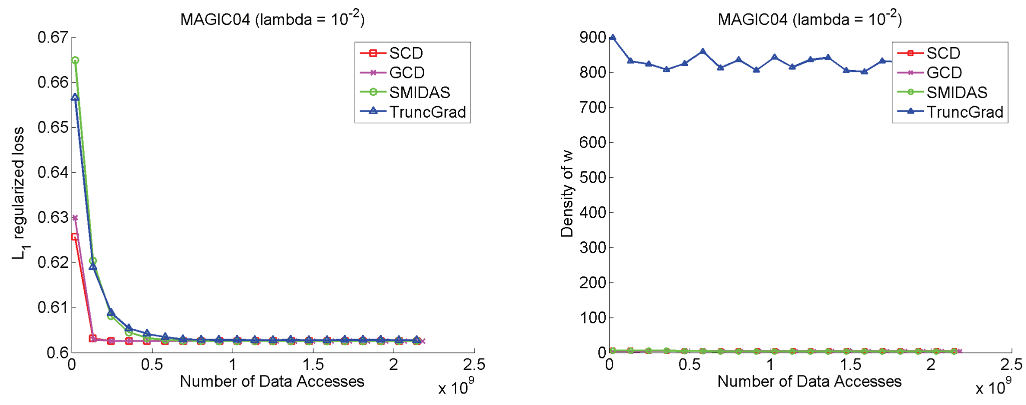


Figure 7: MAGIC04S data set; more regularization

MAGIC04D data set. In all cases, TRUNCGRAD seems to be unable to keep sparsity of the iterates as it progresses. This effect is particularly stark in Figure 8, where all the other algorithms have density levels of a few tens while TRUNCGRAD has almost no sparsity whatsoever.

4.3.3 PARAMETER VALUES AND SOURCE CODE

Two of the algorithms we used above, namely TRUNCGRAD and SMIDAS, have a step-size parameter η. In the interest of reproducible research, we report the values of η used in our experiments in Table 1. The parameter p of SMIDAS was always $\lceil 2\ln(d) \rceil$, where d is the total number of features (including non-relevant features). The source code for a C++ implementation of SCD and SMIDAS can be found at `http://mloss.org` (by searching for either "SCD" or "SMIDAS").

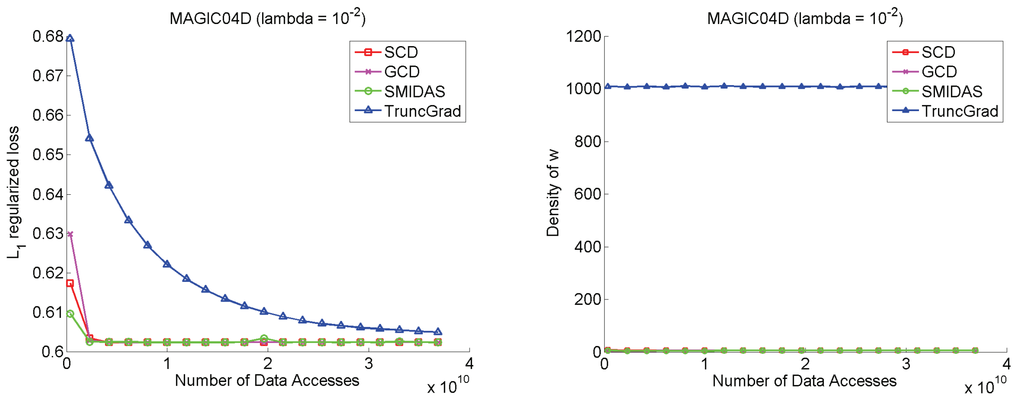


Figure 8: MAGIC04D data set; more regularization

Data Set, λ	SMIDAS	TRUNCGRAD
DUKE, 10^{-6}	50	10^{-1}
DUKE, 10^{-2}	10^{-1}	10^{-2}
ARCENE, 10^{-6}	50	10^{-1}
ARCENE, 10^{-2}	10^{-2}	5×10^{-5}
MAGIC04S, 10^{-6}	10^{-2}	5×10^{-4}
MAGIC04S, 10^{-2}	10^{-4}	5×10^{-5}
MAGIC04D, 10^{-6}	5×10^{-3}	10^{-4}
MAGIC04D, 10^{-2}	10^{-3}	10^{-5}

Table 1: Values of the step-size parameter η used in the experiments for SMIDAS and TRUNCGRAD

Acknowledgments

We thank Jonathan Chang for pointing out some errors in the preliminary conference version of this work. Most of the research reported in this paper was done while the authors were at the Toyota Technological Institute at Chicago (TTIC). We are grateful to TTIC for providing a friendly and stimulating environment to work in.

Appendix A.

Lemma 5 *Let* $\varepsilon \le 0.12$. *There exists an optimization problem of the form:*

$$\min_{\mathbf{w}\in\mathbb{R}^d \,:\, \|\mathbf{w}\|_1 \le B} \quad \frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i) \ ,$$

where L *is the smooth loss function* $L(a,b) = (a-b)^2$, *such that any algorithm which initializes* $\mathbf{w} = 0$ *and updates a single element of the vector* $\mathbf{w}$ *at each iteration, must perform at least* $B^2/16\varepsilon$ *iterations to achieve an* ε *accurate solution.*

Proof We denote the number of non-zeros elements of a vector $\mathbf{w}$ by $\|\mathbf{w}\|_0$. Recall that we denote the average loss by $C(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^m L(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)$. We show an optimization problem of the form given above, for which the optimal solution, denoted $\mathbf{w}^\star$, is dense (i.e., $\|\mathbf{w}^\star\|_0 = d$), while any $\mathbf{w}$ for which

$$C(\mathbf{w}) \le C(\mathbf{w}^\star) + \varepsilon$$

must satisfy $\|\mathbf{w}\|_0 \ge \Omega(B^2/\varepsilon)$. This implies the statement given in the lemma since an iteration bound for the type of algorithms we consider is immediately translated into an upper bound on $\|\mathbf{w}\|_0$.

Let $L(a,b) = (a-b)^2$ and consider the following joint distribution over random variables (X,Y). First, each Y is chosen at random according to $\mathbb{P}[Y=1] = \mathbb{P}[Y=-1] = \frac{1}{2}$. Next, each element j of X is chosen i.i.d. from $\{+1,-1\}$ according to $\mathbb{P}[X_j = y|y] = \frac{1}{2} + \frac{1}{2B}$. This definition implies that:

$$\mathbb{E}_{X_j|Y=y}[X_j] = \tfrac{1}{B}\, y$$

and

$$\mathrm{Var}_{X_j|Y=y}[X_j] = 1 - \tfrac{1}{B^2} \ .$$

Consider the vector $\mathbf{w}_0 = (\frac{B}{d}, \ldots, \frac{B}{d})$. We have

$$\begin{aligned}
\mathbb{E}\left[(\langle \mathbf{w}_0, X\rangle - Y)^2\right] &= \mathbb{E}_Y \mathbb{E}_{X|Y=y}\left[(\langle \mathbf{w}_0, X\rangle - y)^2\right] \\
&= \mathbb{E}_Y \mathrm{Var}_{X|Y=y}\left[\langle \mathbf{w}_0, X\rangle\right] \\
&= \mathbb{E}_Y \frac{B^2}{d} \mathrm{Var}_{X_1|Y=y}\left[X_1\right] \\
&= \mathbb{E}_Y \frac{B^2}{d}\left(1 - \tfrac{1}{B^2}\right) \\
&= \frac{B^2-1}{d} \ .
\end{aligned}$$

Now fix some $\mathbf{w}$ with $\|\mathbf{w}\|_0 \le s$. We have

$$\mu_y := \mathbb{E}_{X|Y=y}[\langle \mathbf{w}, X\rangle] = \tfrac{y}{B} \sum_j w_j \ ,$$

and

$$\begin{aligned}\mathbb{E}\left[(\langle \mathbf{w}, X\rangle - Y)^2\right] &= \mathbb{E}_Y \mathbb{E}_{X|Y=y}\left[(\langle \mathbf{w}, X\rangle - y)^2\right] \\ &= \mathbb{E}_Y \mathrm{Var}_{X|Y=y}[\langle \mathbf{w}, X\rangle] + (\mu_y - y)^2\end{aligned}$$

If $|\sum_j w_j| \le B/2$ then $(\mu_y - y)^2 \ge 1/4$ and thus we obtain from the above that $\mathbb{E}\left[(\langle \mathbf{w}, X\rangle - Y)^2\right] \ge 1/4$. Otherwise,

$$\sqrt{s \sum_j w_j^2} \ge \sum_j |w_j| \ge |\sum_j w_j| \ge B/2 \ ,$$

and thus we have that

$$\begin{aligned}\mathbb{E}[(\langle \mathbf{w}, X\rangle - Y)^2] &\ge \mathbb{E}_Y \mathrm{Var}_{X|Y=y}[\langle \mathbf{w}, X\rangle] \\ &= \mathbb{E}_Y \sum_{j=1}^d w_j^2 \mathrm{Var}_{X_1|Y=y}[X_1] \\ &= \mathbb{E}_Y \sum_{j=1}^d w_j^2 \left(1 - \tfrac{1}{B^2}\right) \\ &= \left(1 - \tfrac{1}{B^2}\right) \sum_{j=1}^d w_j^2 \\ &\ge \left(1 - \tfrac{1}{B^2}\right) \tfrac{B^2}{4s} = \frac{B^2 - 1}{4s} \ .\end{aligned}$$

Choose $B \ge 2$ and $d = 100(B^2 - 1)$, we have shown that if $\|\mathbf{w}\|_0 \le s$ then

$$\mathbb{E}[(\langle \mathbf{w}, X\rangle - Y)^2 - (\langle \mathbf{w}_0, X\rangle - Y)^2] \ge \min\left\{0.24, \tfrac{B^2}{8s}\right\} =: \varepsilon' \ .$$

Now, consider the random variable $Z = (\langle \mathbf{w}, X\rangle - Y)^2 - (\langle \mathbf{w}_0, X\rangle - Y)^2$. This is a bounded random variable (because $|\langle \mathbf{w}, \mathbf{x}\rangle| \le B$) and therefore using Hoeffding inequality we have that with probability of at least $1 - \delta$ over a draw of a training set of m examples we have

$$C(\mathbf{w}) - C(\mathbf{w}_0) \ge \varepsilon' - cB^2 \sqrt{\frac{\log(1/\delta)}{m}} \ ,$$

for some universal constant $c > 0$.

This is true if we first fix $\mathbf{w}$ and then draw the m samples. We want to establish an inequality true for any $\mathbf{w}$ in

$$\mathcal{W} := \{\mathbf{w} \in \mathbb{R}^d \ : \ \|\mathbf{w}\|_0 \le s, \ \|\mathbf{w}\|_1 \le B\} \ .$$

This set has infinitely many elements so we cannot trivially appeal to a union bound. Instead, we create an $\varepsilon'/16B$-cover of $\mathcal{W}$ in the ℓ_1 metric. This has size $\mathcal{N}_1(\mathcal{W}, \varepsilon'/16B)$ where we have the crude estimate,

$$\forall \varepsilon > 0, \ \mathcal{N}_1(\mathcal{W}, \varepsilon) \le d^s \left(\frac{2Bd}{\varepsilon}\right)^s \ .$$

Moreover, if $\|\mathbf{w}-\mathbf{w}'\|_1 \leq \varepsilon'/16B$ then it is easy to see that $|C(\mathbf{w})-C(\mathbf{w}')| \leq \varepsilon'/4$. Therefore, applying a union bound over all vectors in the cover, we obtain that with probability at least $1-\delta$, for *all* such $\mathbf{w} \in \mathcal{W}$ we have

$$C(\mathbf{w})-C(\mathbf{w}_0) \geq \varepsilon' - \varepsilon'/4 - cB^2\sqrt{\frac{\log \mathcal{N}_1(\mathcal{W},\varepsilon'/16B)+\log(1/\delta)}{m}} \ .$$

Taking m large enough, we can guarantee that, with high probability, for all $\mathbf{w} \in \mathcal{W}$,

$$C(\mathbf{w})-C(\mathbf{w}_0) \geq \varepsilon'/2 \ .$$

Finally, we clearly have that $C(\mathbf{w}^\star) \leq C(\mathbf{w}_0)$.

Thus, we have proved the following. Given $B \geq 2$ and s, there exist $\{(\mathbf{x}_i,y_i)\}_{i=1}^m$ in some dimension d, such that

$$\min_{\|\mathbf{w}\|_1 \leq B, \|\mathbf{w}\|_0 \leq s} C(\mathbf{w}) - \min_{\|\mathbf{w}\|_1 \leq B} C(\mathbf{w}) \geq \min\left\{0.12, \tfrac{B^2}{16s}\right\} \ .$$

This concludes the proof of the lemma. ■

Appendix B.

Lemma 6 *Let C be as defined in* (5) *and assume that the second derivative of L with respect to its first argument is bounded by* β. *Then, for any* $j \in [d]$,

$$C(\mathbf{w}+\eta\mathbf{e}^j) \leq C(\mathbf{w}) + \eta(\nabla C(\mathbf{w}))_j + \tfrac{\beta\eta^2}{2} \ .$$

Proof Note that, by assumption on L, for any i,j we have,

$$\begin{aligned} L(\langle \mathbf{w}+\eta\mathbf{e}^j,\mathbf{x}_i\rangle,y_i) &= L(\langle \mathbf{w},\mathbf{x}_i\rangle + \eta x_{i,j},y_i) \\ &\leq L(\langle \mathbf{w},\mathbf{x}_i\rangle,y_i) + \eta L'(\langle \mathbf{w},\mathbf{x}_i\rangle,y_i)x_{i,j} + \tfrac{\beta\eta^2 x_{i,j}^2}{2} \\ &\leq L(\langle \mathbf{w},\mathbf{x}_i\rangle,y_i) + \eta L'(\langle \mathbf{w},\mathbf{x}_i\rangle,y_i)x_{i,j} + \tfrac{\beta\eta^2}{2} \ , \end{aligned}$$

where the last inequality follows because $x_{i,j} \in [-1,+1]$. Adding the above inequalities for $i = 1,\ldots,m$ and dividing by m, we get

$$\begin{aligned} C(\mathbf{w}+\eta\mathbf{e}^j) &\leq C(\mathbf{w}) + \frac{\eta}{m}\sum_{i=1}^m L'(\langle \mathbf{w},\mathbf{x}_i\rangle,y_i)x_{i,j} + \tfrac{\beta\eta^2}{2} \\ &= C(\mathbf{w}) + \eta(\nabla C(\mathbf{w}))_j + \tfrac{\beta\eta^2}{2} \ . \end{aligned}$$

■

Lemma 7 *Let* $a,b,P,T > 0$. *The function* $f : (0,1/a) \to \mathbb{R}$ *defined as,*

$$f(\eta) = \left(\frac{1}{1-a\eta} - 1\right)P + \frac{b/(\eta T)}{1-a\eta}$$

is minimized at

$$\eta^\star = \frac{b}{PT}\left(\sqrt{1+\frac{PT}{ab}}-1\right) ,$$

and the minimum value satisfies

$$f(\eta^\star) \le 4\sqrt{\frac{abP}{T}}+\frac{4ab}{T} .$$

Proof A little rearranging gives,

$$f(\eta) = \frac{1}{\frac{1}{\eta}-a}\left(aP+\frac{b}{\eta^2 T}\right) .$$

This suggests the change of variable $C = 1/\eta$ and we wish to minimize $g : (a,\infty) \to \mathbb{R}$ defined as,

$$g(C) = \frac{1}{C-a}\left(aP+\frac{bC^2}{T}\right) .$$

The expression for the derivative g' is,

$$g'(C) = \frac{b}{T(C-a)^2}\left(C^2-2aC-\frac{aTP}{b}\right) .$$

Setting $g'(C) = 0$ gives a quadratic equation whose roots are,

$$a \pm \sqrt{a^2+\frac{aTP}{b}} .$$

Choosing the larger root (the smaller one is smaller than a) gives us the minimizer,

$$C^\star = a+\sqrt{a^2+\frac{aTP}{b}} .$$

It is easy to see that $g'(C)$ is increasing at $C^\star$ and thus we have a local minima at $C^\star$ (which is also global in this case). The minimizer $\eta^\star$ of $f(\eta)$ is therefore,

$$\eta^\star = \frac{1}{C^\star} = \frac{b}{PT}\left(\sqrt{1+\frac{PT}{ab}}-1\right) .$$

Plugging in the value of $C^\star$ into $g(C)$, we get,

$$\begin{aligned}
g(C^\star) &= \frac{2}{\sqrt{1+\frac{PT}{ab}}}\left(P+\frac{ab}{T}+\sqrt{\frac{a^2b^2}{T^2}+\frac{abP}{T}}\right) \\
&\le \frac{2}{\sqrt{1+\frac{PT}{ab}}}\left(2P+\frac{2ab}{T}\right) \\
&= 4\sqrt{\frac{abP}{T}+\frac{a^2b^2}{T^2}} \\
&\le 4\sqrt{\frac{abP}{T}}+\frac{4ab}{T} .
\end{aligned}$$

Since $g(C^\star) = f(\eta^\star)$, this concludes the proof of the lemma. ■

References

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.

L. Bottou. Stochastic gradient descent examples, Web Page. `http://leon.bottou.org/projects/sgd`.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 20*, pages 161–168, 2008.

L. Bottou and Y. LeCunn. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

K.L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 922–931, 2008.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *International Conference on Machine Learning*, pages 272–279, 2008.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In *Proceedings of the 23rd Annual Conference on Learning Theory*, pages 14–26, 2010.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32 (2):407–499, 2004.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

J. Friedman, T. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

A. Genkin, D. Lewis, and D. Madigan. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.

S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, 2011. available at `http://www.ise.ufl.edu/glan/papers/strongSCOSubmit.pdf`.

A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

K. Koh, S.J. Kim, and S. Boyd. An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, pages 1–33, 2010.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. In *Advances in Neural Information Processing Systems 21*, pages 905–912, 2009.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

Z. Q. Luo and P. Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72:7–35, 1992.

A. Nemirovski and D. Yudin. *Problem complexity and method efficiency in optimization*. Nauka Publishers, Moscow, 1978.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. Technical Report CORE Discussion Paper 2010/02, Center for Operations Research and Econometrics, UCL, Belgium, 2010.

S. Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University, 2007.

S. Shalev-Shwartz and N. Srebro. SVM optimization: Inverse dependence on training set size. In *International Conference on Machine Learning*, pages 928–935, 2008.

S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1 regularized loss minimization. In *International Conference on Machine Learning*, pages 929–936, 2009.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *International Conference on Machine Learning*, pages 807–814, 2007.

S. Shalev-Shwartz, T. Zhang, and N. Srebro. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20:2807–2832, 2010.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

P. Tseng and S. Yun. A block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of Optimization Theory and Applications*, 140:513–535, 2009.

M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Olson, J. R. Marks, and J. R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences USA*, 98(20):11462–11467, 2001.

T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics*, 2(1):224–244, 2008.

L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

T. Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transaction on Information Theory*, 49:682–691, 2003.

T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, 2001.

Journal of Machine Learning Research 12 (2011) 1893-1921 Submitted 7/10; Revised 2/11; Published 6/11

Internal Regret with Partial Monitoring: Calibration-Based Optimal Algorithms

Vianney Perchet VIANNEY.PERCHET@NORMALESUP.ORG
Centre de Mathématiques et de Leurs Applications
École Normale Supérieure
61, avenue du président Wilson
94235 Cachan, France

Editor: Nicolo Cesa-Bianchi

Abstract

We provide consistent random algorithms for sequential decision under partial monitoring, when the decision maker does not observe the outcomes but receives instead random feedback signals. Those algorithms have no internal regret in the sense that, on the set of stages where the decision maker chose his action according to a given law, the average payoff could not have been improved in average by using any other fixed law.

They are based on a generalization of calibration, no longer defined in terms of a Voronoï diagram but instead of a Laguerre diagram (a more general concept). This allows us to bound, for the first time in this general framework, the expected average internal, as well as the usual external, regret at stage n by $O(n^{-1/3})$, which is known to be optimal.

Keywords: repeated games, on-line learning, regret, partial monitoring, calibration, Voronoï and Laguerre diagrams

1. Introduction

Hannan (1957) introduced the notion of regret in repeated games: a player (that will be referred as a decision maker or also a forecaster) has no external regret if, asymptotically, his average payoff could not have been greater if he had known, before the beginning of the game, the empirical distribution of moves of the other player. Blackwell (1956b) showed that the existence of such *externally consistent* strategies, first proved by Hannan (1957), is a consequence of his approachability theorem. A generalization of this result and a more precise notion of regret are due to Foster and Vohra (1997) and Fudenberg and Levine (1999): there exist internally consistent strategies, that is, such that for any of his action, the decision maker has no external regret on the set of stages where he actually chose this specific action. Hart and Mas-Colell (2000) also used Blackwell's approachability theorem to construct explicit algorithms that bound the internal (and therefore the external) regret at stage n by $O\left(n^{-1/2}\right)$.

Some of those results have been extended to the partial monitoring framework, that is, where the decision maker receives at each stage a random signal, whose law might depend on his unobserved payoff. Rustichini (1999) defined and proved the existence of externally consistent strategies, that is, such that the average payoff of the decision maker could not have been asymptotically greater if he had known, before the beginning of the game, the empirical distribution of signals. Actually, the relevant information is a vector of probability distributions, one for each action of the decision maker, that is called *a flag*.

Some algorithms bounding optimally the expected regret by $O\left(n^{-1/3}\right)$ have been exhibited under some strong assumptions on the signalling structure, see Cesa-Bianchi and Lugosi (2006), Theorem 6.7 for the optimality of this bound. For example, Jaksch et al. (2010) considered the Markov decision process framework, Cesa-Bianchi et al. (2005) assumed that payoffs can be deduced from flags and Lugosi et al. (2008) that feedbacks are deterministic (along with the fact that the worst compatible payoff is linear with respect to the flag). When no such assumption is made, Lugosi et al. (2008) provided an algorithm (based on the exponential weight algorithm) that bounds regret by $O\left(n^{-1/5}\right)$.

In this framework, internal regret was defined by Lehrer and Solan (2007); stages are no longer distinguished as a function of the action chosen by the decision maker (as in the full monitoring case) but as a function of its law. Indeed, the evaluation of the payoff (usually called *worst case*) is not linear with respect to the flag. So a best response (in a sense to be defined) to a given flag might consist only in a mixed action (i.e., a probability distribution over the set of actions). Lehrer and Solan (2007) also proved the existence and constructed internally consistent strategies, using the characterization of approachable convex sets due to Blackwell (1956a). Perchet (2009) provided an alternative algorithm, recalled in Section 3.1; this latter is based on calibration, a notion introduced by Dawid (1982). Roughly speaking, these algorithms ε-discretize arbitrarily the space of flags and each point of the discretization is called a possible prediction. Then, stage after stage, they predict what will be the next flag and output a *best response* to it. If the sequence of predictions is calibrated then the average flag, on the set of stages where a specific prediction is made, will be close to this prediction.

Thanks to the continuity of payoff and signaling functions, both algorithms bound the internal regret by $\varepsilon + O\left(n^{-1/2}\right)$. However the first drawback lies in their computational complexities: at each stage, the algorithm of Perchet (2009) solves a system of linear equations while the one Lehrer and Solan (2007), after a projection on a convex set, solves a linear program. In both case, the size of the linear system or program considered is polynomial in ε and exponential in the numbers of actions and signals. The second drawback is that the constants in the rate of convergence depend drastically on ε.

As a consequence, a classic *doubling trick* argument will generate an algorithm with a strongly sub-optimal rate of convergence, that might even depend on the size of the actions sets, and a complexity that increases with time.

Our main result is Theorem 24, stated in Section 3.2: it provides the first algorithm that bounds optimally both internal and external regret by $O\left(n^{-1/3}\right)$ in the general case. It is a modification of the algorithm of Perchet (2009) that does not use an arbitrary discretization but constructs carefully a specific one and then computes, stage by stage, the solution of a system of linear equations of constant size. In Section 4.1, an other algorithm, based on Blackwell's approachability as the one of Lehrer and Solan (2007), with optimal rate and smaller constants is exhibited; it requires however to solve, at each stage, a linear program of constant size.

Section 1 is devoted to the simpler framework of full monitoring. We recall definitions of calibration and regret and we provide a naïve algorithm to construct strategies with internal regret asymptotically smaller than ε. We show how to modify this algorithm, however in a not efficient way, in order to bound optimally the regret by $O\left(n^{-1/2}\right)$. This has to be seen only as a tool that can be easily adapted with partial monitoring in order to reach the optimal bound of $O\left(n^{-1/3}\right)$;

this is done in Section 2. Some extensions (the second algorithm, the so-called *compact case* and variants to strengthen the constants) are presented in Section 3. Some technical proofs can be found in Appendix.

2. Full Monitoring

Consider a two-person game Γ repeated in discrete time, where at stage $n \in \mathbb{N}$, a decision maker, or forecaster, (resp. the environment or Nature) chooses an action $i_n \in I$ (resp. $j_n \in \mathcal{J}$). This generates a payoff $\rho_n = \rho(i_n, j_n)$, where ρ is a mapping from $I \times \mathcal{J}$ to $\mathbb{R}$, and a regret $r_n \in \mathbb{R}^I$ defined by:

$$r_n = \Big[\rho(i, j_n) - \rho(i_n, j_n)\Big]_{i \in I} \in \mathbb{R}^I,$$

where I is the finite cardinality of I (and J the one of $\mathcal{J}$). This vector represents the differences between what the decision maker could have got and what he actually got.

The choices of i_n and j_n depend on the past observations (also called finite history) $h_{n-1} = (i_1, j_1, .., i_{n-1}, j_{n-1})$ and may be random. Explicitly, the set of finite histories is denoted by $H = \bigcup_{n\in\mathbb{N}} (I \times \mathcal{J})^n$, with $(I \times \mathcal{J})^0 = \emptyset$ and a strategy σ of the decision maker is a mapping from H to $\Delta(I)$, the set of probability distributions over I. Given the history $h_n \in (I \times \mathcal{J})^n$, $\sigma(h_n) \in \Delta(I)$ is the law of i_{n+1}. A strategy τ of Nature is defined similarly as a function from H to $\Delta(\mathcal{J})$. A pair of strategies (σ,τ) generates a probability, denoted by $\mathbb{P}_{\sigma,\tau}$, over $(\mathcal{H}, \mathcal{A})$ where $\mathcal{H} = (I \times \mathcal{J})^{\mathbb{N}}$ is the set of infinite histories embedded with the cylinder σ-field.

We extend the payoff mapping ρ to $\Delta(I) \times \Delta(\mathcal{J})$ by $\rho(x, y) = \mathbb{E}_{x,y}[\rho(i, j)]$ and for any sequence $a = (a_m)_{m\in\mathbb{N}}$ and any $n \in \mathbb{N}_*$, we denote by $\overline{a}_n = \frac{1}{n}\sum_{m=1}^{n} a_m$ the average of a up to stage n.

Definition 1 (Hannan, 1957) *A strategy σ of the forecaster is externally consistent if for every strategy τ of Nature:*

$$\limsup_{n\to\infty} \overline{r}^i_n \le 0, \quad \forall i \in I, \quad \mathbb{P}_{\sigma,\tau}\text{−as.}$$

In words, a strategy σ is externally consistent if the forecaster could not have had a greater payoff if he had known, before the beginning of the game, the empirical distribution of actions of Nature. Indeed, the external consistency of σ is equivalent to the fact that :

$$\limsup_{n\to\infty} \max_{x\in\Delta(I)} \rho(x, \overline{j}_n) - \overline{\rho}_n \le 0, \quad \mathbb{P}_{\sigma,\tau}\text{−as.} \tag{1}$$

Foster and Vohra (1997) (see also Fudenberg and Levine, 1999) defined a more precise notion of regret. The internal regret of the stage n, denoted by $R_n \in \mathbb{R}^{I\times I}$, is also generated by the choices of i_n and j_n and its (i,k)-th coordinate is defined by:

$$R^{ik}_n = \begin{cases} \rho(k, j_n) - \rho(i, j_n) & \text{if } i = i_n \\ 0 & \text{otherwise.} \end{cases}.$$

Stated differently, every row of the matrix R_n is null except the i_n-th which is r_n.

Definition 2 (Foster and Vohra, 1997) *A strategy σ of the forecaster is internally consistent if for every strategy τ of Nature:*

$$\limsup_{n\to\infty} \overline{R}^{ik}_n \le 0 \quad \forall i, k \in I, \quad \mathbb{P}_{\sigma,\tau}\text{−as.}$$

We introduce the following notations to define ε-internally consistency. Denote by $N_n(i)$ the set of stages before the n-th where the forecaster chose action i and $j_n(i) \in \Delta(\mathcal{J})$ the empirical distribution of Nature's actions on this set. Formally,

$$N_n(i) = \{m \in \{1,..,n\};\, i_m = i\} \quad \text{and} \quad j_n(i) = \frac{\sum_{m \in N_n(i)} j_m}{|N_n(i)|} \in \Delta(\mathcal{J}). \tag{2}$$

A strategy is ε-internally consistent if for every $i, k \in I$

$$\limsup_{n\to\infty} \frac{|N_n(i)|}{n}\left(\rho(k, j_n(i)) - \rho(i, j_n(i)) - \varepsilon\right) \le 0, \quad \mathbb{P}_{\sigma,\tau}\text{–as.}$$

If we define, for every $\varepsilon \ge 0$, the ε-best response correspondence by :

$$BR_\varepsilon(y) = \left\{x \in \Delta(I);\, \rho(x,y) \ge \max_{z\in\Delta(I)} \rho(z,y) - \varepsilon\right\},$$

then a strategy of the decision maker is ε-internally consistent if any action i is either an ε-best response to the empirical distribution of Nature's actions on $N_n(i)$ or the frequency of i is very small. We will simply denote BR_0 by BR and call it the best response correspondence.

From now on, given two sequences $\left\{l_m \in \mathcal{L}, a_m \in \mathbb{R}^d;\, m \in \mathbb{N}\right\}$ where $\mathcal{L}$ is a finite set, we will define the subset of integers $N_n(l)$ and the average $a_n(l)$ as in Equation (2), that is:

$$N_n(l) = \{m \in \{1,..,n\};\, l_m = l\} \quad \text{and} \quad a_n(l) = \frac{\sum_{m\in N_n(l)} a_m}{|N_n(l)|} \in \mathbb{R}^d.$$

Proposition 3 (Foster and Vohra, 1997) *For every $\varepsilon \ge 0$, there exist ε-internally consistent strategies.*

Although the notion of internal regret is a refinement of the notion of external regret (in the sense that any internally consistent strategy is also externally consistent), Blum and Mansour (2007) proved that any externally consistent algorithm can be efficiently transformed into an internally consistent one (actually they obtained an even stronger property called *swap consistency*).

Foster and Vohra (1997) and Hart and Mas-Colell (2000) proved directly the existence of 0-internally consistent strategies using different algorithms (with optimal rates and based respectively on the Expected Brier Score and Blackwell's approachability theorem). In some sense, we merge these two last proofs in order to provide a new one, given in the following section, that can be extended quite easily to the partial monitoring framework.

2.1 A Naïve Algorithm, Based on Calibration

The algorithm (a similar idea was used by Foster and Vohra, 1997) that constructs an ε-internally consistent strategy is based on this simple fact: if the forecaster can, stage by stage, foresee the law of Nature's next action, say $y \in \Delta(\mathcal{J})$, then he just has to choose any best response to y at the following stage. The continuity of ρ implies that the forecasts need not be extremely precise but only up to some $\delta > 0$.

Let $\{y(l);\, l \in \mathcal{L}\}$ be a δ-grid of $\Delta(\mathcal{J})$ (i.e., a finite set such that for every $y \in \Delta(\mathcal{J})$ there exists $l \in \mathcal{L}$ such that $\|y - y(l)\| \le \delta$) and $i(l)$ be a best response to $y(l)$, for every $l \in \mathcal{L}$. Then if δ is small enough:

$$\|y - y(l)\| \le 2\delta \Rightarrow i(l) \in BR_{2\varepsilon}(y).$$

It is possible to construct a *good sequence of forecasts* by computing a calibrated strategy (introduced by Dawid, 1982 and recalled in the following Subsection 2.1.1).

2.1.1 CALIBRATION

Consider a two-person repeated game Γ_c where, at stage n, Nature chooses the state of the world j_n in a finite set $\mathcal{J}$ and a decision maker (that will be referred in this setting as a predictor) predicts it by choosing $y(l_n)$ in $\mathcal{Y} = \{y(l);\ l \in \mathcal{L}\}$, a finite δ-grid of $\Delta(\mathcal{J})$ (its cardinality is denoted by L). As usual, a behavioral strategy σ of the predictor (resp. τ of Nature) is a mapping from the set of finite histories $H = \bigcup_{n\in\mathbb{N}} (\mathcal{L} \times \mathcal{J})^n$ to $\Delta(\mathcal{L})$ (resp. $\Delta(\mathcal{J})$). We also denote by $\mathbb{P}_{\sigma,\tau}$ the probability generated by the pair (σ,τ) over $(\mathcal{H},\mathcal{A})$ the set of infinite histories embedded with the cylinder topology.

Definition 4 (Dawid, 1982) *A strategy σ of the predictor is calibrated (with respect to $\mathcal{Y} = \{y(l);\ l \in \mathcal{L}\}$) if for every strategy τ of Nature, $\mathbb{P}_{\sigma,\tau}$-as:*

$$\limsup_{n\to\infty} \frac{|N_n(l)|}{n}\left(\|j_n(l) - y(l)\|^2 - \|j_n(l) - y(k)\|^2 \right) \le 0, \quad \forall k,l \in \mathcal{L},$$

where $\|\cdot\|$ is the Euclidian norm of $\mathbb{R}^J$.

In words, a strategy is calibrated if for every $l \in \mathcal{L}$, the empirical distribution of states, on the set of stages where $y(l)$ was predicted, is closer to $y(l)$ than to any other $y(k)$ (or the *frequency of l*, $|N_n(l)|/n$, is small).

Given a finite grid of $\Delta(\mathcal{J})$, the existence of calibrated strategies has been proved by Foster and Vohra (1998) using either the Expected Brier Score or a minmax theorem (actually this second argument is acknowledged to Hart). We give here a construction, related but simpler than the one of Foster and Vohra, due to Sorin (2008).

Proposition 5 (Foster and Vohra, 1998) *For any finite grid $\mathcal{Y}$ of $\Delta(\mathcal{J})$, there exist calibrated strategies with respect to $\mathcal{Y}$ such that for every strategy τ of Nature:*

$$\mathbb{E}_{\sigma,\tau}\left[\max_{l,k\in\mathcal{L}} \frac{|N_n(l)|}{n}\left(\|j_n(l) - y(l)\|^2 - \|j_n(l) - y(k)\|^2 \right)\right] \le O\left(\frac{1}{\sqrt{n}}\right).$$

Proof. Consider the auxiliary game where, at stage $n \in \mathbb{N}$, the predictor (resp. Nature) chooses $l_n \in \mathcal{L}$ (resp. $j_n \in \mathcal{J}$) and the vector payoff is the matrix $U_n \in \mathbb{R}^{L\times L}$ where

$$U_n^{lk} = \begin{cases} \|j_n - y(l)\|^2 - \|j_n - y(k)\|^2 & \text{if } l = l_n \\ 0 & \text{otherwise.} \end{cases}.$$

A strategy σ is calibrated with respect to $\mathcal{L}$ if $\overline{U}_n$ converges to the negative orthant. Indeed for every $l,k \in \mathcal{L}$, the (l,k)-th coordinate of $\overline{U}_n$ is

$$\begin{aligned} \overline{U}_n^{lk} &= \frac{|N_n(l)|}{n} \frac{\sum_{m\in N_n(l)} \|j_m - y(l)\|^2 - \|j_m - y(k)\|^2}{|N_n(l)|} \\ &= \frac{|N_n(l)|}{n}\left(\|j_n(l) - y(l)\|^2 - \|j_n(l) - y(k)\|^2 \right). \end{aligned}$$

Denote by $U_n^+ := \left\{\max\left(0, U_n^{lk}\right)\right\}_{l,k\in\mathcal{L}} =: U_n - U_n^-$ the positive part of U_n and by $\lambda_n \in \Delta(\mathcal{L})$ any invariant measure of U_n^+. We recall that λ is an invariant measure of a nonnegative matrix U if, for every $l \in \mathcal{L}$,

$$\sum_{k\in\mathcal{L}} \lambda(k) U^{kl} = \lambda(l) \sum_{k\in\mathcal{L}} U^{lk}.$$

Its existence is a consequence of Perron-Frobenius Theorem, see, for example, Seneta (1981).

Define the strategy σ of the predictor inductively as follows. Choose arbitrarily $\sigma(\emptyset)$, the law of the first action and at stage $n+1$, play accordingly to any invariant measure of U_n^+. We claim that this strategy is an approachability strategy of the negative orthant of $\mathbb{R}^{L\times L}$ because it satisfies Blackwell's (1956a) sufficient condition:

$$\forall n \in \mathbb{N}, \langle U_n - U_n^-, \mathbb{E}_{\lambda_n}[U_{n+1}|j_{n+1}] - U_n^- \rangle \le 0.$$

Indeed, for every possible $j_{n+1} \in \mathcal{J}$:

$$\langle U_n^+, \mathbb{E}_{\lambda_n}[U_{n+1}|j_{n+1}] \rangle = 0 = \langle U_n^+, U_n^- \rangle,$$

where the second equality follows from the definition of positive and negative parts.

Consider the first equality. The (l,k)-th coordinate of the matrix $\mathbb{E}_{\lambda_n}[U_{n+1}|j_{n+1}]$ is $\lambda_n(l)\left(\|j_{n+1} - y(l)\|^2 - \|j_{n+1} - y(k)\|^2\right)$, therefore the coefficient of $\|j_{n+1} - y(l)\|^2$ in the first term is $\lambda_n(l)\sum_{k\in\mathcal{L}}\left(U_n^+\right)^{lk} - \sum_{k\in\mathcal{L}}\lambda_n(k)\left(U_n^+\right)^{kl}$. This equals 0 since λ_n is an invariant measure of U_n^+.

Blackwell's (1956a) result also implies that $\mathbb{E}_{\sigma,\tau}[\|U_n^+\|] \le 2M_n n^{-1/2}$ for any strategy τ of Nature where $M_n^2 = \sup_{m\le n}\mathbb{E}_{\sigma,\tau}\left[\|U_m\|^2\right] = 4L$. □

Interestingly, the strategy σ we constructed in this proof is actually internally consistent in the game with action spaces $\mathcal{L}$ and $\mathcal{J}$ and payoffs defined by $\rho(l,j) = -\|j - y(l)\|^2$.

Corollary 6 *For any finite grid $\mathcal{Y}$ of $\Delta(\mathcal{J})$, there exists σ, a calibrated strategy with respect to $\mathcal{Y}$, such that for every strategy τ of Nature, with $\mathbb{P}_{\sigma,\tau}$ probability at least $1-\delta$:*

$$\max_{l,k\in\mathcal{L}} \frac{|N_n(l)|}{n}\left(\|j_n(l) - y(l)\|^2 - \|j_n(l) - y(k)\|^2\right) \le \frac{2M_n}{\sqrt{n}} + \Theta_n,$$

$$\begin{aligned}
\text{where}\quad \Theta_n &= \min\left\{\frac{v_n}{\sqrt{n}}\sqrt{2\ln\left(\frac{L^2}{\delta}\right)} + \frac{2}{3}\frac{K_n}{n}\ln\left(\frac{L^2}{\delta}\right), \frac{K_n}{\sqrt{n}}\sqrt{2\ln\left(\frac{L^2}{\delta}\right)}\right\};\\
M_n &= \sup_{m\le n}\sqrt{\mathbb{E}_{\sigma,\tau}\left[\|U_m\|^2\right]} \le 3\sqrt{L};\\
v_n^2 &= \sup_{m\le n}\sup_{l,k\in\mathcal{L}}\mathbb{E}_{\sigma,\tau}\left[\left|U_n^{lk} - \mathbb{E}_{\sigma,\tau}\left[U_n^{lk}\right]\right|^2\right] \le 3;\\
K_n &= \sup_{m\le n}\sup_{l,k\in\mathcal{L}}\left|U_n^{lk} - \mathbb{E}_{\sigma,\tau}\left[U_n^{lk}\right]\right| \le 3.
\end{aligned}$$

Proof. Proposition 5 implies that $\mathbb{E}_{\sigma,\tau}[U_n] \le 2M_n n^{-1/2}$. Hoeffding-Azuma's inequality (see Lemma 28 below in Section 4.3.1) implies that with probability at least $1-\delta$:

$$U_n^{lk} - \mathbb{E}_{\sigma,\tau}\left[U_n^{lk}\right] \le \frac{K_n}{\sqrt{n}}\sqrt{2\ln\left(\frac{1}{\delta}\right)}.$$

Freedman's inequality (an analogue of Bernstein's inequality for martingale), see Freedman (1975, Proposition 2.1) or Cesa-Bianchi and Lugosi (2006, Lemma A.8), implies that with probability at least $1-\delta$:

$$U_n^{lk} - \mathbb{E}_{\sigma,\tau}\left[U_n^{lk}\right] \leq \frac{v_n}{\sqrt{n}}\sqrt{2\ln\left(\frac{1}{\delta}\right)} + \frac{2}{3}\frac{K_n}{n}\ln\left(\frac{1}{\delta}\right).$$

The result is a consequence of these two inequalities and of Proposition 5. □

The definition of Θ_n as a minimum (and the use of Freedman's inequality) will be useful when we will refer to this corollary in the subsequent sections. Obviously, in the current framework, $\Theta_n \leq \frac{3}{\sqrt{n}}\sqrt{2\ln\left(\frac{L^2}{\delta}\right)}$.

2.1.2 BACK TO THE NAÏVE ALGORITHM

Let us now go back to the construction of ε-consistent strategies in Γ. Compute σ, a calibrated strategy with respect to a δ-grid $\mathcal{Y} = \{y(l);\ l \in \mathcal{L}\}$ of $\Delta(\mathcal{J})$ in an abstract calibration game Γ_c. Whenever the decision maker (seen as a predictor) should choose the action l in Γ_c, then he (seen as a forecaster) chooses $i(l) \in BR(y(l))$ in the original game Γ. We claim that this defines a strategy σ_ε which is 2ε-internally consistent.

Proposition 7 (Foster and Vohra, 1997) *For every $\varepsilon > 0$, the strategy σ_ε described above is 2ε-internally consistent.*

Proof. By definition of a calibrated strategy, for every $\eta > 0$, there exists with probability 1, an integer $N \in \mathbb{N}$ such that for every $l,k \in \mathcal{L}$ and for every $n \geq N$:

$$\frac{|N_n(l)|}{n}\left(\|j_n(l)-y(l)\|^2 - \|j_n(l)-y(k)\|^2\right) \leq \eta.$$

Since $\{y(k);\ k \in \mathcal{L}\}$ is a δ-grid of $\Delta(\mathcal{J})$, for every $l \in \mathcal{L}$ and every $n \in \mathbb{N}$, there exists $k \in \mathcal{L}$ such that $\|j_n(l)-y(k)\|^2 \leq \delta^2$, hence $\|j_n(l)-y(l)\|^2 \leq \delta^2 + \eta\frac{n}{|N_n(l)|}$. Therefore, since $i(l) \in BR(y(l))$:

$$\frac{|N_n(l)|}{n} \geq \frac{\eta}{\delta^2} \Rightarrow \|j_n(l)-y(l)\|^2 \leq 2\delta^2 \Rightarrow \rho(k,j_n(l)) - \rho(i(l),j_n(l)) \leq 2\varepsilon, \quad \forall k \in I,$$

for every $l \in \mathcal{L}$ and $n \geq N$. The (i,k)-th coordinate of R_n satisfies:

$$\begin{aligned}
\frac{|N_n(i)|}{n}\left(R_n^{ik} - 2\varepsilon\right) &\leq \frac{1}{n}\sum_{m\in N_n(i)}\left(\rho(k,j_m) - \rho(i,j_m) - 2\varepsilon\right)\\
&= \frac{1}{n}\sum_{l:i(l)=i}\sum_{m\in N_n(l)}\left(\rho(k,j_m) - \rho(i,j_m) - 2\varepsilon\right)\\
&= \sum_{l:i(l)=i}\frac{|N_n(l)|}{n}\left(\rho(k,j_n(l)) - \rho(i(l),j_n(l)) - 2\varepsilon\right).
\end{aligned}$$

Recall that either $\frac{|N_n(l)|}{n} \geq \frac{\eta}{\delta^2}$ and $\rho(k,j_n(i)) - \rho(i(l),j_n(l)) - 2\varepsilon \leq 0$, or $\frac{|N_n(l)|}{n} < \frac{\eta}{\delta^2}$. Since ρ is bounded (by $M_\rho > 0$), then :

$$\frac{|N_n(i)|}{n}\left(R_n^{ik} - 2\varepsilon\right) \leq \eta\frac{2M_\rho L}{\delta^2}, \quad \forall i \in I, \forall k \in I, \forall n \geq N,$$

which implies that σ is 2ε-internally consistent. □

Remark 8 *This naïve algorithm only achieves ε-consistency and Proposition 5 implies that*

$$\mathbb{E}_{\sigma,\tau}\left[\max_{i,k\in I}\left(R_n^{ik}-\varepsilon\right)\right]\leq O\left(\frac{1}{\sqrt{n}}\right).$$

The constants depend drastically on L, which is in the current framework in the order of ε^J, therefore it is not possible to obtain 0-internally consistency at the same rate with a classic doubling trick argument, that is, use a 2^{-k}-internally consistent strategy on N_k stages, then switch to a $2^{-(k+1)}$-internally consistent strategy, and so on (see Sorin, 1990, Proposition 3.2, page 56).

Moreover, since this algorithm is based on calibration, it computes at each stage an invariant measure of a non-negative matrix; this can be done, using Gaussian elimination, with $O\left(L^3\right)$ operations, thus this algorithm is far from being efficient (since its computational complexity is polynomial in ε and exponential in J). There exist 0*-internally consistent algorithms, see, for example, the reduction of Blum and Mansour (2007), that do not have this exponential dependency in the complexity or in the constants.*

On the bright side, this algorithm can be modified to obtain 0-consistency at optimal rate; obviously, it will still not be efficient with full monitoring (see Section 2.3). However, it has to be understood as a tool that can be easily adapted in order to exhibit, in the partial monitoring case, an optimal internal consistent algorithm (see Section 3.2). And in that last framework, it is not clear that we can remove the dependency on L (especially for the internal regret).

2.2 Calibration and Laguerre Diagram

Given a finite subset of Voronoï sites $\{z(l)\in\mathbb{R}^d;\ l\in\mathcal{L}\}$, the l-th Voronoï cell $V(l)$, or the cell associated to $z(l)$, is the set of points closer to $z(l)$ than to any other $z(k)$:

$$V(l)=\left\{Z\in\mathbb{R}^d;\ \|Z-z(l)\|^2\leq\|Z-z(k)\|^2,\quad\forall k\in\mathcal{L}\right\},$$

where $\|\cdot\|$ is the Euclidian norm of $\mathbb{R}^d$. Each $V(l)$ is a polyhedron (as the intersection of a finite number of half-spaces) and $\{V(l);\ l\in\mathcal{L}\}$ is a covering of $\mathbb{R}^d$. A calibrated strategy with respect to $\{z(l);\ l\in\mathcal{L}\}$ has the property that for every $l\in\mathcal{L}$, the frequency of l goes to zero, or the empirical distribution of states on $N_n(l)$, converges to $V(l)$.

The naïve algorithm uses the Voronoï diagram associated to an arbitrary grid of $\Delta(\mathcal{J})$ and assigns to every small cell an ε-best reply to every point of it; this is possible by continuity of ρ. A calibrated strategy ensures that $j_n(l)$ converges to $V(l)$ (or the frequency of l is small), thus choosing $i(l)$ on $N_n(l)$ was indeed a ε-best response to $j_n(l)$. With this approach, we cannot construct immediately 0-internally consistent strategy. Indeed, this would require that for every $l\in\mathcal{L}$ there exists a 0-best response $i(l)$ to every element y in $V(l)$. However, there is no reason for them to share a common best response because $\{z(l);\ l\in\mathcal{L}\}$ is chosen arbitrarily.

On the other hand, consider the simple game called *matching pennies*. Both players have two action Heads and Tails, so $\Delta(\mathcal{J})=\Delta(I)=[0,1]$, seen as the probability of choosing T. The payoff is 1 if both players choose the same action and -1 otherwise. Action H (resp. T) is a best response for Player 1 to any y in $[0,1/2]$ (resp. in $[1/2,1]$). These two segments are exactly the cells of the Voronoï diagram associated to $\{y(1)=1/4,y(2)=3/4\}$, therefore, performing a calibrated strategy with respect to $\{y(1), y(2)\}$ and playing H (resp. T) on the stages of type 1 (resp. 2) induces a 0-internally consistent strategy of Player 1.

This idea can be generalized to any game. Indeed, by Lemma 10 stated below, $\Delta(\mathcal{J})$ can be decomposed into polytopial best-response areas (a polytope is the convex hull of a finite number of points, its vertices). Given such a polytopial decomposition, one can find a finer Voronoï diagram (i.e., any best-response area is an union of Voronoï cells) and finally use a calibrated strategy to ensure convergence with respect to this diagram.

Although the construction of such a diagram is quite simple in $\mathbb{R}$, difficulties arise in higher dimension, even in $\mathbb{R}^2$. More importantly, the number of Voronoï sites can depend not only on the number of defining hyperplanes but also on the angles between them (thus being arbitrarily large even with a few hyperplanes). On the other hand, the description of a Laguerre diagram (this concept generalizes Voronoï diagrams) that refines a polytopial decomposition is quite simple and is described in Proposition 11 below. For this reason, we will consider from now on this kind of diagram (sometimes also called Power diagram) .

Given a subset of Laguerre sites $\{z(l) \in \mathbb{R}^d;\ l \in \mathcal{L}\}$ and weights $\{\omega(l) \in \mathbb{R};\ l \in \mathcal{L}\}$, the l-th Laguerre cell $P(l)$ is defined by:

$$P(l) = \left\{ Z \in \mathbb{R}^d;\ \|Z - z(l)\|^2 - \omega(l) \leq \|Z - z(k)\|^2 - \omega(k), \quad \forall k \in \mathcal{L} \right\},$$

where $\|\cdot\|$ is the Euclidian norm of $\mathbb{R}^d$. Each $P(l)$ is a polyhedron and $\mathcal{P} = \{P(l);\ l \in \mathcal{L}\}$ is a covering of $\mathbb{R}^d$.

Definition 9 *A covering $\mathcal{K} = \{K^i;\ i \in I\}$ of a polytope K with non-empty interior is a polytopial complex of K if for every i, j in the finite set I, K^i is a polytope with non-empty interior and the polytope $K^i \cap K^j$ has empty interior.*

This definition extends naturally to a polytope K with empty interior, if we consider the affine subspace generated by K.

Lemma 10 *There exists a subset $I' \subset I$ such that $\{B^i;\ i \in I'\}$ is a polytopial complex of $\Delta(\mathcal{J})$, where B^i is the i-th best response area defined by*

$$B^i = \{y \in \Delta(\mathcal{J});\ i \in BR(y)\} = BR^{-1}(i).$$

Proof. For any $y \in \Delta(\mathcal{J})$, $\rho(\cdot, y)$ is linear on $\Delta(I)$ thus it attains its maximum on I and $\bigcup_{i\in I} B^i = \Delta(\mathcal{J})$. Without loss of generality, we can assume that each B^i is non-empty, otherwise we drop the index i. For every $i, k \in I$, $\rho(i,\cdot) - \rho(k,\cdot)$ is linear on $\Delta(\mathcal{J})$ therefore B^i is a polytope; it is indeed defined by

$$\begin{aligned} B^i &= \{y \in \Delta(\mathcal{J});\ \rho(i,y) \geq \rho(k,y), \forall k \in I\} \\ &= \bigcap_{k \in I} \{y \in \mathbb{R}^J;\ \rho(i,y) - \rho(k,y) \geq 0\} \cap \Delta(\mathcal{J}), \end{aligned}$$

so it is the intersection of a finite number of half-spaces and the polytope $\Delta(\mathcal{J})$.

Moreover if B_0^{ik}, the interior of $B^i \cap B^k$, is non-empty then $\rho(i,\cdot)$ equals $\rho(k,\cdot)$ on the subspace generated by B_0^{ik} and therefore on $\Delta(\mathcal{J})$; consequently $B^i = B^k$. Denote by I' any subset of I such that for every $i \in I$, there exists exactly one $i' \in I'$ such that $B^i = B^{i'} \neq \emptyset$, then $\{B^i;\ i \in I'\}$ is a polytopial complex of $\Delta(\mathcal{J})$. □

Proposition 11 *Let $\mathcal{K} = \{K^i;\ i \in I\}$ be a polytopial complex of a polytope $K \subset \mathbb{R}^d$. Then there exists $\{z(l) \in \mathbb{R}^d, \omega(l) \in \mathbb{R};\ l \in \mathcal{L}\}$, a finite set of Laguerre sites and weights, such that the Laguerre diagram $\mathcal{P} = \{P(l);\ l \in \mathcal{L}\}$ refines $\mathcal{K}$, that is, every K^i is a finite union of cells.*

Proof. Let $\mathcal{K} = \{K^i;\ i \in I\}$ be a polytopial complex of $K \subset \mathbb{R}^d$. Each K^i is a polytope, thus defined by a finite number of hyperplanes. Denote by $\mathcal{H} = \{H_t;\ t \in \mathcal{T}\}$ the set of all defining hyperplanes (the finite cardinality of $\mathcal{T}$ is denoted by T) and $\widehat{\mathcal{K}} = \{\widehat{K}^l;\ l \in \mathcal{L}\}$ the finest decomposition of $\mathbb{R}^d$ induced by $\mathcal{H}$ (usually called arrangement of hyperplanes) which by definition refines $\mathcal{K}$. Theorem 3 and Corollary 1 of Aurenhammer (1987) imply that $\widehat{\mathcal{K}}$ is the Laguerre diagram associated to some $\{z(l), \omega(l);\ l \in \mathcal{L}\}$ whose exact computation requires the following notation:

i) for every $t \in \mathcal{T}$, let $c_t \in \mathbb{R}^d$ and $b_t \in \mathbb{R}$ (which can, without loss of generality, be assumed to be non zero) such that
$$H_t = \left\{X \in \mathbb{R}^d;\ \langle X, c_t \rangle = b_t\right\}.$$

ii) For every $l \in \mathcal{L}$ and $t \in \mathcal{T}$, $\sigma_t(l) = 1$ if the origin of $\mathbb{R}^d$ and $\widehat{K}^l$ are in the same halfspace defined by H_t and $\sigma_t(l) = -1$ otherwise.

iii) For every $l \in \mathcal{L}$, we define:
$$z(l) = \frac{\sum_{t\in\mathcal{T}} \sigma_t(l) c_t}{T} \quad \text{and} \quad \omega(l) = \|z(l)\|^2 + 2\frac{\sum_{t\in\mathcal{T}} \sigma_t(l) b_t}{T}.$$

Note that one can add the same constant to every weight $\omega(l)$. □

Buck (1943) proved that the number of cells defined by T hyperplanes in $\mathbb{R}^d$ is bounded by $\sum_{k=0}^{d} \binom{T}{k} =: \phi(T,d)$, where $\binom{T}{k}$ is the binomial coefficient, T *choose* k. Moreover, T is smaller than $I(I-1)/2$ (in the case where each K^i has a non-empty intersection with every other polytope), so $L \leq \phi\left(\frac{I^2}{2}, d\right)$.

If $d \geq n$, then $\phi(n,d) = 2^n$. Pascal's rule and a simple induction imply that, for every $n, d \in \mathbb{N}$, $\phi(n,d) \leq (n+1)^d$. Finally, for any $n \geq 2d$, by noticing that
$$\frac{\binom{n}{d} + \binom{n}{d-1} + .. + \binom{n}{0}}{\binom{n}{d}} \leq \sum_{m=0}^{d} \left(\frac{d}{n-d+1}\right)^m \leq \sum_{m=0}^{\infty} \left(\frac{d}{n-d+1}\right)^m = \frac{n-d+1}{n-2d+1} \leq 1+d,$$
we can conclude that $\phi(n,d) \leq (1+d)\binom{n}{d} \leq (1+d)\frac{n^d}{d!}$.

Lemma 12 *Let $\mathcal{P} = \{P(l);\ l \in \mathcal{L}\}$ be a Laguerre diagram associated to the set of sites and weights $\{z(l) \in \mathbb{R}^d, \omega(l) \in \mathbb{R};\ l \in \mathcal{L}\}$. Then, there exists a positive constant $M_P > 0$ such that for every $Z \in \mathbb{R}^d$ if*
$$\|Z - z(l)\|^2 - \omega(l) \leq \|Z - z(k)\|^2 - \omega(k) + \varepsilon, \quad \forall l, k \in \mathcal{L} \tag{3}$$
then $d(Z, P(l))$ is smaller than $M_P\varepsilon$.

The proof can be found in Appendix A.1; the constant M_P depends on the Laguerre diagram, and more precisely on the inner products $\langle c_t, c_{t'} \rangle$, for every $t, t' \in \mathcal{T}$.

2.3 Optimal Algorithm with Full Monitoring

We reformulate Proposition 5 and Corollary 6 in terms of Laguerre diagram.

Theorem 13 *For any set of sites and weights $\{y(l) \in \mathbb{R}^J, \omega(l) \in \mathbb{R};\ l \in \mathcal{L}\}$ there exists a strategy σ of the predictor such that for every strategy τ of Nature:*

$$\mathbb{E}_{\sigma,\tau}\left[\left\|\left(U_{\omega,n}\right)^{+}\right\|\right] \leq O\left(\frac{1}{\sqrt{n}}\right) \text{ where } U_{\omega,n} \text{ is defined by :}$$

$$U^{lk}_{\omega,n} = \begin{cases} [\|j_n - y(l)\|^2 - \omega(l)] - [\|j_n - y(k)\|^2 - \omega(k)] & \text{if } l = l_n \\ 0 & \text{otherwise} \end{cases} .$$

Corollary 14 *For any set of sites and weights $\{y(l) \in \mathbb{R}^J, \omega(l) \in \mathbb{R};\ l \in \mathcal{L}\}$, there exists a strategy σ of the predictor such that, for every strategy τ of Nature, with $\mathbb{P}_{\sigma,\tau}$ probability at least $1-\delta$:*

$$\max_{l,k\in\mathcal{L}} \frac{|N_n(l)|}{n}\left(\left[\|j_n(l) - y(l)\|^2 - \omega(l)\right] - \left[\|j_n(l) - y(k)\|^2 - \omega(k)\right]\right) \leq \frac{2M_n}{\sqrt{n}} + \Theta_n$$

$$\begin{aligned}
\text{where } M_n &= \sup_{m\leq n}\sqrt{\mathbb{E}_{\sigma,\tau}\left[\|U_{\omega,m}\|^2\right]} \leq 4\sqrt{L}\|(b,c)\|_\infty ; \\
\Theta_n &= \min\left\{\frac{v_n}{\sqrt{n}}\sqrt{2\ln\left(\frac{L^2}{\delta}\right)} + \frac{2}{3}\frac{K_n}{n}\ln\left(\frac{L^2}{\delta}\right), \frac{K_n}{\sqrt{n}}\sqrt{2\ln\left(\frac{L^2}{\delta}\right)}\right\}; \\
v_n^2 &= \sup_{m\leq n}\sup_{l,k\in\mathcal{L}} \mathbb{E}_{\sigma,\tau}\left[\left|U^{lk}_{\omega,m} - \mathbb{E}_{\sigma,\tau}\left[U^{lk}_{\omega,m}\right]\right|^2\right] \leq 4\|(b,c)\|^2_\infty ; \\
K_n &= \sup_{m\leq n}\sup_{l,k\in\mathcal{L}} \left|U^{lk}_{\omega,m} - \mathbb{E}_{\sigma,\tau}\left[U^{lk}_{\omega,m}\right]\right| \leq 4\|(b,c)\|_\infty , \\
\|(b,c)\|_\infty &= \sup_{t\in\mathcal{T}}\|c_t\| + \sup_{t\in\mathcal{T}}|b_t| .
\end{aligned}$$

Such a strategy is said to be calibrated with respect to $\{y(l), \omega(l);\ l \in \mathcal{L}\}$.

The proof is identical to the one of Proposition 5 and Corollary 6. We have now the material to construct our new *tool algorithm*:

Theorem 15 *There exists an internally consistent strategy σ of the forecaster such that for every strategy τ of Nature and every $n \in \mathbb{N}$, with $\mathbb{P}_{\sigma,\tau}$ probability greater than $1-\delta$:*

$$\max_{i,k\in I} R^{ik}_n \leq O\left(\sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{n}}\right).$$

Proof. The existence of a Laguerre Diagram $\{Y(l);\ l \in \mathcal{L}\}$ associated to a finite set $\{y(l) \in \mathbb{R}^J, \omega(l) \in \mathbb{R};\ l \in \mathcal{L}\}$ that refines $\{B^i;\ i \in I\}$ is implied by Lemma 10 and Proposition 11. So, for every $l \in \mathcal{L}$, there exists $i(l)$ such that $Y(l) \subset B^{i(l)}$. As in the naïve algorithm, the strategy σ of the decision maker is constructed through a strategy $\widehat{\sigma}$ calibrated with respect to $\{y(l), \omega(l);\ l \in \mathcal{L}\}$.

Whenever, accordingly to $\widehat{\sigma}$, the decision maker (seen as a predictor) should play l in Γ_c, then he (seen as a forecaster) plays $i(l)$ in Γ.

If we denote by $\widetilde{j_n}(l)$ the projection of $j_n(l)$ onto $Y(l)$ then:

$$\begin{aligned}
R_n^{ik} &= \sum_{l:i(l)=i} \frac{|N_n(l)|}{n}\Big(\rho(k, j_n(l)) - \rho(i(l), j_n(l))\Big)\\
&\leq \sum_{l:i(l)=i} \frac{|N_n(l)|}{n}\left(\Big[\rho(k, j_n(l)) - \rho(k, \widetilde{j_n}(l))\Big] + \Big[\rho(i(l), \widetilde{j_n}(l)) - \rho(i(l), j_n(l))\Big]\right)\\
&\leq \sum_{l:i(l)=i} \frac{|N_n(l)|}{n}\left(2M_\rho \left\|\widetilde{j_n}(l) - j_n(l)\right\|\right)\\
&\leq (2M_\rho M_P L)\max_{l,k\in\mathcal{L}} \frac{|N_n(l)|}{n}\left(\Big[\|j_n(l)-y(l)\|^2 - \omega(l)\Big] - \Big[\|j_n(l)-y(k)\|^2-\omega(k)\Big]\right)
\end{aligned}$$

where the second inequality is due to the fact that $i(l) \in BR(\widetilde{j_n}(l))$ and the third to the fact that ρ is M_ρ-Lipschitz. The fourth inequality is a consequence of Lemma 12.

Corollary 14 yields that for every strategy τ of Nature, with $\mathbb{P}_{\sigma,\tau}$ probability at least $1-\delta$:

$$\max_{l,k} \frac{N_n(l)}{n}\left(\Big[\|j_n(l)-y(l)\|^2 - \omega(l)\Big] - \Big[\|j_n(l)-y(k)\|^2-\omega(k)\Big]\right) \leq \frac{8\sqrt{L}\|(b,c)\|_\infty}{\sqrt{n}} + \frac{4\|(b,c)\|_\infty}{\sqrt{n}}\sqrt{2\ln\left(\frac{L^2}{\delta}\right)},$$

therefore with $\Omega_0 = 16M_\rho M_P L^{3/2}\|(b,c)\|_\infty$ and $\Omega_1 = 8M_\rho M_P L^{1/2}\|(b,c)\|_\infty$ one has that for every strategy of Nature and with probability at least $1-\delta$:

$$\max_{i,k\in I} R_n^{ik} = \max_{i,k\in I}\frac{|N_n(i)|}{n}\Big(\rho(k,j_n(i)) - \rho(i,j_n(i))\Big) \leq \frac{\Omega_0}{\sqrt{n}} + \frac{\Omega_1}{\sqrt{n}}\sqrt{2\ln\left(\frac{L^2}{\delta}\right)}.$$

□

Remark 16 *Theorem 15 is already well-known. The construction of this internally consistent strategy relies on Theorem 13, which is implied by the existence of internally consistent strategies... Moreover, as mentioned before, it is far from being efficient since L, that enters both in the computational complexity and in the constant, is polynomial in I^J. There exist efficient algorithms, see, for example, Foster and Vohra (1997) or Blum and Mansour (2007).*

However, the calibration is defined in the space of Nature's action, where real payoffs are irrelevant; they are only used to decide which action is associated to each prediction. Therefore the algorithm does not require that the forecaster observes his real payoffs, as long as he knows what is the best response to his information (Nature's action in this case). This is precisely why our algorithm can be generalized to the partial monitoring framework.

The polytopial decomposition of $\Delta(\mathcal{J})$ induced by $\{b_t, c_t;\ t\in\mathcal{T}\}$ is exactly the same as the one induced by $\{\gamma b(t), \gamma c(t);\ t\in\mathcal{T}\}$ for any $\gamma > 0$. Thus, by choosing γ small enough, $\|(b,c)\|_\infty$ and therefore the constants in Corollary 14 can be arbitrarily small (i.e., multiplied by any $\gamma > 0$).

However, these two Laguerre diagrams are associated to the sets of sites and weights $\mathcal{L}(1)$ and $\mathcal{L}(\gamma)$, where $\mathcal{L}(\gamma) = \{\gamma z(l), \gamma\omega(l) + \gamma^2\|z(l)\|^2 - \gamma\|z(l)\|;\ l \in \mathcal{L}\}$. If $\mathcal{L}(\gamma)$ is used instead of $\mathcal{L}(1)$, then the constant M_P defined in Lemma 12 should be divided by γ. So, as expected, the constants in the proof of Theorem 15 do not depend on γ. From now on, we will assume that $\|(b,c)\|_\infty$ is smaller than 1.

3. Partial Monitoring

In the partial monitoring framework, the decision maker does not observe Nature's actions. There is a finite set of signals $\mathcal{S}$ (of cardinality S) such that, at stage n the forecaster receives only a random signal $s_n \in \mathcal{S}$. Its law is $s(i_n, j_n)$ where s is a mapping from $I \times \mathcal{J}$ to $\Delta(\mathcal{S})$, known by the decision maker.

We define the mapping $\mathbf{s}$ from $\Delta(\mathcal{J})$ to $\Delta(\mathcal{S})^I$ by $\mathbf{s}(y) = \Big(\mathbb{E}_y\left[s(i,j)\right]\Big)_{i\in I} \in \Delta(\mathcal{S})^I$. Any element of $\Delta(\mathcal{S})^I$ is called a flag (it is a vector of probability distributions over $\mathcal{S}$) and we will denote by $\mathcal{F}$ the range of $\mathbf{s}$. Given a flag f in $\mathcal{F}$, the decision maker cannot distinguish between any different mixed actions y and y' in $\Delta(\mathcal{J})$ that generate f, that is, such that $\mathbf{s}(y) = \mathbf{s}(y') = f$. Thus $\mathbf{s}$ is the maximal informative mapping about Nature's action. We denote by $f_n = \mathbf{s}(j_n)$ the (unobserved) flag of stage $n \in \mathbb{N}$.

Example 1 *Label efficient prediction (Cesa-Bianchi and Lugosi, 2006, Example 6.8):*

Consider the following game. Nature chooses an outcome G or B and the forecaster can either observe the actual outcome (action o) or choose to not observe it and pick a label g or b. His payoff is equal to 1 if he chooses the right label and otherwise is equal to 0. Payoffs and laws of signals are defined by the following matrices (where a, b and c are three different probabilities over a finite given set S).

$$\text{Payoffs:}\quad \begin{array}{c|c|c|} & G & B \\ \hline o & 0 & 0 \\ \hline g & 0 & 1 \\ \hline b & 1 & 0 \\ \hline \end{array} \qquad \text{and signals:}\quad \begin{array}{c|c|c|} & G & B \\ \hline o & a & b \\ \hline g & c & c \\ \hline b & c & c \\ \hline \end{array}$$

Action G, whose best response is g, generates the flag (a,c,c) and action B, whose best response is b, generates the flag (b,c,c). In order to distinguish between those two actions, the forecaster needs to know $s(o,y)$ although action o is never a best response (but is purely informative).

The worst payoff compatible with x and $f \in \mathcal{F}$ is defined by:

$$W(x,f) = \inf_{y\in\mathbf{s}^{-1}(f)} \rho(x,y),$$

and W is extended to $\Delta(\mathcal{S})^I$ by $W(x,f) = W\left(x, \Pi_{\mathcal{F}}(f)\right)$.

As in the full monitoring case, we define, for every $\varepsilon \geq 0$, the ε-best response multivalued mapping $BR_\varepsilon : \Delta(\mathcal{S})^I \rightrightarrows \Delta(I)$ by :

$$BR_\varepsilon(f) = \left\{ x \in \Delta(I);\ W(x,f) \geq \sup_{z\in\Delta(I)} W(z,f) - \varepsilon \right\}.$$

Given a flag $f \in \Delta(\mathcal{S})^I$, the function $W(\cdot, f)$ may not be linear so the best response of the forecaster might not contain any element of I.

Example 2 *Matching pennies in the dark:*

Consider the matching pennies game where the forecaster does not observe the coin but always receives the same signal c: every choice of Nature generates the same flag (c,c). For every $x \in [0,1] = \Delta(\{H,T\})$ (the probability of playing T), the worst compatible payoff $W(x,(c,c)) = \min_{y\in\Delta(J)} \rho(x,y)$ is equal to $-|1-2x|$ thus is non-negative only for $x = 1/2$. Therefore the only best response of the forecaster is to play $\frac{1}{2}H + \frac{1}{2}T$, while actions H and T give the worst payoff of -1.

The definition of external consistency and especially Equation (1) extend naturally to this framework: a strategy of the decision maker is externally consistent if he could not have improved his payoff by knowing, before the beginning of the game, the average flag:

Definition 17 (Rustichini, 1999) *A strategy σ of the forecaster is externally consistent if for every strategy τ of Nature:*

$$\limsup_{n\to+\infty} \max_{z\in\Delta(I)} W(z,f_n) - \rho_n \leq 0, \quad \mathbb{P}_{\sigma,\tau}\text{-}as.$$

The main issue is the definition of internally consistency. In the full monitoring case, the forecaster has no internal regret if, for every $i \in I$, the action i is a best-response to the empirical distribution of Nature's actions, on the set of stages where i was actually chosen. In the partial monitoring framework, the decision maker's action should be a best response to the average flag. Since it might not belong to I but rather to $\Delta(I)$, we will (following Lehrer and Solan, 2007) distinguish the stages not as a function of the action actually chosen, but as a function of its law.

We make an extra assumption on the characterization of the forecaster's strategy: it can be generated by a finite family of mixed actions $\{x(l) \in \Delta(I);\ l \in \mathcal{L}\}$ such that, at stage $n \in \mathbb{N}$, the forecaster chooses a type l_n and, given that type, the law of his action i_n is $x(l_n) \in \Delta(I)$.

Denote by $N_n(l) = \{m \in \{1,..,n\};\ l_m = l\}$ the set of stages before the n-th whose type is l. Roughly speaking, a strategy will be ε-internally consistent (with respect to the set $\mathcal{L}$) if, for every $l \in \mathcal{L}$, $x(l)$ is an ε-best response to $f_n(l)$, the average flag on $N_n(l)$ (or the frequency of the type l, $|N_n(l)|/n$, converges to zero).

The finiteness of $\mathcal{L}$ is required to get rid of strategies that trivially insure that every frequency converges to zero (for instance by choosing only once every mixed action). The choice of $\{x(l);\ l \in \mathcal{L}\}$ and the description of the strategies are justified more precisely below by Remark 21 in Section 3.2.

Definition 18 (Lehrer and Solan, 2007) *For every $n \in \mathbb{N}$ and every $l \in \mathcal{L}$, the average internal regret of type l at stage n is*

$$\mathcal{R}_n(l) = \sup_{x\in\Delta(I)} \left[W(x,f_n(l)) - \rho_n(l)\right].$$

A strategy σ of the forecaster is $(\mathcal{L},\varepsilon)$-internally consistent if for every strategy τ of Nature:

$$\limsup_{n\to+\infty} \frac{|N_n(l)|}{n}\left(\mathcal{R}_n(l) - \varepsilon\right) \leq 0, \quad \forall l \in \mathcal{L}, \quad \mathbb{P}_{\sigma,\tau}\text{-}as.$$

In words, a strategy is $(\mathcal{L},\varepsilon)$-internally consistent if, for every $l \in \mathcal{L}$, the forecaster could not have had, for sure, a better payoff (of at least ε) if he had known, before the beginning of the game, the average flag on $N_n(l)$ (or the frequency of l is small).

3.1 A Naïve Algorithm

Theorem 19 (Lehrer and Solan, 2007) *For every $\varepsilon > 0$, there exist $(\mathcal{L}, \varepsilon)$-internally consistent strategies.*

Lehrer and Solan (2007) proved the existence and constructed such strategies and an alternative, yet close, algorithm has been provided by Perchet (2009). The main ideas behind them are similar to the full monitoring case so we will quickly describe them. For simplicity, we assume in the following sketch of the proof, that the decision maker fully observes the sequence of flags $f_n = \mathbf{s}(j_n) \in \Delta(\mathcal{S})^I$.

Recall that W is continuous (Lugosi et al., 2008, Proposition A.1), so for every $\varepsilon > 0$ there exist two finite families $\mathcal{G} = \{f(l) \in \Delta(\mathcal{S})^I;\ l \in \mathcal{L}\}$, a δ-grid of $\Delta(\mathcal{S})^I$, and $X = \{x(l) \in \Delta(I);\ l \in \mathcal{L}\}$ such that if f is δ-close to $f(l)$ and x is δ-close to $x(l)$ then x belongs to $BR_\varepsilon(f)$. A calibrated algorithm ensures that:

i) $f_n(l)$ is asymptotically δ-close to $f(l)$, because it is closer to $f(l)$ than to every other $f(k)$;

ii) $i_n(l)$ converges to $x(l)$ as soon as $|N_n(l)|$ is big enough, because on $N_n(l)$ the choices of action of the decision maker are independent and identically distributed accordingly to $x(l)$;

iii) $\rho_n(l)$ converges to $\rho(x(l), j_n(l))$ which is greater than $W\Big(x(l), f_n(l)\Big)$ because $j_n(l)$ generates the flag $f_n(l)$.

Therefore, $W\Big(x(l), f_n(l)\Big)$ is close to $W\Big(x(l), f(l)\Big)$ which is greater than $W\Big(z, f(l)\Big)$ for any $z \in \Delta(I)$. As a consequence $\rho_n(l)$ is asymptotically greater (up to some $\varepsilon > 0$) than $\sup_z W\Big(z, f_n(l)\Big)$, as long as $|N_n(l)|$ is big enough.

The difference between the two algorithm lies in the construction of a calibrated strategy. On one hand, the algorithm of Lehrer and Solan (2007) reduces to Blackwell's approachability of some convex set $\mathcal{C} \subset \mathbb{R}^{LSI}$; it therefore requires to solve at each stage a linear program of size polynomial in ε^{SI}, after a projection on $\mathcal{C}$. On the other hand, the algorithm of Perchet (2009) is based on the construction given in Section 2.1.1; it solves at each stage a system of linear equation of size also polynomial in ε^{SI}.

The conclusions of the full monitoring case also apply here: these highly non-efficient algorithms cannot be used directly to construct $(\mathcal{L}, 0)$-internally consistent strategy with optimal rates since the constants depend drastically on ε . We will rather prove that one can define wisely once for all $\{f(l), \omega(l);\ l \in \mathcal{L}\}$ and $\{x(l);\ l \in \mathcal{L}\}$ (see Proposition 20 and Proposition 11) so that $x(l) \in \Delta(I)$ is a 0-best response to any flag f in $P(l)$, the Laguerre cell associated to $f(l)$ and $\omega(l)$.

The strategy associated with these choices will be $(\mathcal{L}, 0)$-internally consistent, with an optimal rate of convergence and a computational complexity polynomial in L.

3.2 Optimal Algorithms

As in the full monitoring framework (cf Lemma 10), we define for every $x \in \Delta(I)$ the x-best response area B^x as the set of flags to which x is a best response :

$$B^x = \left\{f \in \Delta(\mathcal{S})^I;\ x \in BR(f)\right\} = BR^{-1}(x).$$

Since W is continuous, the family $\{B^x;\ x \in \Delta(I)\}$ is a covering of $\Delta(\mathcal{S})^I$. However, one of its finite subsets can be decomposed into a finite polytopial complex:

Proposition 20 *There exists a finite family $X = \{x(l) \in \Delta(I);\ l \in \mathcal{L}\}$ such that the family $\left\{B^{x(l)};\ l \in \mathcal{L}\right\}$ of associated best response area can be further subdivided into a polytopial complex of $\Delta(\mathcal{S})^I$.*

The rather technical proof can be found in Appendix A.2. In this framework and because of the lack of linearity of W, any $B^{x(l)}$ might not be convex nor connected. However, each one of them is a finite union of polytopes and the family of all those polytopes is a complex of $\Delta(\mathcal{S})^I$.

Remark 21 *As a consequence of Proposition 20, there exists a finite set $X \subset \Delta(I)$ that contains a best response to any flag f. In particular, if the decision maker could observe the flag f_n before choosing his action x_n then, at every stage, x_n would be in X. So in the description of the strategies of the forecaster, the finite set $\{x(l);\ l \in \mathcal{L}\} = X$ is in fact intrinsic that is, determined by the description of the payoff and signal functions.*

As a consequence of this remark, mentioning $\mathcal{L}$ is irrelevant; so we will, from now on, simply speak of *internally consistent strategies*.

3.2.1 OUTCOME DEPENDENT SIGNALS

In this section, we assume that the laws of the signal received by the decision maker are independent of his action. Formally, for every $i, i' \in I$, the two mappings $s(i,\cdot)$ and $s(i',\cdot)$ are equal. Therefore, $\mathcal{F}$ (the set of realizable flags) can be seen as a polytopial subset of $\Delta(\mathcal{S})$. Proposition 20 holds in this framework, hence there exists a finite family $\{x(l);\ l \in \mathcal{L}\}$ such that for any flag $f \in \mathcal{F}$, there is some $l \in \mathcal{L}$ such that $x(l)$ is a best-reply to f. Moreover, for a fixed $l \in \mathcal{L}$, the set of such flags is a polytope.

Theorem 22 *There exists an internally consistent strategy σ such that for every strategy τ of Nature, with $\mathbb{P}_{\sigma,\tau}$-probability at least $1-\delta$:*

$$\sup_{l \in \mathcal{L}} \frac{|N_n(l)|}{n} \mathcal{R}_n(l) \leq O\left(\sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{n}}\right).$$

Proof. Propositions 11 and 20 imply the existence of two finite families $\{x(l);\ l \in \mathcal{L}\}$ and $\{f(l), \omega(l);\ l \in \mathcal{L}\}$ such that $x(l)$ is a best response to any f in $P(l)$, the Laguerre cell associated to $f(l)$ and $\omega(l)$. Assume, for the moment, that for any two different l and k in $\mathcal{L}$, the probability measures $x(l)$ and $x(k)$ are different.

The strategy σ is defined as follows. Compute a strategy $\widehat{\sigma}$ calibrated with respect to $\{f(l), \omega(l);\ l \in \mathcal{L}\}$. When the decision maker (seen as a predictor) should choose $l \in \mathcal{L}$ accordingly to $\widehat{\sigma}$, then he (seen as a forecaster) plays accordingly to $x(l)$ in the original game. Corollary 14 (with the assumption that $\|(b,c)\|_\infty$ is smaller than 1) implies that with $\mathbb{P}_{\sigma,\tau}$ probability at least $1-\delta_1$:

$$\max_{l \in \mathcal{L}} \frac{|N_n(l)|}{n} \left(\left[\|s_n(l) - f(l)\|^2 - \omega(l) \right] - \left[\|s_n(l) - f(k)\|^2 - \omega(k) \right] \right) \leq$$
$$\frac{8\sqrt{L}}{\sqrt{n}} + \frac{4}{\sqrt{n}} \sqrt{2 \ln\left(\frac{L^2}{\delta_1}\right)},$$

therefore combined with Lemma 12, this yields that :

$$\max_{l\in\mathcal{L}} \frac{|N_n(l)|}{n}\left\|s_n(l)-\widetilde{f}_n(l)\right\| \leq \frac{8M_P\sqrt{L}}{\sqrt{n}}+\frac{4M_P}{\sqrt{n}}\sqrt{2\ln\left(\frac{L^2}{\delta_1}\right)}, \tag{4}$$

where $\widetilde{f}_n(l)$ is the projection of $s_n(l)$ onto $P(l)$.

Hoeffding-Azuma's inequality implies that with $\mathbb{P}_{\sigma,\tau}$ probability at least $1-\delta_2$:

$$\max_{l\in\mathcal{L}} \frac{|N_n(l)|}{n}\left\|s_n(l)-f_n(l)\right\| \leq \sqrt{\frac{2\ln\left(\frac{2SL}{\delta_2}\right)}{n}} \tag{5}$$

and with probability at least $1-\delta_3$:

$$\max_{l\in\mathcal{L}} \frac{|N_n(l)|}{n}\left|\rho_n(l)-\rho(x(l),j_n(l))\right| \leq M_\rho\sqrt{\frac{2\ln\left(\frac{2L}{\delta_3}\right)}{n}}. \tag{6}$$

W is M_W-Lipschitz in f (see Lugosi et al., 2008) and $\mathbf{s}(j_n(l))=f_n(l)$ therefore:

$$\rho_n(l)\geq W\left(x(l),\widetilde{f}_n(l)\right)-\left|\rho_n(l)-\rho(x(l),j_n(l))\right|-M_W\left\|f_n(l)-\widetilde{f}_n(l)\right\| \tag{7}$$

and

$$\begin{aligned}\max_{x\in\Delta(I)} W\left(x,f_n(l)\right) &\leq \max_{x\in\Delta(I)} W\left(x,\widetilde{f}_n(l)\right)+M_W\left(\left\|s_n(l)-f_n(l)\right\|+\left\|s_n(l)-\widetilde{f}_n(l)\right\|\right)\\ &\leq W\left(x(l),\widetilde{f}_n(l)\right)+M_W\left(\left\|s_n(l)-f_n(l)\right\|+\left\|s_n(l)-\widetilde{f}_n(l)\right\|\right) \end{aligned} \tag{8}$$

since $x(l)$ is a best response to $\widetilde{f}_n(l)$. Equations (7) and (8) yield

$$\mathcal{R}_n(l)\leq 2M_W\left\|s_n(l)-f_n(l)\right\|+2M_W\left\|s_n(l)-\widetilde{f}_n(l)\right\|+\left|\rho_n(l)-\rho(x(l),j_n(l))\right|. \tag{9}$$

Combining Equations (4), (5), (6) and (9) gives that with probability at least $1-\delta$, if we define $\Omega_0=16M_PM_W\sqrt{L}$, $\Omega_1=\left(2M_W+8M_WM_P+M_\rho\right)$ and $\Omega_2=L(L+2S+2)$:

$$\sup_{l\in\mathcal{L}}\frac{|N_n(l)|}{n}\mathcal{R}_n(l)\leq\frac{\Omega_0}{\sqrt{n}}+\frac{\Omega_1}{\sqrt{n}}\sqrt{2\ln\left(\frac{2\Omega_2}{\delta}\right)}$$

If there exist l and k such that $x(l)=x(k)$, then although the decision maker made two different predictions $f(l)$ or $f(k)$, he played accordingly to the same probability $x(l)=x(k)$. Define $N_n(l,k)$ as the set of stages where the decision maker predicts either $f(l)$ or $f(k)$ up to stage n, $f_n(l,k)$ as the average flag on this set, $\rho_n(l,k)$ as the average payoff and $\mathcal{R}_n(l,k)$ as the regret. Since $W(x,\cdot)$ is convex for every $x\in\Delta(I)$, then $\max_{x\in\Delta(I)}W(x,\cdot)$ is also convex so

$$\frac{|N_n(l,k)|}{n}\max_{x\in\Delta(I)}W(x,f_n(l,k))\leq\frac{|N_n(l)|}{n}\max_{x\in\Delta(I)}W(x,f_n(l))+\frac{|N_n(k)|}{n}\max_{x\in\Delta(I)}W(x,f_n(k))$$

$$\text{and}\quad -\frac{|N_n(l,k)|}{n}\rho_n(l,k) = -\frac{|N_n(l)|}{n}\rho_n(l) - \frac{|N_n(k)|}{n}\rho_n(k)$$

so we still have

$$\frac{|N_n(l,k)|}{n}\mathcal{R}_n(l,k) \leq O\left(\sqrt{\frac{\ln\left(\frac{1}{\delta}\right)}{n}}\right).$$

Hence the previous bound holds up to a factor L. □

Remark 23 *Lugosi et al. (2008) have constructed an externally consistent strategy, that is, such that, asymptotically, for any strategy τ of Nature:*

$$\rho_n \geq \max_{z\in\Delta(I)} W\left(z, f_n\right), \quad \mathbb{P}_{\sigma,\tau}-\text{as.}$$

The final argument in the proof of Theorem 22 also implies that an internally consistent strategy is also externally consistent, hence we can compare bounds between our algorithm.

If the signals are deterministic, Lugosi et al. (2008)'s efficient algorithm has an expected regret smaller than $O\left(n^{-1/2}\right)$. However this bound became, with random signals, $O\left(n^{-1/4}\right)$. Thus our algorithm, along with computing no internal regret, has a better rate of convergence, the optimal one. Concerning the computational complexity, the true purpose of this algorithm being the minimization of internal regret, it is not efficient to bound external regret.

3.2.2 Action-Outcome Dependant Signals

In this section, we consider the most general framework and we assume that the laws of the signals might depend on the decision maker's actions. Our main result is the following:

Theorem 24 *There exists an internally consistent strategy σ such that, for every strategy τ of Nature, with $\mathbb{P}_{\sigma,\tau}$ probability at least $1-\delta$:*

$$\max_{l\in\mathcal{L}} \frac{|N_n(l)|}{n}\mathcal{R}_n(l) \leq O\left(\frac{1}{n^{1/3}}\sqrt{\ln\left(\frac{1}{\delta}\right)} + \frac{1}{n^{2/3}}\ln\left(\frac{1}{\delta}\right)\right).$$

Proof. The proof is essentially the same as the one of Theorem 22, so we can assume that $x(l) \neq x(k)$ for any two different l and k in $\mathcal{L}$. The only difference is due to the fact that at stage $n \in \mathbb{N}$, the unobserved flag f_n has to be estimated, see, for example, Lugosi et al. (2008).

Following Auer et al. (2002/03), we define for every $l \in \mathcal{L}$ and $n \in \mathbb{N}$, the $\widehat{\gamma}_n$-perturbation of $x(l)$ by $\widehat{x}(l,n) = (1-\widehat{\gamma}_n)x(l) + \widehat{\gamma}_n u$ where u is the uniform probability over I and $(\widehat{\gamma}_n)_{n\in\mathbb{N}}$ is a non-negative non-increasing sequence. For every $n \in \mathbb{N}$, let

$$e_n = \left(\frac{\mathbb{1}_{i=i_n}}{\widehat{x}(l_n,n)[i_n]}\left(\mathbb{1}_{s=s_n}\right)_{s\in\mathcal{S}}\right)_{i\in I} \in \left(\mathbb{R}^S\right)^I,$$

where $\widehat{x}(l_n,n)[i_n] \geq \gamma_n = \widehat{\gamma}_n/I > 0$ is the weight put by $\widehat{x}(l_n,n)$ on i_n. With this notation, e_n is an unbiased estimator of f_n since $\mathbb{E}_{\sigma,\tau}\left[e_n|h^{n-1}\right] = f_n$, seen as an element of $\left(\mathbb{R}^S\right)^I$.

We define now the strategy of the forecaster. Assume that in an auxiliary game Γ_c, a predictor computes $\widetilde{\sigma}$, a calibrated strategy with respect to $\{f(l), \omega(l);\ l \in \mathcal{L}\}$, but where the state at stage n is the estimator $e_n \in \mathbb{R}^{IS}$. When the decision maker (seen as a predictor) should choose l_n accordingly to $\widetilde{\sigma}$ in Γ_c, then he (seen as a forecaster) chooses i_n accordingly to $\widehat{x}(l_n)$ in the original game.

In order to use Corollary 14, we need to bound v_n, M_n and K_n. In the current framework and thanks to Proposition 11, one has for every $l,k \in \mathcal{L}$ and $n \in \mathbb{N}$:

$$U^{l,k}_{\omega,n} = 2\mathbb{1}_{l=l_n} \sum_{t \in \mathcal{T}} \frac{\sigma_t(k) - \sigma_t(l)}{T} \Big(\langle e_n, c_t \rangle + b_t \Big),$$

so using the fact that $\|(b,c)\|^2_\infty = 1$ and the definition of e_n:

$$\sup_{l,k \in \mathcal{L}} \sup_{m \leq n} \mathbb{E}_{\sigma,\tau} \left[\left| U^{l,k}_{\omega,m} \right|^2 \right] \leq 16 \mathbb{E}_{\sigma,\tau} \left[\|e_n\|^2 \right] \|(b,c)\|^2_\infty \leq 16 \sum_{i \in I} \frac{\widehat{x}(l_n,n)[i]}{(\widehat{x}(l_n,n)[i])^2} \leq 16 \frac{I}{\gamma_n}.$$

As a consequence, $K_n \leq 4\frac{1}{\gamma_n}$, $v_n \leq 4\sqrt{\frac{I}{\gamma_n}}$ and $M_n \leq 4\sqrt{\frac{LI}{\gamma_n}}$. Lemma 12 implies that, with $\mathbb{P}_{\sigma,\tau}$ probability at least $(1-\delta_1)$, for every $l \in \mathcal{L}$:

$$\frac{|N_n(l)|}{n} \left\| e_n(l) - \widetilde{f}_n(l) \right\| \leq \frac{8\sqrt{LI}M_P}{\sqrt{\gamma_n}n} + \frac{8\sqrt{I}M_P}{\sqrt{\gamma_n}n} \sqrt{2 \ln \left(\frac{L^2}{\delta_1} \right)} + \frac{8}{3} \frac{M_P}{\gamma_n n} \ln \left(\frac{L^2}{\delta_1} \right),$$

where $\widetilde{f}_n(l)$ is the projection of $e_n(l)$ onto $P(l)$.

Following Lugosi et al. (2008), since for every $i \in I$ and $s \in \mathcal{S}$, $\mathbb{E}_{\sigma,\tau} \left[|e^{i,s}_n|^2 \right] \leq 1/\gamma_n$, Freedman's inequality implies that with probability at least $1-\delta_2$, for every $l \in \mathcal{L}$

$$\frac{|N_n(l)|}{n} \left\| e_n(l) - f_n(l) \right\| \leq \sqrt{IS} \left(\sqrt{2 \frac{1}{n\gamma_n} \ln \left(\frac{2LIS}{\delta_2} \right)} + \frac{2}{3n\gamma_n} \ln \left(\frac{2LIS}{\delta_2} \right) \right).$$

Hoeffding-Azuma's inequality implies that with probability at least $1-\delta_3$:

$$\max_{l \in \mathcal{L}} \frac{N_n(l)}{n} \Big| \rho_n(l) - \rho(x(l), j_n(l)) \Big| \leq M_\rho \sqrt{\frac{2}{n} \ln \left(\frac{2L}{\delta_3} \right)} + 2M_\rho \frac{\sum_{m \in N_n(l)} \widehat{\gamma}_m}{n},$$

and by taking $\gamma_n = n^{-1/3}$, one has $\sum_{m \in N_n(l)} \widehat{\gamma}_m \leq \frac{3I}{2} n^{2/3}$. As a consequence, for every $l \in \mathcal{L}$, with probability at least $1-\delta$:

$$\frac{N_n(l)}{n} \mathcal{R}_n(l) \leq \frac{\Omega_1}{n^{1/3}} + \frac{\Omega_2}{n^{1/3}} \sqrt{2 \ln \left(\frac{2\Omega_5}{\delta} \right)} + \frac{\Omega_3}{n^{1/2}} \sqrt{2 \ln \left(\frac{2\Omega_5}{\delta} \right)} + \frac{2}{3} \frac{\Omega_4}{n^{2/3}} \ln \left(\frac{2\Omega_5}{\delta} \right)$$

with the constants defined by $\Omega_1 = 16 M_P M_W \sqrt{LI} + 3 M_W M_\rho I$, $\Omega_2 = 2M_W \sqrt{I} \left(8M_P + \sqrt{S} \right)$, $\Omega_3 = M_\rho$, $\Omega_4 = 2M_W (4M_P + \sqrt{IS})$ and $\Omega_5 = L(L + 2 + 2IS)$. They can be decreased if concentration inequalities in Hilbert spaces are used (see Section 4.3). □

In the label efficient prediction game defined in Example 1, for every strategy σ of the decision maker there exists a sequence of outcomes such that the forecaster expected regret is greater than $n^{-1/3}/7$, see Cesa-Bianchi et al. (2005, Theorem 5.1). Therefore the rate of $n^{-1/3}$ of our algorithm is optimal for both internal and external regret.

The computational complexity of this internally consistent algorithm is polynomial in L. Thus it can be seen, in some sense, as an efficient one. A question left open is the existence of an algorithm whose computational complexity is polynomial in the minimal number of best-response areas required to cover $\Delta(\mathcal{S})^I$, see Proposition 20.

The following Section 4.1 deals with a simpler question and exhibits an internally consistent algorithm which requires to solve at each stage a linear program of size polynomial in L_0, the minimal number of polytopes on which BR is constant, instead of a system of linear equations of size L.

4. Concluding Remarks

This section sheds light on some improvements of the computational complexity and constants of our algorithm and also on the possibility to remove the assumption that $\mathcal{J}$ is finite.

4.1 Second Algorithm: Calibration and Polytopial Complex

The algorithms we described are quite easy to run stage by stage since the forecaster only needs to compute some invariant measures of non-negative matrices. However, they require to construct the Laguerre diagram $\mathcal{P} = \{P(l);\ l \in \mathcal{L}\}$ given the set $\{b_t, c_t;\ t \in \mathcal{T}\}$. And we have shown that L, which is a factor both in the complexity of the algorithms and in their rate of convergence, can be in the order of T^{SI} hence polynomial in L_0^{SI}.

This section is devoted to a modification of the algorithm that does not require to compute a Laguerre diagram but which is more difficult, stage by stage, to implement. The only difference between the two algorithms is in the definition of calibration.

Let $\{K(l);\ l \in \mathcal{L}_0\}$ be a finite polytopial complex of $\Delta(\mathcal{J})$. It is defined by two finite families $\left\{c_t \in \mathbb{R}^J, b_t \in \mathbb{R};\ t \in \mathcal{T}\right\}$ and $\{\mathcal{T}(l) \subset \mathcal{T};\ l \in \mathcal{L}\}$ such that:

$$K(l) = \{y \in \Delta(\mathcal{J});\ \langle y, c_t\rangle \le b_t, \forall t \in \mathcal{T}(l) \subset \mathcal{T}\}, \quad \forall l \in \mathcal{L}_0 .$$

Let us define $(c_{t,l}, b_{t,l}) = (c_t, b_t)$ if $t \in \mathcal{T}(l)$ and $(c_{t,l}, b_{t,l}) = (0,0)$ otherwise. Then we can rewrite $K(l) = \{y \in \Delta(\mathcal{J});\ \langle y, c_{t,l}\rangle \le b_{t,l}, \forall t \in \mathcal{T}\}$.

Definition 25 *A strategy σ is calibrated with respect to the complex $\{K(l);\ l \in \mathcal{L}_0\}$ if for every strategy τ of Nature, $\mathbb{P}_{\sigma,\tau}$-as:*

$$\limsup_{n\to\infty} \frac{|N_n(l)|}{n}\Big(\langle j_n(l), c_{t,l}\rangle - b_{t,l}\Big) \le 0, \quad \forall t \in \mathcal{T}, \forall l \in \mathcal{L}_0 .$$

Theorem 26 *There exist calibrated strategies with respect to any finite polytopial complex $\{K(l);\ l \in \mathcal{L}_0\}$.*

Proof. Consider the following auxiliary two-person game Γ'_c, where at stage $n \in \mathbb{N}$ the predictor (resp. Nature) chooses $l_n \in \mathcal{L}_0$ (resp. $j_n \in \mathcal{J}$) which generates the vector payoff $U_n \in \mathbb{R}^{TL_0}$ defined by:

$$U_n^{lk} = \begin{cases} \langle \mathbb{1}_{j_n=j}, c_{t,l}\rangle - b_{t,l} & \text{if } l = l_n \\ 0 & \text{otherwise.} \end{cases} .$$

Any strategy that approaches the negative orthant Ω_- in Γ'_c is calibrated with respect to the complex $\{K(l);\, l \in \mathcal{L}_0\}$.

Blackwell's characterization of approachable convex sets (Blackwell, 1956a, Theorem 3) implies that the predictor can approach the convex set Ω_- if (and only if) for every mixed action of Nature in $\Delta(\mathcal{J})$, he has an action $x \in \Delta(\mathcal{L}_0)$ such that the expected payoff is in Ω_-. Given $y_n \in \Delta(\mathcal{J})$, choosing $l(y_n) \in \mathcal{L}_0$, where $l(y_n)$ is the index of the polytope that contains y_n, ensures that $\mathbb{E}_{y_n,l(y_n)}[U_n]$ is in Ω_-. Therefore there exist calibrated strategies with respect to any polytopial complex. $\square$

This modification of the definition of calibration does not change the other part of our algorithms nor the remaining of the proofs (in particular, to calibrate the sequence of unobserved flags, the forecaster must use $\widehat{\gamma}_n$-perturbations). The constants in the rates of convergence are now smaller since L_0 can be much smaller than L and in Γ'_c, $\mathbb{E}[\|U_n\|^2]$ is bounded by $O\left(\frac{T_0}{\gamma_n}\right)$ where $T_0 = \sup_{l \in \mathcal{L}_0} T(l)$ is the maximum number of hyperplanes defining a polytope of the complex.

The main argument behind this algorithm (i.e., the characterization of approachable convex sets of Blackwell, 1956a) is quite close, in spirit, to the one of Lehrer and Solan (2007). Note that however, with our representation, the projection on Ω_- can be computed linearly in TL_0, so polynomially in L_0. Therefore, it reduces to the construction of an approachability strategy and so, as shown by Blackwell (1956a), to the resolution, at each stage, of a linear programming of size polynomial in L_0.

4.2 Extension to the Compact Case

We prove in this section that the finiteness of $\mathcal{J}$ is not required.

Assume that instead of choosing j_n at stage $n \in \mathbb{N}$, which generates the flag $f_n = \mathbf{s}(j_n)$ and an outcome vector $\left(\rho(i, j_n)\right)_{i \in I}$, Nature chooses directly an outcome vector $O_n \in [-1,1]^I$ and a flag f_n which belongs to $\mathbf{s}(O_n)$ where $\mathbf{s}$ is a multivalued mapping from $[-1,1]^I$ into $\Delta(\mathcal{S})^I$. As before, the decision maker's payoff is $O_n^{i_n}$ (the i_n-th coordinate of O_n) and he receives a signal s_n whose law is $f_n^{i_n}$. Strategies of the forecaster and consistency are defined as before.

Theorem 27 *If the graph of* $\mathbf{s}$ *is a polytope, then there exists an internally consistent strategy* σ *such that, for every strategy* τ *of Nature, with* $\mathbb{P}_{\sigma,\tau}$ *probability at least* $1-\delta$*:*

$$\max_{l \in \mathcal{L}} \frac{|N_n(l)|}{n} \mathcal{R}_n(l) \leq O\left(\frac{1}{n^{1/3}}\sqrt{\ln\left(\frac{1}{\delta}\right)} + \frac{1}{n^{2/3}}\ln\left(\frac{1}{\delta}\right)\right).$$

The proof of this result is identical to the one of Theorem 24.

Note that the assumption that the graph of $\mathbf{s}$ is a polytope is fulfilled in the finite dimension case. The mapping $\mathbf{s}$ is multivalued since in finite dimension there might exist two different mixed actions y_1, y_1 in $\Delta(\mathcal{J})$ that generate the same outcome vector (i.e., $\rho(\cdot, y_1) = \rho(\cdot, y_2) = O$) but different flags (i.e., $f_1 = \mathbf{s}(y_1) \neq \mathbf{s}(y_2) = f_2$). Hence we should have $f_1, f_2 \in \mathbf{s}(O)$.

4.3 Strengthening of the Constants

We propose two different ideas to strengthen the constants of our algorithm. First, we can use (as did Lugosi et al., 2008) only one concentration inequality for every coordinate of the vector $U_{\omega,n}$

instead of one concentration inequality per coordinate. Second, we can implement sparser vector payoffs (so that its norm decreases) by looking at a slight different definition of calibration.

4.3.1 CONCENTRATION INEQUALITIES IN HILBERT SPACES

The rates of convergence of our algorithms rely mainly on three properties: Blackwell's approachability theorem, Hoeffding-Azuma's and Freedman's inequalities. These tools allowed us to study the convergence of a sequence of vectors U_n^+ towards 0. Approachability is well defined for sequences of vectors, however the two concentration inequalities hold only for real valued martingales. To circumvent this issue, we used in the proofs the fact that if a process $\left\{U_n \in \mathbb{R}^d\right\}_{n\in\mathbb{N}}$ is a martingale then, for each coordinate, the process $\left\{U_n^k \in \mathbb{R}\right\}_{n\in\mathbb{N}}$ is a real valued martingale. This does not use the fact that U_n might be sparse and the use of concentration inequalities in Hilbert space can sharpen the constant.

Indeed, recall Hoeffding-Azuma's inequality:

Lemma 28 (Hoeffding, 1963; Azuma, 1967) *Let U_n be a sequence of martingale differences bounded by K, that is, for every $n \in \mathbb{N}$, $\mathbb{E}_{\sigma,\tau}[U_{n+1}|h_n] = 0$ and $|U_n| < K$.*

Then for every $n \in \mathbb{N}$ and every $\varepsilon > 0$:

$$\mathbb{P}_{\sigma,\tau}\left(|U_n| \geq \varepsilon\right) \leq 2\exp\left(\frac{-n\varepsilon^2}{2K^2}\right),$$

which can be expressed as

$$\mathbb{P}_{\sigma,\tau}\left(|U_n| \leq K\sqrt{\frac{2}{n}\ln\left(\frac{2}{\delta}\right)}\right) \geq 1-\delta.$$

Chen and White (1996) proved an equivalent property for vector martingale in $\mathbb{R}^d$.

Lemma 29 (Chen and White, 1996) *Let U_n be a sequence of martingale differences in $\mathbb{R}^d$ bounded almost-surely by $K > 0$. Then for every $n \in \mathbb{N}$ and for every $\varepsilon > 0$:*

$$\mathbb{P}_{\sigma,\tau}\left(\|U_n\| \geq \varepsilon\right) \leq 2\max\left\{1, \sqrt{\frac{n\varepsilon^2}{2K^2}}\right\}\exp\left(\frac{-n\varepsilon^2}{2K^2}\right) \leq 2\exp\left(-\alpha\frac{n\varepsilon^2}{2K^2}\right),$$

for every $\alpha \leq 1 - \frac{1}{2e}$ (which equals approximatively 0.81).

Assume that for every $n \in \mathbb{N}$, $\|U_n\|_\infty \leq \|U\|_\infty$ and $\|U_n\|_2 \leq \|U\|_2$; we can deduce from the use of only Hoeffding-Azuma's inequality that:

$$\mathbb{P}_{\sigma,\tau}\left(\max_{l,k}\frac{|N_n(l)|}{n}\left|U_n^{l,k}\right| \geq \varepsilon\right) \leq 2L^2\exp\left(\frac{-n\varepsilon^2}{2\|U\|_\infty^2}\right).$$

However, Chen and White's result, along with the fact that $\|U_n\| \leq L$, implies that:

$$\mathbb{P}_{\sigma,\tau}\left(\max_{l,k}\frac{|N_n(l)|}{n}\left|U_n^{l,k}\right| \geq \varepsilon\right) \leq 2\exp\left(\frac{-n\varepsilon^2}{4\|U\|_2^2}\right)$$

which can reduce the dependency in L. The effects is even more dramatic when estimating the sequences of flags, since e_n has only positive component (so $\|e_n\|_\infty = \|e_n\|_2$).

There also exist variants of Bernstein's inequality, see, for example, Yurinskii (1976) in Hilbert spaces that can be used in order to get more precise constants.

4.3.2 CALIBRATION WITH RESPECT OF NEIGHBORHOODS

Definition 30 *Given a finite set $\mathcal{Y} = \{y(l) \in \mathbb{R}^d, \omega(l) \in \mathbb{R};\ l \in \mathcal{L}\}$, $y(k)$ is a neighbor of $y(l)$ if $k \neq l$ and the dimension of $P(l) \cap P(k)$ is equal to $d-1$.*

We defined a calibrated strategy with respect to $\mathcal{Y}$, as a strategy σ such that $j_n(l)$ is asymptotically closer to $y(l)$ than to any other $y(k)$ as soon as the frequency of l does not go to zero. In fact, $j_n(l)$ needs only to be closer to $y(l)$ than to any of its neighbors. So one can construct *neighbors*-calibrated strategies by modifying the algorithm given in Proposition 5; the payoff at stage n is now denoted by U'_n and is defined by:

$$\left(U'_n\right)^{lk} = \begin{cases} \|j_n - y(l)\|^2 - \|j_n - y(k)\|^2 & \text{if } l = l_n \text{ and } k \text{ is a neighbor of } l \\ 0 & \text{otherwise} \end{cases}.$$

The strategy consisting in choosing an invariant measure of $\left(U'_n\right)^+$ is calibrated and the squared maximal second order moment $M_n^2 = \sup_{m \leq n} \mathbb{E}_{\sigma,\tau}\left[\|U_m\|^2\right]$ equals $4\mathcal{N}$, where $\mathcal{N}$ is the maximal number of neighbors. This latter can be much smaller than 4, and the gain from this modification is limpid if we consider ε-calibration.

Indeed, in order to construct such strategies, we usually take any ε-discretization of $\Delta(J)$ so that $L = O\left(\varepsilon^{-(J-1)}\right)$. However, there exists a discretization such that $\mathcal{N} = 2^{-(J-1)}$, which is independent of ε.

Acknowledgments

I deeply thank my PhD advisor Sylvain Sorin for its great help and support. I also acknowledge very useful comments of Gilles Stoltz. This paper was supported by ANR under grant ANR-10-BLAN-0112.

Appendix A. Proofs of Technical Results

This section is devoted to the proofs of previously mentioned results, that is, Lemma 12 and Proposition 20.

A.1 Proof of Lemma 12

Let $l \in \mathcal{L}$ be fixed. we denote by $\mathcal{C} = \left\{c_t \in \mathbb{R}^d;\ t \in \mathcal{T}(l)\right\}$ the finite family of normal vectors to $(d-1)$-faces of $P(l)$ and by $\mathcal{B} = \{b_t \in \mathbb{R};\ t \in \mathcal{T}(l)\}$ the family of scalars such that :

$$P(l) = \left\{Z \in \mathbb{R}^d;\ \langle Z, c_t \rangle \leq b_t, \forall t \in \mathcal{T}(l)\right\}.$$

Any points satisfying Equation (3) belongs to

$$P_\varepsilon(l) = \left\{Z \in \mathbb{R}^d;\ \langle Z, c_t \rangle \leq b_t + \varepsilon, \forall t \in \mathcal{T}(l)\right\}.$$

For any vertex v of $P(l)$, there exists $t_1, .., t_d \in \mathcal{T}(l)$ such that

$$v = \bigcap_{k=1}^{d} \left\{Z \in \mathbb{R}^d;\ \langle Z, c_{t_k} \rangle = b_{t_k}\right\}$$

and $\{c_{t_1},..,c_{t_d}\}$ is a basis of $\mathbb{R}^d$. If we denote by v_ε the point defined by

$$v_\varepsilon = \bigcap_{k=1}^{d} \left\{ Z \in \mathbb{R}^d;\ \langle Z, c_{t_k} \rangle = b_{t_k} + \varepsilon \right\}$$

then $P_\varepsilon(l)$ is included in the convex hull of every v_ε.

Equation (3) can be rephrased as: if x belongs to $P_\varepsilon(l)$ then $d(x, P(l))$ is smaller than $M_P\varepsilon$. Therefore it is enough to prove this property for every v_ε since $d(\cdot, P(l))$ is a convex mapping thus maximized over a polytope on one of its vertices.

With these notations, for every $k \in \{1,..,d\}$, $\langle v_\varepsilon - v, c_{t_k} \rangle = \varepsilon$ and there exists a unique decomposition $v_\varepsilon - v = \sum_{k=1}^{d} \alpha_k c_{t_k}$. Define the symmetric $d \times d$ Gram matrix Q_l by $Q_l^{kk'} = \langle c_{t_k}, c_{t_{k'}} \rangle$ and $\alpha = (\alpha_1,..,\alpha_d)$. Then following classical properties hold:

1) $\|v_\varepsilon - v\|^2 = \alpha^T Q_l \alpha$ and there exist a diagonal matrix $D = diag(\lambda_1,..,\lambda_d)$ with $0 < \lambda_1 \leq .. \leq \lambda_d$ and a $d \times d$ matrix P and such that $P^{-1} = P^T$ and $Q_l = P^T D P$;

2) $Q\alpha = \underline{\varepsilon} = (\varepsilon,..,\varepsilon)$ therefore $\alpha = Q_l^{-1}\underline{\varepsilon}$;

3) $\|v_\varepsilon - v\|^2 = (Q_l^{-1}\underline{\varepsilon})^T Q_l (Q_l^{-1}\underline{\varepsilon}) = \underline{\varepsilon}^T P^T D^{-1} P \underline{\varepsilon} \leq \varepsilon^2 d \lambda_1^{-1}$.

Therefore, for any $Z \in P_\varepsilon$, and in particular for any point that satisfies Equation (3), $\|Z - \Pi_l(Z)\| \leq \max_v \|v_\varepsilon - v\| \leq \varepsilon.\sqrt{d}\sqrt{\lambda_1}^{-1}$. The result follows from the fact that L is finite. The constant M_P in Lemma 12 is smaller than the square root of the inverse of the smallest eigenvalue of all Q_l times $\sqrt{d}$; it depends on the inner products $\langle c_t, c_{t'} \rangle$ and on the dimension of $\mathcal{F}$.

A.2 Proof of Proposition 20

Definition 31 *Let K be a polytope. A correspondence $B : K \rightrightarrows \mathbb{R}^d$ is polytopial constant, if there exists $\{K(l);\ l \in \mathcal{L}\}$ a finite polytopial complex of K and $\{x(l);\ l \in \mathcal{L}\}$ such that $x(l) \in B(f)$ for every $f \in K(l)$.*

Let us now restate Proposition 20:

Proposition 32 *BR is polytopial constant.*

This theorem is well-known and quite useful in the full monitoring case (see for example the Lemke and Howson, 1964 algorithm). In the *compact case*, Proposition 20 becomes:

Proposition 33 *If* **s** *has a polytopial graph, then BR is polytopial constant.*

The proofs of both propositions rely on polytopial parameterized max-min programs defined in the next subsection.

A.2.1 Constant Solution of a Polytopial Parameterized Max-Min Program

A Polytopial Parameterized Max-Min Program (PPMP) is defined as follows. Let $\mathcal{X}$ and $\mathcal{Y}$ be two Euclidian spaces of respective dimension d_1 and d_2. Consider the program (P_f) (depending on a parameter f that belongs to some polytope $\mathcal{F}$ in $\mathbb{R}^{d_3}$) that is defined by

$$(P_f): \quad \max_{\substack{x \in \mathcal{X} \\ s.t.\ Dx \leq d}} \quad \min_{\substack{y \in \mathcal{Y} \\ s.t.\ E_f y \leq e_f}} \quad xAy,$$

where A is a $d_1 \times d_2$ matrix, $\{E_f, e_f;\ f \in \mathcal{F}\}$ is a family of matrices and vectors (we do not specify the sizes the matrices, as long as each inequality makes sense) and D, d are also a fixed matrix and vector such that the admissible set $\mathcal{D} = \{x \in \mathcal{X};\ Dx \leq d\}$ is a polytope. The solution set of (P_f) is denoted by $B(f) \subset \mathcal{X}$ and this defines a multivalued mapping $B(\cdot)$ from $\mathcal{F}$ into $\mathcal{X}$.

Theorem 34 *Assume that the correspondence S defined by:*

$$S: \begin{array}{ccc} \mathcal{F} & \rightrightarrows & \mathcal{Y} \\ f & \mapsto & S_f = \{y \in \mathcal{Y};\ E_f y \leq e_f\} \end{array}$$

has a polytopial graph $\mathbf{S}$. Then $B : \mathcal{F} \rightrightarrows \mathcal{X}$ is polytopial constant.

Figure 1 illustrates ideas of the proof for a simple example.

Proof. Before going into full details, we first recall the following properties:

i) A linear program is minimized on a vertex of the polytopial feasible set (this is actually implied by the following point);

ii) Rockafellar (1970, Theorem 27.4, page 270): Given $x \in \mathcal{D}$ and $f \in \mathcal{F}$, if y minimizes xAy on S_f then

$$-xA \in NC_{S_f}(y)\,,$$

where $NC_E(y)$ is the normal cone to the convex set $E \subset \mathbb{R}^d$ at $y \in E$ defined by :

$$NC_E(y) = \left\{p \in \mathbb{R}^d;\ \langle p, z-y \rangle, \forall z \in E\right\}\,;$$

iii) Ziegler (1995, Example 7.3, page 193): If P is a polytope then the finite family $\{NC_P(v);\ v \text{ is a vertex of } P\}$ is a polyhedral complex of $\mathbb{R}^d$ called a normal fan (i.e., it is a finite family of polyhedra that cover $\mathbb{R}^d$ and such that each pair has an intersection with empty interior);

iv) Billera and Sturmfels (1992, page 530): Since for every $f \in \mathcal{F}$, $S_f = \Pi^{-1}(f)$ where $\Pi : \mathbf{S} \subset \mathcal{F} \times \mathcal{Y} \to \mathcal{F}$ is the projection with respect to first coordinates, then there exists $\{K(l);\ l \in \mathcal{L}\}$, a polytopial complex of $\mathcal{F}$ such that the normal fan to S_f is constant on every $K(l)$ (this can alternatively be deduced from the following point);

v) Rambau and Ziegler (1996, Proposition 2.4, page 221): On each of these polytopes $K(l)$, the mapping $f \mapsto S_f$ is linear. In particular, there exists a finite family of affine functions $Y(l)$ from $K(l)$ to $\mathcal{Y}$ such that the vertices of S_f are exactly $\{y(f);\ y(\cdot) \in Y(l)\}$.

Points i) and ii) imply that if x_f maximizes (P_f), then the latter is minimized at some vertex of S_f denoted by y_f, again because of point i). Therefore it can be assumed that $-x_f A$ is a vertex of the polytope $NC_{S_f}(y_f) \cap \mathcal{D}_{A-}$ where $\mathcal{D}_{A-} := \{-xA;\ x \in \mathcal{D}\}$. Thus $B(f)$, the solution set to (P_f) contains at least an element of

$$\mathbf{X}_f = \left\{x \in \mathcal{D};\ -xA \text{ is a vertex of } \mathcal{D}_{A-} \cap NC_{S_f}(y_f), \text{for some vertex } y_f \text{ of } S_f\right\}\,.$$

By point iii), the normal fan and therefore $\mathbf{X}_f$ are constant on $K(l)$. The latter can also be assumed to be finite by taking a unique representant $x \in \mathbf{X}_f$ for every vertices of the intersection of the normal fan and $\mathcal{D}_{A-}$. Since the number of different fans is finite, for any $f \in \mathcal{F}$, the solution set to (P_f) contains at least an element of the finite set $\mathbf{X} = \bigcup_{f\in\mathcal{F}} \mathbf{X}_f$.

Moreover, for every $\mathbf{x} \in \mathbf{X}$:

$$\begin{aligned}
B^{-1}(\mathbf{x}) &= \left\{ f \in \mathcal{F};\ \min_{y\in S_f} \mathbf{x}Ay \geq \max_{x'\in\mathcal{D}} \min_{y\in S_f} x'Ay \right\} \\
&= \bigcup_{l\in\mathcal{L}} \left\{ f \in K(l);\ \min_{y\in S_f} \mathbf{x}Ay \geq \max_{x'\in\mathcal{D}} \min_{y\in S_f} x'Ay \right\} \\
&= \bigcup_{l\in\mathcal{L}} \bigcap_{\mathbf{x}'\in\mathbf{X}} \left\{ f \in K(l);\ \min_{y\in S_f} \mathbf{x}Ay \geq \min_{y\in S_f} \mathbf{x}'Ay \right\} \\
&= \bigcup_{l\in\mathcal{L}} \bigcap_{\mathbf{x}'\in\mathbf{X}} \bigcup_{y'(\cdot)\in Y(l)} \left\{ f \in K(l);\ \min_{y\in S_f} \mathbf{x}Ay \geq \mathbf{x}'Ay'(f) \right\} \\
&= \bigcup_{l\in\mathcal{L}} \bigcap_{\mathbf{x}'\in\mathbf{X}} \bigcup_{y'(\cdot)\in Y(l)} \bigcap_{y(\cdot)\in Y(l)} \left\{ f \in K(l);\ \mathbf{x}Ay(f) \geq \mathbf{x}'Ay'(f) \right\},
\end{aligned}$$

where, respectively, the second line is a consequence of point iv), the third line of the definition of $\mathbf{X}$ and the fourth and fifth lines of points i) and v).

By point v), the two mapping $y(\cdot)$ and $y'(\cdot)$ are affine on $K(l)$, so each possible set

$$\left\{ f \in K(l);\ \mathbf{x}Ay(f) \geq \mathbf{x}'Ay'(f) \right\}$$

is a polytope as the intersection of an half-space and the polytope $K(l)$. Since, the intersection of a union of polytopes remains a union of polytopes, for every $\mathbf{x} \in \mathbf{X}$, $B^{-1}(\mathbf{x})$ is a finite union of polytopes and B is polytopial constant. □

We can now prove simultaneously Propositions 32 and 33:

A.2.2 Proof of Propositions 32 and 33

Since $\mathbf{s}$ is linear, its graph, denoted by $\mathbf{S}$, is a polytope. Theorem 34 (with $\mathcal{D} = \Delta(I)$) implies that the solution, denoted by $B(f)$ for every $f \in \mathcal{F}$, of the parameterized program

$$\max_{x\in\Delta(I)} \min_{y\in\mathbf{s}^{-1}(f)} \rho(x,y)$$

is polytopial constant. We denote by $\{K(l);\ l \in \mathcal{L}\}$ a corresponding polytopial complex. If B is constant on $K(l)$, then it is also constant on $\widehat{K}(l) = \Pi_{\mathbf{S}}^{-1}(K(l))$, which is a finite union of polytopes. □

References

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32:48–77 (electronic), 2002/03.

F. Aurenhammer. A criterion for the affine equivalence of cell complexes in $\mathbb{R}^d$ and convex polyhedra in $\mathbb{R}^{d+1}$. *Discrete Comput. Geom.*, 2:49–64, 1987.

K. Azuma. Weighted sums of certain dependent random variables. *Tôhoku Math. J. (2)*, 19:357–367, 1967.

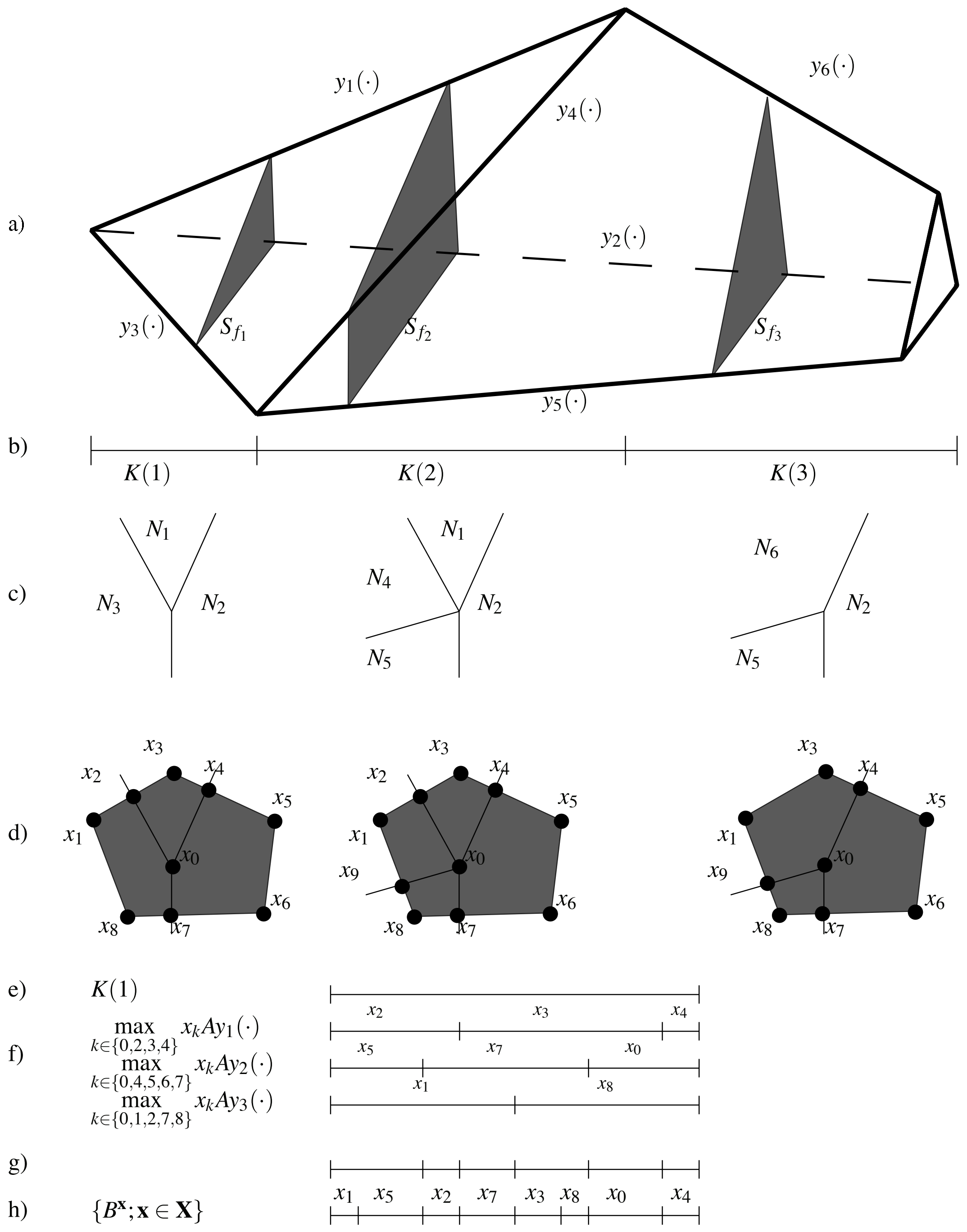

a)
$y_1(\cdot)$
$y_2(\cdot)$
$y_3(\cdot)$
$y_4(\cdot)$
$y_5(\cdot)$
$y_6(\cdot)$
S_{f_1}
S_{f_2}
S_{f_3}
b)
$K(1)$
$K(2)$
$K(3)$
c)
N_1 N_2 N_3
N_1 N_2 N_4 N_5
N_6 N_2 N_5
d)
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9
x_0 x_1 x_3 x_4 x_5 x_6 x_7 x_8 x_9
e)
$K(1)$
f)
$\max_{k\in\{0,2,3,4\}} x_kAy_1(\cdot)$
x_2 x_3 x_4
$\max_{k\in\{0,4,5,6,7\}} x_kAy_2(\cdot)$
x_5 x_7 x_0
$\max_{k\in\{0,1,2,7,8\}} x_kAy_3(\cdot)$
x_1 x_8
g)
h)
$\{B^{\mathbf{x}};\mathbf{x}\in\mathbf{X}\}$
x_1 x_5 x_2 x_7 x_3 x_8 x_0 x_4

Figure 1: Construction of $\mathbf{X}$ and the complex. From top to bottom:
a) The graph $\mathbf{S}$, with $\mathcal{F}\subset\mathbb{R}$ and some S_f in shaded.
b) The polytopial complex with constant normal fan (c) and $\mathbf{X}_f$ (d).
f) On each line, $y_i(\cdot)$ is a minimizer of $\min_{y\in S_f} x_kAy$.
g) On the first cell, $\max\limits_{x\in\mathcal{D}}\min\limits_{y\in S_f} xAy = \max\{x_2Ay_1(f);x_5Ay_2(f);x_1Ay_3(f)\}$.
h) The final polytopial complex of $K(1)$.

L. J. Billera and B. Sturmfels. Fiber polytopes. *The Annals of Mathematics*, 135(3):pp. 527–549, 1992.

D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific J. Math.*, 6:1–8, 1956a.

D. Blackwell. Controlled random walks. In *Proceedings of the International Congress of Mathematicians, 1954, Amsterdam, vol. III*, pages 336–338, 1956b.

A. Blum and Y. Mansour. From external to internal regret. *J. Mach. Learn. Res.*, 8:1307–1324 (electronic), 2007.

R. C. Buck. Partition of space. *Amer. Math. Monthly*, 50:541–544, 1943.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, 2006.

N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Minimizing regret with label efficient prediction. *IEEE Trans. Inform. Theory*, 51:2152–2162, 2005.

X. Chen and H. White. Laws of large numbers for Hilbert space-valued mixingales with applications. *Econometric Theory*, 12:284–304, 1996.

A. P. Dawid. The well-calibrated Bayesian. *J. Amer. Statist. Assoc.*, 77:605–613, 1982.

D. P. Foster and R. V. Vohra. Calibrated learning and correlated equilibrium. *Games Econom. Behav.*, 21:40–55, 1997.

D. P. Foster and R. V. Vohra. Asymptotic calibration. *Biometrika*, 85:379–390, 1998.

D. A. Freedman. On tail probabilities for martingales. *Ann. Probability*, 3:100–118, 1975.

D. Fudenberg and D. K. Levine. Conditional universal consistency. *Games Econom. Behav.*, 29: 104–130, 1999.

J. Hannan. Approximation to Bayes risk in repeated play. In *Contributions to the Theory of Games*, volume 3 of *Annals of Mathematics Studies*, pages 97–139. Princeton University Press, Princeton, N. J., 1957.

S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963.

T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.*, 11:1563–1600, 2010.

E. Lehrer and E. Solan. Learning to play partially-specified equilibrium. *manuscript*, 2007.

C. E. Lemke and J. T. Howson, Jr. Equilibrium points of bimatrix games. *J. Soc. Indust. Appl. Math.*, 12:413–423, 1964.

G. Lugosi, S. Mannor, and G. Stoltz. Strategies for prediction under imperfect monitoring. *Math. Oper. Res.*, 33:513–528, 2008.

V. Perchet. Calibration and internal no-regret with random signals. *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, pages 68–82, 2009.

J. Rambau and G. M. Ziegler. Projections of polytopes and the generalized Baues conjecture. *Discrete Comput. Geom.*, 16:215–237, 1996.

R. T. Rockafellar. *Convex Analysis*. Princeton Mathematical Series, No. 28. Princeton University Press, Princeton, N.J., 1970.

A. Rustichini. Minimizing regret: the general case. *Games Econom. Behav.*, 29:224–243, 1999.

E. Seneta. *Nonnegative Matrices and Markov Chains*. Springer Series in Statistics. Springer-Verlag, New York, second edition, 1981.

S. Sorin. Supergames. In *Game theory and applications (Columbus, OH, 1987)*, Econom. Theory Econometrics Math. Econom., pages 46–63. Academic Press, San Diego, CA, 1990.

S. Sorin. *Lectures on Dynamics in Games*. Unpublished Lecture Notes, 2008.

V. Yurinskii. Exponential inequalities for sums of random vectors. *Journal of Multivariate Analysis*, 6:473 – 499, 1976.

G. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995.

Journal of Machine Learning Research 12 (2011) 1923-1953 Submitted 9/09; Revised 1/11; Published 6/11

Dirichlet Process Mixtures of Generalized Linear Models

Lauren A. Hannah LH140@DUKE.EDU
Department of Statistical Science
Duke University
Durham, NC 27708, USA

David M. Blei BLEI@CS.PRINCETON.EDU
Department of Computer Science
Princeton University
Princeton, NJ 08544, USA

Warren B. Powell POWELL@PRINCETON.EDU
Department of Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544, USA

Editor: Carl Edward Rasmussen

Abstract

We propose Dirichlet Process mixtures of Generalized Linear Models (DP-GLM), a new class of methods for nonparametric regression. Given a data set of input-response pairs, the DP-GLM produces a global model of the joint distribution through a mixture of local generalized linear models. DP-GLMs allow both continuous and categorical inputs, and can model the same class of responses that can be modeled with a generalized linear model. We study the properties of the DP-GLM, and show why it provides better predictions and density estimates than existing Dirichlet process mixture regression models. We give conditions for weak consistency of the joint distribution and pointwise consistency of the regression estimate.

Keywords: Bayesian nonparametrics, generalized linear models, posterior consistency

1. Introduction

In this paper, we examine the general regression problem. The general regression problem models a response variable Y as dependent on a set of covariates x,

$$Y \,|\, x \sim f(m(x)).$$

The function $m(x)$ is the *mean function*, which maps the covariates to the conditional mean of the response; the distribution f characterizes the deviation of the response from its conditional mean. The simplest example is linear regression, where $m(x)$ is a linear function of x, and f is a Gaussian distribution with mean $m(x)$ and fixed variance.

Generalized linear models (GLMs) extend linear regression to many types of response variables (McCullagh and Nelder, 1989). In their canonical form, a GLM assumes that the conditional mean of the response is a linear function of the covariates, and that the response distribution is in an exponential family. Many classical regression and classification methods are GLMs, including logistic regression, multinomial regression, and Poisson regression.

The GLM framework makes two assumptions about the relationship between the covariates and the response. First, the covariates enter the distribution of the response through a linear function; a non-linear function may be applied to the output of the linear function, but only one that does not depend on the covariates. Second, the variance of the response cannot depend on the covariates. Both these assumptions can be limiting—there are many applications where we would like the response to be a non-linear function of the covariates or where our uncertainty around the response might depend on the covariates. In this paper, we develop a general regression algorithm that relaxes both of these assumptions. Our method captures arbitrarily shaped response functions and heteroscedasticity, that is, the property of the response distribution where both its mean and variance change with the covariates, while still retaining the flexibility of GLMs.

Our idea is to model the mean function $m(x)$ by a mixture of simpler "local" response distributions $f_i(m_i(x))$, each one applicable in a region of the covariates that exhibits similar response patterns. To handle multiple types of responses, each local regression is a GLM. This means that each $m_i(x)$ is a linear function, but a non-linear mean function arises when we marginalize out the uncertainty about which local response distribution is in play. (See Figure 1 for an example with one covariate and a continuous response function.) Furthermore, our method captures heteroscedasticity: the variance of the response function can vary across mixture components and, consequently, varies as a function of the covariates.

Finally, we use a Bayesian nonparametric mixture model to let the data determine both the number and form of the local mean functions. This is critical for modeling arbitrary response distributions: complex response functions can be constructed with many local functions, while simple response functions need only a small number. Unlike frequentist nonparametric regression methods, for example, those that create a mean function for each data point, the Bayesian nonparametric approach uses only as complex a model as the data require. Moreover, it produces a generative model. It can be used to infer properties other than the mean function, such as the conditional variance or response quantiles.

Thus, we develop *Dirichlet process mixtures of generalized linear models* (DP-GLMs), a regression tool that can model many response types and many response shapes. DP-GLMs generalize several existing Bayesian nonparametric regression models (Müller et al., 1996; Shahbaba and Neal, 2009) to a variety of response distributions. We derive Gibbs sampling algorithms for fitting and predicting with DP-GLMs. We investigate some asymptotic properties, including weak consistency of the joint density estimate and consistency of the regression estimate. We study DP-GLMs with several types of data.

The paper is organized as follows. In Section 2, we review the current research on Bayesian nonparametric regression and discuss how the DP-GLM extends this field. In Section 3, we review Dirichlet process mixture models and generalized linear models. In Section 4, we construct the DP-GLM and derive algorithms for posterior computation. In Section 5 we give general conditions for weak consistency of the joint density model and consistency of the regression estimate; we give several models where the conditions hold. In Section 6 we study the DP-GLM and other methods on three data sets; our study illustrates that the DP-GLM provides a powerful nonparametric regression model that can be used in many types of data analysis.

2. Related Work

Existing methods for Bayesian nonparametric regression include Gaussian processes (GP), Bayesian regression trees, and Dirichlet process mixtures.

GP priors assume that the observations arise from a Gaussian process model with known covariance function form (Rasmussen and Williams, 2006). GPs are can model many response types, including continuous, categorical, and count data (Rasmussen and Williams, 2006; Adams et al., 2009). With the proper choice of covariance function, GPs can handle continuous and discrete covariates (Rasmussen and Williams, 2006; Qian et al., 2008). GPs assume that the response exhibits a constant covariance; this assumption is relaxed with Dirichlet process mixtures of GPs (Rasmussen and Ghahramani) or treed GPs (Gramacy and Lee, 2008).

Regression tree models, such as classification and regression trees (CART) (Brieman et al., 1984), are a natural way to handle regression with continuous, categorical or mixed data. They split the data into a fixed, tree-based partitioning and fit a regression model within each leaf of the tree. Bayesian regression trees place a prior over the size of the tree and can be viewed as an automatic bandwidth selection method for CART (Chipman et al., 1998). Bayesian trees have been expanded to include linear models (Chipman et al., 2002) and GPs (Gramacy and Lee, 2008) in the leaf nodes.

The Dirichlet process has been applied to regression problems. West et al. (1994), Escobar and West (1995) and Müller et al. (1996) used joint Gaussian mixtures for continuous covariates and response. Rodriguez et al. (2009) generalized this method using dependent DPs, that is, Dirichlet processes with a Dirichlet process prior on their base measures, in a setting with a response defined as a set of functionals. However, regression by a joint density estimate poses certain challenges. The balance between fitting the response and the covariates, which often outnumber the response, can be slanted toward fitting the covariates at the cost of fitting the response.

To avoid these issues—which amount to over-fitting the covariate distribution and under-fitting the response—some researchers have developed methods that use local weights on the covariates to produce local response DPs. This has been achieved with kernels and basis functions (Griffin and Steel, 2010; Dunson et al., 2007), GPs (Gelfand et al., 2005) and general spatial-based weights (Griffin and Steel, 2006, 2010; Duan et al., 2007). Still other methods, again based on dependent DPs, capture similarities between clusters, covariates or groups of outcomes, including in non-continuous settings (De Iorio et al., 2004; Rodriguez et al., 2009). The method presented here is equally applicable to the continuous response setting and tries to balance its fit of the covariate and response distributions by introducing local GLMs—the clustering structure is based on both the covariates and how the response varies with them.

There is less research about Bayesian nonparametric models for other response types. Mukhopadhyay and Gelfand (1997) and Ibrahim and Kleinman (1998) used a DP prior for the random effects portion of a GLM. Likewise, Amewou-Atisso et al. (2003) used a DP prior to model arbitrary symmetric error distributions in a semi-parametric linear regression model. These methods still maintain the assumption that the covariates enter the model linearly and in the same way. Our work is closest to Shahbaba and Neal (2009). They proposed a model that mixes over both the covariates and response, where the response is drawn from a multinomial logistic model. The DP-GLM is a generalization of their idea.

Asymptotic properties of Dirichlet process mixture models have been studied mostly in the context of density estimation, specifically consistency of the posterior density for DP Gaussian mixture models (Barron et al., 1999; Ghosal et al., 1999; Ghosh and Ramamoorthi, 2003; Walker, 2004;

Tokdar, 2006) and semi-parametric linear regression models (Amewou-Atisso et al., 2003; Tokdar, 2006). Recently, the posterior properties of DP regression estimators have been studied. Rodriguez et al. (2009) showed point-wise consistency (asymptotic unbiasedness) for the regression estimate produced by their model assuming continuous covariates under different treatments with a continuous responses and a conjugate base measure (normal-inverse Wishart). In Section 5 we show weak consistency of the joint density estimate produced by the DP-GLM. This is used to show pointwise consistency of the regression estimate in both the continuous and categorical response settings. In the continuous response setting, our results generalize those of Rodriguez et al. (2009) and Rodrıguez (2009). In the categorical response setting, our theory provides results for the classification model of Shahbaba and Neal (2009).

3. Mathematical Background

In this section we provide mathematical background. We review Dirichlet process mixture models and generalized linear models.

3.1 Dirichlet Process Mixture Models

The *Dirichlet process* (DP) is a distribution over distributions (Ferguson, 1973). It is denoted,

$$G \sim \mathrm{DP}(\alpha G_0),$$

where G is a random distribution. There are two parameters. The base distribution G_0 is a distribution over the same space as G. For example, if G is a distribution on reals then G_0 must be a distribution on reals too. The concentration parameter α is a positive scalar. One property of the DP is that random distributions G are discrete, and each places its mass on a countably infinite collection of atoms drawn from G_0.

Consider the model

$$\begin{aligned} G &\sim \mathrm{DP}(\alpha G_0), \\ \theta_i &\sim G. \end{aligned}$$

Marginalizing out the random distribution, the joint distribution of n replicates of θ_i is

$$p(\theta_{1:n} \,|\, \alpha G_0) = \int \left(\prod_{i=1}^{n} G(\theta_i) \right) P(G) dG.$$

This joint distribution has a simpler form. The conditional distribution of θ_n given $\theta_{1:(n-1)}$ follows a Polya urn distribution (Blackwell and MacQueen, 1973),

$$\theta_n | \theta_{1:(n-1)} \sim \frac{1}{\alpha + n - 1} \sum_{i=1}^{n-1} \delta_{\theta_i} + \frac{\alpha}{\alpha + n - 1} \mathbb{G}_0. \tag{1}$$

With this conditional distribution, we use the chain rule to specify the joint distribution.

Equation (1) reveals the *clustering property* of the joint distribution of $\theta_{1:n}$: there is a positive probability that each θ_i will take on the value of another θ_j, leading some of the variables to share values. This equation also reveals the roles of scaling parameter α and base distribution $\mathbb{G}_0$. The unique values contained in $\theta_{1:n}$ are drawn independently from $\mathbb{G}_0$, and the parameter α determines how likely θ_{n+1} is to be a newly drawn value from $\mathbb{G}_0$ rather than take on one of the values from $\theta_{1:n}$.

In a DP mixture, θ_i is a latent variable that parameterizes the distribution of an observed data point, point (Antoniak, 1974),

$$\begin{aligned} P &\sim \mathrm{DP}(\alpha\mathbb{G}_0), \\ \Theta_i &\sim P, \\ x_i|\theta_i &\sim f(\cdot\,|\,\theta_i). \end{aligned}$$

Consider the posterior distribution of $\theta_{1:n}$ given $x_{1:n}$. Because of the clustering property, observations group according to their shared parameters. Unlike finite clustering models, however, the number of groups is not assumed known in advance of seeing the data. For this reason, DP mixtures are sometimes called "infinite clustering" models.

3.2 Generalized Linear Models

Generalized linear models (GLMs) build on linear regression to provide a flexible suite of predictive models. GLMs relate a linear model to a response via a link function; examples include familiar models like logistic regression, Poisson regression, and multinomial regression. See McCullagh and Nelder (1989).

GLMs have three components: the conditional probability model of response Y given covariates x, the linear predictor, and the link function. GLMs assume that the response distribution is in the exponential family,

$$f(y|\eta) = \exp\left(\frac{y\eta - b(\eta)}{a(\phi)} + c(y,\phi)\right).$$

Here we give the canonical form of the exponential family, where a, b, and c are known functions specific to the exponential family, ϕ is a scale parameter (sometimes called a dispersion parameter), and η is the canonical parameter. A linear predictor, $X\beta$, is used to determine the canonical parameter through a set of transformations. The mean response is $b'(\eta) = \mu = \mathbb{E}[Y|X]$ (Brown, 1986). However, we can choose a link function g such that $\mu = g^{-1}(X\beta)$, which defines η equal to $X\beta$.

4. Dirichlet Process Mixtures of Generalized Linear Models

We now turn to Dirichlet process mixtures of generalized linear models (DP-GLMs), a Bayesian predictive model that places prior mass on a large class of response densities. Given a data set of covariate-response pairs, we describe Gibbs sampling algorithms for approximate posterior inference and prediction. We derive theoretical properties of the DP-GLM in Section 5.

4.1 Model Formulation

In a DP-GLM, we assume that the covariates X are modeled by a mixture of exponential-family distributions, the response Y is modeled by a GLM conditioned on the covariates, and that these models are connected by associating a set of GLM coefficients with each exponential family mixture component. Let $\theta = (\theta_x, \theta_y)$ be the bundle of parameters over X and $Y\,|\,X$, and let $\mathbb{G}_0$ be a base measure on the space of both. For example, θ_x might be a set of d-dimensional multivariate Gaussian

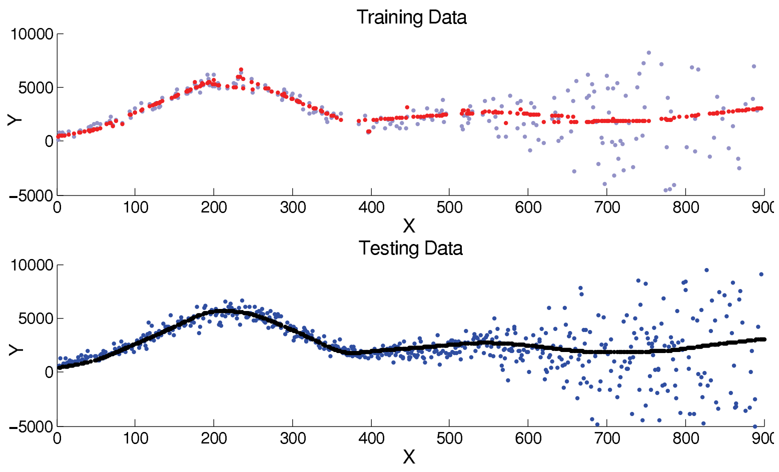


Figure 1: The top figure shows the training data (gray) fitted into clusters, with the prediction given a single sample from the posterior, $\theta^{(i)}$ (red). The bottom figure shows the smoothed regression estimate (black) for the Gaussian model of Equation (2) with the testing data (blue). Data plot multipole moments (X) against power spectrum C_ℓ (Y) for cosmic microwave background radiation (Bennett et al., 2003).

location and scale parameters for a vector of continuous covariates; θ_y might be a $d+2$-vector of reals for their corresponding GLM linear prediction coefficients, along with a GLM dispersion parameter. The full model is

$$\begin{aligned} P &\sim DP(\alpha\mathbb{G}_0), \\ \theta = (\theta_{i,x}, \theta_{i,y})|P &\sim P, \\ X_i|\theta_{i,x} &\sim f_x(\cdot|\theta_{i,x}), \\ Y_i|x_i, \theta_{i,y} &\sim GLM(\cdot|X_i, \theta_{i,y}). \end{aligned}$$

The density f_x describes the covariate distribution; the GLM for y depends on the form of the response (continuous, count, category, or others) and how the response relates to the covariates (i.e., the link function).

The Dirichlet process clusters the covariate-response pairs (x, y). When both are observed, that is, in “training,” the posterior distribution of this model will cluster data points according to near-by covariates that exhibit the same kind of relationship to their response. When the response is not observed, its predictive expectation can be understood by clustering the covariates based on the training data, and then predicting the response according to the GLM associated with the covariates' cluster. The DP prior acts as a kernel for the covariates; instead of being a Euclidean metric, the DP measures the distance between two points by the probability that the hidden parameter is shared. See Figure 1 for a demonstration of the DP-GLM.

We now give a few examples of the DP-GLM that will be used throughout this paper.

4.1.1 EXAMPLE: GAUSSIAN MODEL

We now give an example of the DP-GLM for continuous covariates/response that will be used throughout the rest of the paper. For continuous covariates/response in $\mathbb{R}$, we model locally with a Gaussian distribution for the covariates and a linear regression model for the response. The covariates have mean $\mu_{i,j}$ and variance $\sigma^2_{i,j}$ for the j^{th} dimension of the i^{th} observation; the covariance matrix is diagonal in this example. The GLM parameters are the linear predictor $\beta_{i,0},\dots,\beta_{i,d}$ and the response variance $\sigma^2_{i,y}$. Here, $\theta_{x,i} = (\mu_{i,1:d},\sigma_{i,1:d})$ and $\theta_{y,i} = (\beta_{i,0:d},\sigma_{i,y})$. This produces a mixture of multivariate Gaussians. The full model is,

$$\begin{aligned} P &\sim DP(\alpha\mathbb{G}_0), && (2)\\ \theta_i|P &\sim P, &&\\ X_{i,j}|\theta_{i,x} &\sim N\left(\mu_{ij},\sigma^2_{ij}\right), && j=1,\dots,d,\\ Y_i|X_i,\theta_{i,y} &\sim N\left(\beta_{i0}+\sum_{j=1}^{d}\beta_{ij}X_{ij},\sigma^2_{iy}\right). && \end{aligned}$$

This model has been proposed by West et al. (1994), Escobar and West (1995) and Müller et al. (1996). However, they use a fully populated covariance matrix that gives *de facto* β parameters. This is computationally expensive for larger problems and adds posterior likelihood associated with the covariates, rather than the response. A discussion of the problems associated with the latter issue is given in Section 4.4.

4.1.2 EXAMPLE: MULTINOMIAL MODEL (SHAHBABA AND NEAL, 2009)

This model was proposed by Shahbaba and Neal (2009) for nonlinear classification, using a Gaussian mixture to model continuous covariates and a multinomial logistic model for a categorical response with K categories. The covariates have mean $\mu_{i,j}$ and variance $\sigma^2_{i,j}$ for the j^{th} dimension of the i^{th} observation; the covariance matrix is diagonal for simplicity. The GLM parameters are the K linear predictor $\beta_{i,0,k},\dots,\beta_{i,d,k}$, $k=1,\dots,K$. The full model is,

$$\begin{aligned} P &\sim DP(\alpha\mathbb{G}_0), && (3)\\ \theta_i|P &\sim P, &&\\ X_{i,j}|\theta_{i,x} &\sim N\left(\mu_{ij},\sigma^2_{ij}\right), && j=1,\dots,d,\\ \mathbb{P}(Y_i=k|X_i,\theta_{i,y}) &= \frac{\exp\left(\beta_{i,0,k}+\sum_{j=1}^{d}\beta_{i,j,k}X_{i,j}\right)}{\sum_{\ell=1}^{K}\exp\left(\beta_{i,0,\ell}+\sum_{j=1}^{d}\beta_{i,j,\ell}X_{i,j}\right)}, && k=1,\dots,K. \end{aligned}$$

4.1.3 EXAMPLE: POISSON MODEL WITH CATEGORICAL COVARIATES

We model the categorical covariates by a mixture of multinomial distributions and the count response by a Poisson distribution. If covariate j has K categories, let $(p_{i,j,1},\dots,p_{i,j,K})$ be the probabilities for categories $1,\dots,K$. The covariates are then coded by indicator variables, $\mathbf{1}_{\{X_{i,j}=k\}}$, which are used with the linear predictor, $\beta_i, 0, \beta_{i,1,1:K}, \ldots, \beta_{i,d,1:K}$. The full model is,

$$\begin{aligned}
P &\sim DP(\alpha\mathbb{G}_0), && (4)\\
\theta_i|P &\sim P, &&\\
\mathbb{P}(X_{i,j}=k|\theta_{i,x}) &= p_{i,j,k}, && j=1,\ldots,d,\ k=1,\ldots,K,\\
\lambda_i|X_i,\theta_{i,y} &= \exp\left(\beta_{i,0}+\sum_{j=1}^{d}\sum_{k=1}^{K}\beta_{i,j,k}\mathbf{1}_{\{X_{i,j}=k\}}\right), &&\\
\mathbb{P}(Y_i=k|X_i,\theta_{i,y}) &= \frac{e^{-\lambda_i}\lambda_i^k}{\ell!}, && k=0,1,2,\ldots.
\end{aligned}$$

We apply Model (4) to data in Section 6.

4.2 Heteroscedasticity and Overdispersion

One advantage of the DP-GLM is that it provides a strategy for handling common problems in predictive modeling. Many models, such as GLMs and Gaussian processes, make assumptions about data dispersion and homoscedasticity. Overdispersion occurs in single parameter GLMs when the data variance is larger than the variance predicted by the model mean. Mukhopadhyay and Gelfand (1997) have successfully used DP mixtures over GLM intercept parameters to create classes of models that include overdispersion. The DP-GLM retains this property, but is not limited to linearity in the covariates.

A model is *homoscedastic* when the response variance is across constant all covariates; a model is *heteroscedastic* when the response variance changes with the covariates. Models like GLMs are homoscedastic and give poor fits when that assumption is violated in the data. In contrast, the DP-GLM captures heteroscedasticity when mixtures of GLMs are used. The mixture model setting allows variance to be modeled by a separate parameter in each cluster or by a collection of clusters in a single covariate location. This leads to smoothly transitioning heteroscedastic posterior response distributions.

This property is shown in Figure 2, where we compare a DP-GLM to a homoscedastic model (Gaussian processes) and heteroscedastic modifications of homoscedastic models (treed Gaussian processes and treed linear models). The DP-GLM is robust to heteroscedastic data—it provides a smooth mean function estimate, while the other models are not as robust or provide non-smooth estimates.

4.3 Posterior Prediction With a DP-GLM

The DP-GLM is used in prediction problems. Given a collection of covariate-response pairs $D = (X_i,Y_i)_{i=1}^n$, we estimate the joint distribution of $(X,Y)\,|\,D$. For a new set of covariates x, we use the joint to compute the conditional distribution, $Y\,|\,x,D$ and the conditional expectation, $\mathbb{E}[Y\,|\,x,D]$. We give the step-by-step process for formulating specific DP-GLM models and computing the conditional distribution of the response.

4.3.1 Choosing the Mixture Component and GLM

We begin by choosing f_x and the GLM. The Dirichlet process mixture model and GLM provide flexibility in both the covariates and the response. Dirichlet process mixture models allow many

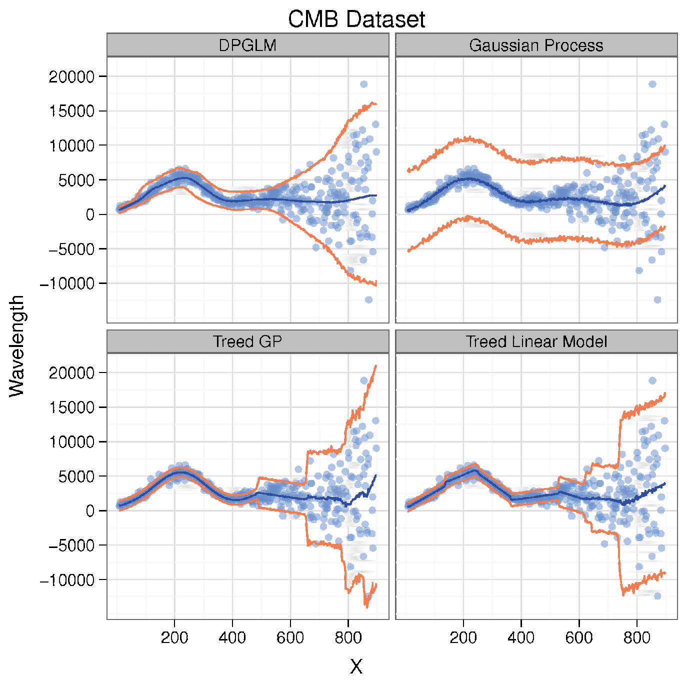


Figure 2: Modeling heteroscedasticity with the DP-GLM and other Bayesian nonparametric methods. The estimated mean function is given along with a 90% predicted confidence interval for the estimated underlying distribution. DP-GLM produces a smooth mean function and confidence interval.

types of variables to be modeled by the covariate mixture and subsequently transformed for use as a covariate in the GLM. Note that certain mixture distributions support certain types of covariates but may not necessarily be a good fit. The same care that goes into choosing distributions and GLMs in a parametric setting is required here.

4.3.2 CHOOSING THE BASE MEASURE AND OTHER HYPERPARAMETERS

The choice of the base measure $\mathbb{G}_0$ affects how expressive the DP-GLM is, the computational efficiency of the prediction and whether some theoretical properties, such as asymptotic unbiasedness, hold. For example, $\mathbb{G}_0$ for the Gaussian model is a distribution over $(\mu_i, \sigma_i, \beta_{i,0:d}, \sigma_{i,y})$. A conjugate base measure is normal-inverse-gamma for each covariate dimension and multivariate normal inverse-gamma for the response parameters. This $\mathbb{G}_0$ allows all continuous, integrable distributions to be supported, retains theoretical properties, such as asymptotic unbiasedness, and yields efficient posterior approximation by collapsed Gibbs sampling (Neal, 2000). In summary, the base measure is chosen in line with data size, distribution type, distribution features (such as heterogeneity, and others) and computational constraints.

Hyperparameters for the DP-GLM include the DP scaling parameter α and hyperparameters parameters for the base measure $\mathbb{G}_0$. We can place a gamma prior on α (Escobar and West, 1995); the parameters of $\mathbb{G}_0$ may also have a prior. Each level of prior reduces the influence of the hyperparameters, but adds computational complexity to posterior inference (Escobar and West, 1995).

4.3.3 Approximating the Posterior and Forming Predictions

We derive all quantities of interest—that is, conditional distributions and expectations—from the posterior of the joint distribution of (x,y). Define $f(x,y\,|\,D)$ as the joint posterior distribution given data D and $f(x,y\,|\,\theta_{1:n})$ as the joint distribution given parameters $\theta_{1:n}$ that are associated with data $D=(X_i,Y_i)_{i=1}^n$. The posterior can be expressed through a conditional expectation,

$$f(x,y\,|\,D)=\mathbb{E}\left[f(x,y\,|\,\theta_{1:n})\,|\,D\right]. \tag{5}$$

While the true posterior distribution, $f(x,y\,|\,D)$, may be impossible to compute, the joint distribution conditioned on $\theta_{1:n}$ has the form

$$f(x,y\,|\,\theta_{1:n})=\frac{\alpha}{\alpha+n}\int_{\mathcal{T}}f_y(y|x,\theta)f_x(x|\theta)\mathbb{G}_0(d\theta)+\frac{1}{\alpha+n}\sum_{i=1}^n f_y(y|x,\theta_i)f_x(x|\theta_i).$$

We approximate the expectation in Equation (5) by Monte Carlo integration using M posterior samples of $\theta_{1:n}$,

$$f(x,y\,|\,D)\approx\frac{1}{M}\sum_{m=1}^M f(x,y\,|\,\theta_{1:n}^{(m)}).$$

We use Markov chain Monte Carlo (MCMC), specifically Gibbs sampling, to obtain M i.i.d. samples from this distribution. (See Escobar, 1994, MacEachern, 1994, Escobar and West, 1995 and MacEachern and Müller, 1998 for foundational work; Neal, 2000 provides a review and state of the art algorithms.) We construct a Markov chain on the hidden variables $\theta_{1:n}$ such that its limiting distribution is the posterior. We give implementation details in Appendix A.

We use a similar strategy to construct the conditional distribution of $Y\,|\,X=x,D$. The conditional distribution is

$$f(Y\,|\,X=x,D)=\frac{f(Y,x\,|\,D)}{\int f(y,x\,|\,D)dy}.$$

Again using M i.i.d. samples from the posterior of $\theta_{1:n}\,|\,D$,

$$\begin{aligned}f(Y\,|\,X=x,D)&\approx\frac{1}{M}\sum_{m=1}^M f(Y\,|\,X=x,\theta_{1:n}^{(m)}),\\&=\frac{1}{M}\sum_{m=1}^M\frac{\alpha\int_{\mathcal{T}}f_y(Y|X=x,\theta)f_x(x|\theta)\mathbb{G}_0(d\theta)+\sum_{i=1}^n f_y(Y|X=x,\theta_i^{(m)})f_x(x|\theta_i^{(m)})}{\alpha\int_{\mathcal{T}}f_x(x|\theta)\mathbb{G}_0(d\theta)+\sum_{i=1}^n f_x(x|\theta_i^{(m)})}.\end{aligned}$$

We use the same methodology to compute the conditional expectation of the response given a new set of covariates x and the observed data D, $\mathbb{E}[Y\,|\,X=x,D]$. Again using iterated expectation, we condition on the latent variables,

$$\mathbb{E}\left[Y\,|\,X=x,D\right]=\mathbb{E}\left[\mathbb{E}\left[Y\,|\,X=x,\theta_{1:n}\right]\,|\,D\right]. \tag{6}$$

Conditional on the latent parameters $\theta_{1:n}$ that generated the observed data, the inner expectation is

$$\mathbb{E}[Y|X=x,\theta_{1:n}]=\frac{\alpha\int_{\mathcal{T}}\mathbb{E}\left[Y|X=x,\theta\right]f_x(x|\theta)\mathbb{G}_0(d\theta)+\sum_{i=1}^n\mathbb{E}\left[Y|X=x,\theta_i\right]f_x(x|\theta_i)}{\alpha\int_{\mathcal{T}}f_x(x|\theta)\mathbb{G}_0(d\theta)+\sum_{i=1}^n f_x(x|\theta_i)}.$$

Since we assume Y is a GLM, $\mathbb{E}\left[Y|X=x,\theta\right]$ is available in closed form as a function of x and θ.

The outer expectation of Equation (6) is usually intractable. We approximate it by Monte Carlo integration with M posterior samples of $\theta_{1:n}$,

$$\mathbb{E}\left[Y \,|\, X = x, D\right] \approx \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}\left[Y \,|\, X = x, \theta_{1:n}^{(m)}\right].$$

4.4 Comparison to the Dirichlet Process Mixture Model Regression

The DP-GLM models the response Y conditioned on the covariates X. An alternative is one where we model (X,Y) from a common mixture component in a classical DP mixture (see Section 3), and then form the conditional distribution of the response from this joint. We investigate the mathematical differences between these approaches and the consequences of those differences. (They are compared empirically in Section 6.)

A Dirichlet process mixture model (DPMM) has the form,

$$\begin{aligned} P &\sim DP(\alpha\mathbb{G}_0), \qquad (7)\\ \theta_i|P &\sim P,\\ X_i|\theta_{i,x} &\sim f_x(x|\theta_{i,x}),\\ Y_i|\theta_{i,y} &\sim f_y(y|\theta_{i,y}). \end{aligned}$$

This model has been studied in Escobar and West (1995) where (X_i, Y_i) are assumed to have a joint Gaussian distribution. When the covariance matrix is assumed to be diagonal, the regression estimate is generally poor. However, when the covariance matrix is assumed to be fully populated, computation becomes difficult with more than a few covariate dimensions. We focus on the case with diagonal covariance. We study why it performs poorly and how the DP-GLM improves on it with minimal increase in computational difficulty. The difference between Model (7) and the DP-GLM is that the distribution of Y given θ is conditionally independent of the covariates X. This difference has consequences on the posterior distribution and, thus, the posterior predictions.

One consequence is that the GLM response component acts to remove boundary bias for samples near the boundary of the covariates in the training data set. The GLM fits a linear predictor through the training data; all predictions for boundary and out-of-sample covariates follow the local predictors. The traditional DP model, however, only fits a local mean; all boundary and out-of-sample predictions center around that mean. The boundary effects are compared in Figure 3. The DP-GLM can be viewed as a Bayesian analogy of a locally linear kernel estimator while the regular DP is similar to the Nadaraya-Watson kernel estimator (Nadaraya, 1964; Watson, 1964).

Another consequence is that the proportion of the posterior likelihood devoted to the response differs between the two methods. Consider the log of the posterior of the DPMM given in Model (7). Assume that f_y is a single parameter exponential, where $\theta_y = \beta$,

$$\ell(\theta^{dp} \,|\, D) \propto \sum_{i=1}^{K} \left[\ell(\beta_{C_i}) + \sum_{c \in C_i} \ell(y_c \,|\, \beta_{C_i}) + \sum_{j=1}^{d} \ell(\theta_{C_i,x_j} \,|\, D)\right]. \qquad (8)$$

Here, ℓ denotes log likelihood and "$\propto$" means "proportional in the log space." The log of the DP-GLM posterior for a single parameter exponential family GLM, where $\theta_y = (\beta_0, \ldots, \beta_d)$, has the form,

$$\ell(\theta^{dpglm} \,|\, D) \propto \sum_{i=1}^{K} \left[\sum_{j=0}^{d} \ell(\beta_{C_i,j}) + \sum_{c \in C_i} \ell(y_c \,|\, \beta_{C_i}^T x_c) + \sum_{j=1}^{d} \ell(\theta_{C_i,x_j} \,|\, D)\right]. \qquad (9)$$

As the number of covariates grows, the likelihood associated with the covariates grows in both equations. However, the likelihood associated with the response also grows with the extra response parameters in Equation (9), whereas it is fixed in Equation (8).

These posterior differences lead to two predictive differences. First, the DP-GLM is much more resistant to dimensionality than the DPMM. Since the number of response related parameters grows with the number of covariate dimensions in the DP-GLM, the relative posterior weight of the response does not shrink as quickly in the DP-GLM as it does in the DPMM. This keeps the response variable important in the selection of the mixture components and makes the DP-GLM a better predictor than the DPMM as the number of dimensions grows.

As the dimensionality grows, however, the DP-GLM produces less stable predictions than the DPMM. While the additional GLM parameters help maintain the relevance of the response, they also add noise to the prediction. This is seen in Figure 3. The GLM parameters in this figure have a Gaussian base measure, effectively creating a local ridge regression.[1] In lower dimensions, the DP-GLM produced more stable results than the DPMM because a smaller number of larger clusters were required to fit the data well. The DPMM, however, consistently produced stable results in higher dimensions as the response became more of a sample average than a local average. The DPMM has the potential to predict well if changes in the mean function coincide with underlying local modes of the covariate density. However, the DP-GLM forces the covariates into clusters that coincide more with the response variable due to the inclusion of the slope parameters.

We now discuss the theoretical properties of the DP-GLM.

5. Asymptotic Properties of the DP-GLM Model

In this section, we study the asymptotic properties of the DP-GLM model, namely weak consistency of the joint density estimate and pointwise consistency (asymptotic unbiasedness) of the regression estimate. Consistency is the notion that posterior distribution accumulates in regions close to the true distribution. Weak consistency assures that the posterior distribution accumulates in regions of densities where "properly behaved" functions (i.e., bounded and continuous) integrated with respect to the densities in the region are arbitrarily close to the integral with respect to the true density. We then use the weak consistency results to give conditions for asymptotic unbiasedness of the regression estimate. Both consistency and asymptotic unbiasedness act as frequentist justification of Bayesian methods; more observations lead to models that tend toward the "correct" value. Neither weak consistency nor asymptotic unbiasedness are guaranteed for Dirichlet process mixture models.

Notation for this section is more complicated than the notation for the model. Let $f_0(x,y)$ be the true joint distribution of (x,y); in this case, we will assume that f_0 is a density. Let $\mathcal{F}$ be the set of all density functions over (x,y). Let Π^f be the prior over $\mathcal{F}$ induced by the DP-GLM model. Let $\mathbb{E}_{f_0}[\cdot]$ denote the expectation under the true distribution and $\mathbb{E}_{\Pi^f}[\cdot]$ be the expectation under the prior Π^f.

In general, an estimator is a function of observations. Assuming a true distribution of those observations, an estimator is called unbiased if its expectation under that distribution is equal to the value that it estimates. If an estimator has this property, it is called consistent. In the case of

1. In unpublished results, we tried other base measures, such as a Laplacian distribution. They produced less stable results than the Gaussian base measure.

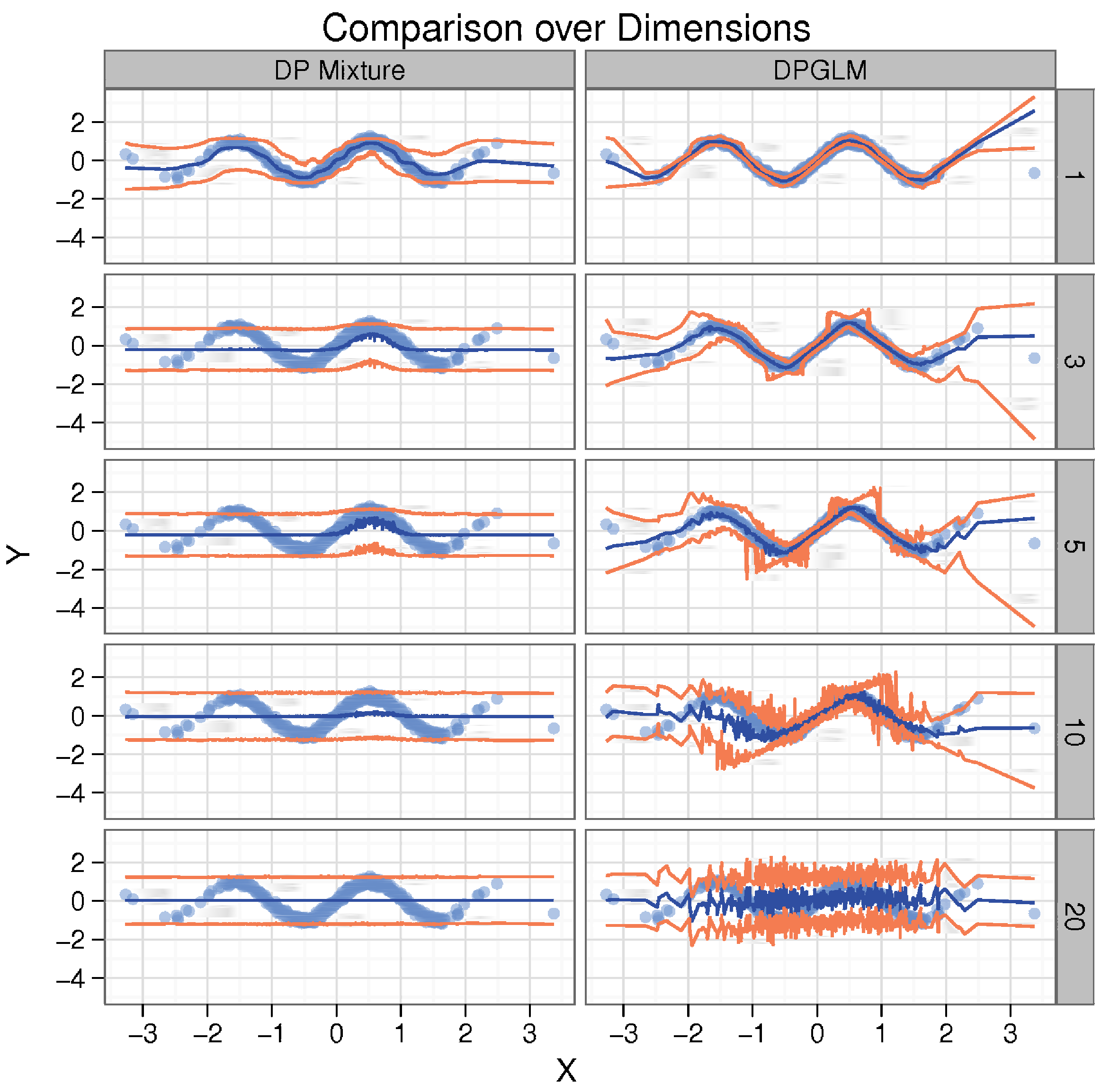


Figure 3: A plain Dirichlet process mixture model regression (left) versus DP-GLM, plotted against the number of spurious dimensions (vertical plots). We give the estimated mean function along with a 90% predicted confidence interval for the estimated underlying distribution. Data have one predictive covariate and a varying number of spurious covariates. The covariate data were generated by a mixture model. DP-GLM produces a smoother mean function and is much more resistant to spurious dimensionality.

DP-GLM, that would mean for every x in a fixed domain $\mathcal{A}$ and every $n > 0$,

$$\mathbb{E}_{f_0}\left[\mathbb{E}_{\Pi^f}[Y|x,(X_i,Y_i)_{i=1}^n]\right] = \mathbb{E}_{f_0}[Y|x].$$

Since we use Bayesian priors in DP-GLM, we will have bias in almost all cases. The best we can hope for is a consistent estimator, where as the number of observations grows to infinity, the mean function estimate converges to the true mean function. That is, for every $x \in \mathcal{A}$,

$$\mathbb{E}_{\Pi^f}[Y|x,(X_i,Y_i)_{i=1}^n] \to \mathbb{E}_{f_0}[Y|x] \quad \text{as } n \to \infty.$$

5.1 Weak Consistency of the Joint Posterior Distribution

Weak consistency is the idea that the posterior distribution, $\Pi^f(f\,|\,(X_i,Y_i)_{i=1}^n)$ collects in weak neighborhoods of the true distribution, $f_0(x,y)$. A weak neighborhood of f_0 of radius ε, $\mathcal{W}_\varepsilon(f_0)$, is defined as follows,

$$\mathcal{W}_\varepsilon(f_0) = \left\{ f : \left| \int f_0(x,y)g(x,y)dxdy - \int f(x,y)g(x,y)dxdy \right| < \varepsilon \right\}$$

for every bounded, continuous function g. Aside from guaranteeing that the posterior collects in regions close to the true distribution, weak consistency can be used to show consistency of the regression estimate under certain conditions. We give conditions for weak consistency for joint posterior distribution of the Gaussian and multinomial models and use these results to show consistency of the regression estimate for these same models.

We now give a theorem for the asymptotic unbiasedness of the Gaussian model.

Theorem 1 *Let Π^f be the prior induced by the Gaussian model of Equation (2). If $f_0(x,y)$ has compact support, is absolutely continuous over that domain and $\mathbb{G}_0$ has support $\mathbb{R}^d \times \mathbb{R}_+^d \times \mathbb{R}^{d+1} \times \mathbb{R}_+$, then*

$$\Pi^f(\mathcal{W}_\varepsilon(f_0)\,|\,(X_i,Y_i)_{i=1}^n) \to 1$$

as $n \to \infty$ for every $\varepsilon > 0$.

Posterior consistency of similar models, namely Dirichlet process mixtures of Gaussians, has been extensively studied by Ghosal et al. (1999), Ghosh and Ramamoorthi (2003), and Tokdar (2006) and convergence rates in Walker et al. (2007). The compact support condition for f_0 allows for broad array of base measures to produce weakly consistent posteriors. See Tokdar (2006) for results on non-compactly supported f_0.

We now give an analogous theorem for the multinomial model.

Theorem 2 *Let Π^f be the prior induced by the multinomial model of Equation (3). If $f_0(x)$ has compact support, is absolutely continuous, $\mathbb{G}_0$ has support $\mathbb{R}^d \times \mathbb{R}_+^d \times \mathbb{R}^{d+1}$, and $\mathbb{P}_{f_0}[Y=k\,|\,X=x]$ is absolutely continuous in x for $k=1,\dots,K$, then*

$$\Pi^f(\mathcal{W}_\varepsilon(f_0)\,|\,(X_i,Y_i)_{i=1}^n) \to 1$$

as $n \to \infty$ for every $\varepsilon > 0$.

The proofs of Theorems 1 and 2 are given in the Appendix.

5.2 Consistency of the Regression Estimate

We approach consistency of the regression estimate by using weak consistency for the posterior of the *joint* distribution and then placing additional integrability constraints on the base measure $\mathbb{G}_0$. We now give results for the Gaussian and multinomial models.

Theorem 3 *Let Π^f be the prior induced by the Gaussian model of Equation (2). If*

(i) $\mathbb{G}_0$ *and* f_0 *satisfy the conditions of Theorem 1, and*

(ii) $\int(\beta_0 + \sum_{i=1}^d \beta_i x_i)\mathbb{G}_0(d\beta) < \infty$ *for every* $x \in \mathcal{C}$,

then

$$\lim_{n\to\infty} \mathbb{E}_{f_0}\left[\mathbb{E}_{\Pi^f}[Y|x,(X_i,Y_i)_{i=1}^n]\right] = \mathbb{E}_{f_0}[Y|x]$$

almost surely $\mathbb{P}_{f_0}^\infty$.

Similarly, we give a theorem for the multinomial model.

Theorem 4 *Let Π^f be the prior induced by the multinomial model of Equation (3). If*

(i) $\mathbb{G}_0$ *and* f_0 *satisfy the conditions of Theorem 2, and*

(ii) $\mathbb{P}_{f_0}[Y = k \,|\, X = x]$ *is continuous in* x *for* $k = 1,\dots,K$,

then

$$\lim_{n\to\infty} \mathbb{E}_{f_0}\left[\mathbb{P}_{\Pi^f}[Y = k|x,(X_i,Y_i)_{i=1}^n]\right] = \mathbb{P}_{f_0}[Y = k|x]$$

almost surely $\mathbb{P}_{f_0}^\infty$ *for* $k = 1,\dots,K$.

See Appendix B for proofs of Theorems 3 and 4.

5.3 Consistency Example: Gaussian Model

Examples of prior distributions that satisfy Theorems 1 and 3 are as follows.

5.3.1 NORMAL-INVERSE-WISHART

Note that in the Gaussian case, slope parameters can be generated by a full covariance matrix: using a conjugate prior, a Normal-Inverse-Wishart, will produce an instance of the DP-GLM. Define the following model, which was used by Müller et al. (1996),

$$\begin{aligned} P &\sim DP(\alpha\mathbb{G}_0), \\ \theta_i \,|\, P &\sim P, \\ (X_i,Y_i)\,|\,\theta_i &\sim N(\mu,\Sigma). \end{aligned} \tag{10}$$

The last line of Model (10) can be broken down in the following manner,

$$\begin{aligned} X_i\,|\,\theta_i &\sim N(\mu_x,\Sigma_x), \\ Y_i\,|\,\theta_i &\sim N\left(\mu_y + b^T\Sigma_x^{-1}b(X_i-\mu_x), \sigma_y^2 - b^T\Sigma_x^{-1}b\right), \end{aligned}$$

where

$$\mu = \begin{bmatrix} \mu_y \\ \mu_x \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} \sigma_y^2 & b^T \\ b & \Sigma_x \end{bmatrix}.$$

We can then define β as,

$$\beta_0 = \mu_y - b^T \Sigma_x^{-1} \mu_x, \qquad \beta_{1:d} = b^T \Sigma_x^{-1}.$$

The base measure $\mathbb{G}_0$ is defined as,

$$(\mu, \Sigma) \sim Normal\ Inverse\ Wishart(\lambda, \nu, a, B).$$

Here λ is a mean vector, ν is a scaling parameter for the mean, a is a scaling parameter for the covariance, and B is a covariance matrix.

5.3.2 Diagonal Normal-Inverse-Gamma

It is often more computationally efficient to specify that Σ_x is a diagonal matrix. In this case, we can specify a conjugate base measure component by component:

$$\begin{aligned}
\sigma_{i,j} &\sim Inverse\ Gamma(a_j, b_j), && j = 1, \dots, d, \\
\mu_{i,j} \,|\, \sigma_{i,j} &\sim N(\lambda_j, \sigma_{i,j}/\nu_j), && j = 1, \dots, d, \\
\sigma_{i,y} &\sim Inverse\ Gamma(a_y, b_y), \\
\beta_{i,j} \,|\, \sigma_{i,y} &\sim N_{d+1}(\lambda_y, \sigma_y/\nu_y).
\end{aligned}$$

The Gibbs sampler can still be collapsed, but the computational cost is much lower than the full Normal-Inverse-Wishart.

5.3.3 Normal Mean, Log Normal Variance

Conjugate base measures tie the mean to the variance and can be a poor fit for small, heteroscedastic data sets. The following base measure was proposed by Shahbaba and Neal (2009),

$$\begin{aligned}
\log(\sigma_{i,j}) &\sim N(m_{j,\sigma}, s_{j,\sigma}^2), && j = y, 1, \dots, d, \\
\mu_{i,j} &\sim N(m_{j,\mu}, s_{j,\mu}^2), && j = 1, \dots, d, \\
\beta_{i,j} &\sim N(m_{j,\beta}, s_{j,\beta}^2) && j = 0, \dots, d.
\end{aligned}$$

5.4 Consistency Example: Multinomial Model

Now consider the multinomial model of Shahbaba and Neal (2009), given in Model (3),

$$\begin{aligned}
P &\sim DP(\alpha \mathbb{G}_0), \\
\theta_i | P &\sim P, \\
X_{i,j} | \theta_{i,x} &\sim N\left(\mu_{ij}, \sigma_{ij}^2\right), && j = 1, \dots, d, \\
\mathbb{P}(Y_i = k | X_i, \theta_{i,y}) &= \frac{\exp\left(\beta_{i,0,k} + \sum_{j=1}^d \beta_{i,j,k} X_{i,j}\right)}{\sum_{\ell=1}^K \exp\left(\beta_{i,0,\ell} + \sum_{j=1}^d \beta_{i,j,\ell} X_{i,j}\right)}, && k = 1, \dots, K.
\end{aligned}$$

Examples of prior distributions that satisfy Theorems 2 and 4 are as follows.

5.4.1 NORMAL-INVERSE-WISHART

The covariates have a Normal-Inverse-Wishart base measure while the GLM parameters have a Gaussian base measure,

$$\begin{aligned} (\mu_{i,x}, \Sigma_{i,x}) &\sim \mathit{Normal\ Inverse\ Wishart}(\lambda, \nu, a, B), \\ \beta_{i,j,k} &\sim N(m_{j,k}, s_{j,k}^2), \qquad j = 0,\dots,d,\ \ k = 1,\dots,K. \end{aligned}$$

5.4.2 DIAGONAL NORMAL-INVERSE-GAMMA

It is often more computationally efficient to specify that Σ_x is a diagonal matrix. Again, we can specify a conjugate base measure component by component while keeping the Gaussian base measure on the GLM components,

$$\begin{aligned} \sigma_{i,j} &\sim \mathit{Inverse\ Gamma}(a_j, b_j), && j = 1,\dots,d, \\ \mu_{i,j} \,|\, \sigma_{i,j} &\sim N(\lambda_j, \sigma_{i,j}/\nu_j), && j = 1,\dots,d, \\ \beta_{i,j,k} \,|\, \sigma_{i,y} &\sim N(m_{j,k}, s_{j,k}^2), && j = 0,\dots,d,\ \ k = 1,\dots,K. \end{aligned}$$

5.4.3 NORMAL MEAN, LOG NORMAL VARIANCE

Likewise, for heteroscedastic covariates we can use the log normal base measure of Shahbaba and Neal (2009),

$$\begin{aligned} \log(\sigma_{i,j}) &\sim N(m_{j,\sigma}, s_{j,\sigma}^2), && j = 1,\dots,d, \\ \mu_{i,j} &\sim N(m_{j,\mu}, s_{j,\mu}^2), && j = 1,\dots,d, \\ \beta_{i,j,k} &\sim N(m_{j,k,\beta}, s_{j,k,\beta}^2) && j = 0,\dots,d,\ \ k = 1,\dots,K. \end{aligned}$$

6. Empirical Study

We compare the performance of DP-GLM regression to other regression methods. We studied data sets that illustrate the strengths of the DP-GLM, including robustness with respect to data type, heteroscedasticity and higher dimensionality than can be approached with traditional methods. Shahbaba and Neal (2009) used a similar model on data with categorical covariates and count responses; their numerical results were encouraging. We tested the DP-GLM on the following data sets.

6.1 Data Sets

We selected three data sets with continuous response variables. They highlight various data difficulties within regression, such as error heteroscedasticity, moderate dimensionality (10–12 covariates), various input types and response types.

- **Cosmic Microwave Background (CMB) (Bennett et al., 2003).** The data set consists of 899 observations which map positive integers $\ell = 1, 2, \dots, 899$, called 'multipole moments,' to the power spectrum C_ℓ. Both the covariate and response are considered continuous. The data pose challenges because they are highly nonlinear and heteroscedastic. Since this data set is only two dimensions, it allows us to easily demonstrate how the various methods approach estimating a mean function while dealing with non-linearity and heteroscedasticity.

- **Concrete Compressive Strength (CCS) (Yeh, 1998).** The data set has eight covariates: the components cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate and fine aggregate, all measured in kg per m^3, and the age of the mixture in days; all are continuous. The response is the compressive strength of the resulting concrete, also continuous. There are 1,030 observations. The data have relatively little noise. Difficulties arise from the moderate dimensionality of the data.

- **Solar Flare (Solar) (Bradshaw, 1989).** The response is the number of solar flares in a 24 hour period in a given area; there are 11 categorical covariates. 7 covariates are binary and 4 have 3 to 6 classes for a total of 22 categories. The response is the sum of all types of solar flares for the area. There are 1,389 observations. Difficulties are created by the moderately high dimensionality, categorical covariates and count response. Few regression methods can appropriately model this data.

Data set testing sizes ranged from very small (20 observations) to moderate sized (800 observations). Small data set sizes were included due to interests in (future) online applications.

6.2 Competitors

The competitors represent a variety of regression methods; some methods are only suitable for certain types of regression problems.

- **Ordinary Least Squares (OLS).** A parametric method that often provides a reasonable fit when there are few observations. Although OLS can be extended for use with any set of basis functions, finding basis functions that span the true function is a difficult task. We naively choose $[1\,X_1 \dots X_d]^T$ as basis functions. OLS can be modified to accommodate both continuous and categorical inputs, but it requires a continuous response function.

- **CART.** A nonparametric tree regression method (Brieman et al., 1984) generated by the Matlab function classregtree. It accommodates both continuous and categorical inputs and any type of response.

- **Bayesian CART.** A tree regression model with a prior over tree size (Chipman et al., 1998); it was implemented in R with the tgp package.

- **Bayesian Treed Linear Model.** A tree regression model with a prior over tree size and a linear model in each of the leaves (Chipman et al., 2002); it was implemented in R with the tgp package.

- **Gaussian Processes (GP).** A nonparametric method that can accommodate only continuous inputs and continuous responses. GPs were generated in Matlab by the program gpml of Rasmussen and Williams (2006).

- **Treed Gaussian Processes.** A tree regression model with a prior over tree size and a GP on each leaf node (Gramacy and Lee, 2008); it was implemented in R with the tgp package.

Method	Mean Absolute Error					Mean Square Error				
Training set size	30	50	100	250	500	30	50	100	250	500
DP-GLM	0.58	0.51	0.49	**0.48**	**0.45**	**1.00**	**0.94**	**0.91**	**0.94**	**0.83**
Linear Regression	0.66	0.65	0.63	0.65	0.63	1.08	1.04	1.01	1.04	0.96
CART	0.62	0.60	0.60	0.56	0.56	1.45	1.34	1.43	1.29	1.41
Bayesian CART	0.66	0.64	0.54	0.50	0.47	1.04	1.01	0.93	**0.94**	0.84
Treed Linear Model	0.64	0.52	0.49	**0.48**	0.46	1.10	0.95	0.93	0.95	0.85
Gaussian Process	0.55	0.53	0.50	0.51	0.47	1.06	0.97	0.93	0.96	0.85
Treed GP	**0.52**	**0.49**	**0.48**	**0.48**	0.46	1.03	0.95	0.95	0.96	0.89

Table 1: Mean absolute and square errors for methods on the CMB data set by training data size. The best results for each size of training data are in bold.

- **Basic DP Regression.** Similar to DP-GLM, except the response is a function only of μ_y, rather than $\beta_0 + \sum \beta_i x_i$. For the Gaussian model,

$$\begin{aligned} P &\sim DP(\alpha \mathbb{G}_0), \\ \theta_i | P &\sim P, \\ X_i | \theta_i &\sim N(\mu_{i,x}, \sigma_{i,x}^2), \\ Y_i | \theta_i &\sim N(\mu_{i,y}, \sigma_{i,y}^2). \end{aligned}$$

  This model was explored in Section 4.4.

- **Poisson GLM (GLM).** A Poisson generalized linear model, used on the Solar Flare data set. It is suitable for count responses.

6.3 Cosmic Microwave Background (CMB) Results

For this data set, we used a Gaussian model with base measure

$$\begin{aligned} \mu_x &\sim N(m_x, s_x^2), & \sigma_x^2 &\sim \exp\left\{N(m_{x,s}, s_{x,s}^2)\right\}, \\ \beta_{0:d} &\sim N(m_{y,0:d}, s_{y,0:d}^2), & \sigma_y^2 &\sim \exp\left\{N(m_{x,s}, s_{x,s}^2)\right\}. \end{aligned}$$

This prior was chosen because the variance tails are heavier than an inverse gamma and the mean is not tied to the variance. It is a good choice for heterogeneous data because of those features. Computational details are given in Appendix C.

All non-linear methods except for CART (DP-GLM, Bayesian CART, treed linear models, GPs and treed GPs) did comparably on this data set; CART had difficulty finding an appropriate bandwidth. Linear regression did poorly due to the non-linearity of the data set. Fits for heteroscedasticity for the DP-GLM, GPs, treed GPs and treed linear models on 250 training data points can be seen in Figure 2. See Figure 4 and Table 1 for results.

6.4 Concrete Compressive Strength (CCS) Results

The CCS data set was chosen because of its moderately high dimensionality and continuous covariates and response. For this data set, we used a Gaussian model and a conjugate base measure with

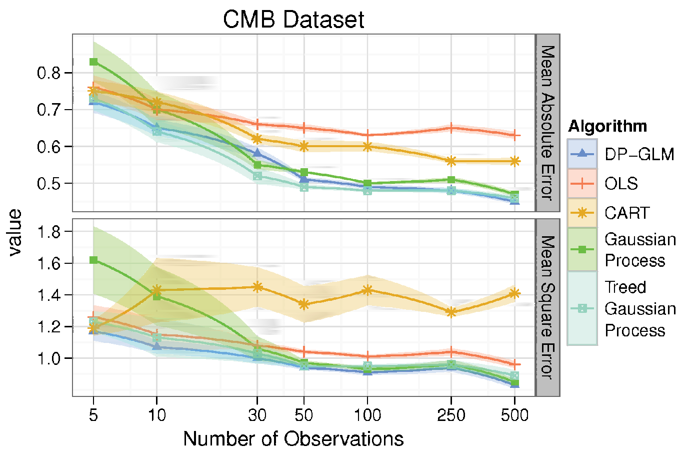


Figure 4: The average mean absolute error (top) and mean squared error (bottom) for ordinary least squares (OLS), tree regression, Gaussian processes and DP-GLM on the CMB data set. The data were normalized. Mean $+/-$ one standard deviation are given for each method.

conditionally independent covariate and response parameters,

$$\begin{aligned}(\mu_x, \sigma_x^2) &\sim Normal-Inverse-Gamma(m_x, s_x, a_x, b_x),\\ (\beta_{0:d}, \sigma_y^2) &\sim Multivariate\ Normal-Inverse-Gamma(M_y, S_y, a_y, b_y).\end{aligned}$$

This base measure allows the sampler to be fully collapsed but has fewer covariate-associated parameters than a full Normal-Inverse-Wishart base measure, giving it a better fit in a moderate dimensional setting. In testing, it also provided better results for this data set than the exponentiated Normal base measure used for the CMB data set; this is likely due to the low noise and variance of the CCS data set. Computational details are given in Appendix C.

Results on this data set were more varied than those for the CMB data set. GPs had the best performance overall; on smaller sets of training data, the DP-GLM outperformed frequentist CART. Linear regression, basic DP regression and Bayesian CART all performed comparatively poorly. Treed linear models and treed GPs performed very well most of the time, but had convergence problems leading to overall higher levels of predictive error. Convergence issues were likely caused by the moderate dimensionality (8 covariates) of the data set. See Figure 5 and Table 2 for results.

6.5 Solar Flare Results

The Solar data set was chosen to demonstrate the flexibility of DP-GLM. Many regression techniques cannot accommodate categorical covariates and most cannot accommodate a count-type re-

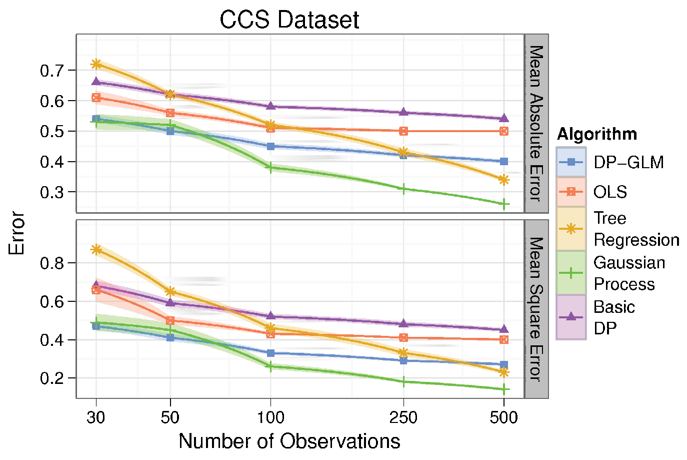


Figure 5: The average mean absolute error (top) and mean squared error (bottom) for ordinary least squares (OLS), tree regression, Gaussian processes, location/scale DP and the DP-GLM Poisson model on the CCS data set. The data were normalized. Mean $+/-$ one standard deviation are given for each method.

Method	Mean Absolute Error					Mean Squared Error				
	30	50	100	250	500	30	50	100	250	500
DP-GLM	0.54	0.50	0.45	0.42	0.40	**0.47**	0.41	0.33	0.28	0.27
Location/Scale DP	0.66	0.62	0.58	0.56	0.54	0.68	0.59	0.52	0.48	0.45
Linear Regression	0.61	0.56	0.51	0.50	0.50	0.66	0.50	0.43	0.41	0.40
CART	0.72	0.62	0.52	0.43	0.34	0.87	0.65	0.46	0.33	0.23
Bayesian CART	0.78	0.72	0.63	0.55	0.54	0.95	0.80	0.61	0.49	0.46
Treed Linear Model	1.08	0.95	0.60	0.35	1.10	7.85	9.56	4.28	0.26	1232
Gaussian Process	**0.53**	0.52	**0.38**	0.31	0.26	0.49	0.45	**0.26**	**0.18**	0.14
Treed GP	0.73	**0.40**	0.47	**0.28**	**0.22**	1.40	**0.30**	3.40	0.20	**0.11**

Table 2: Mean absolute and square errors for methods on the CCS data set by training data size. The best results for each size of training data are in bold.

sponse. For this data set, we used the following DP-GLM,

$$\begin{aligned} P &\sim DP(\alpha\mathbb{G}_0), \\ \theta_i \,|\, P &\sim P, \\ X_{i,j} \,|\, \theta_i &\sim (p_{i,j,1}, \dots, p_{i,j,K(j)}), \\ Y_i \,|\, \theta_i &\sim Poisson\left(\beta_{i,0} + \sum_{j=1}^{d} \sum_{k=1}^{K(j)} \beta_{i,j,k} \mathbf{1}_{\{X_{i,j}=k\}}\right). \end{aligned}$$

We used a conjugate covariate base measure and a Gaussian base measure for β,

$$(p_{j,1}, \dots, p_{j,K(j)}) \sim Dirichlet(a_{j,1}, \dots, a_{j,K(j)}), \qquad \beta_{j,k} \sim N(m_{j,k}, s_{j,k}^2).$$

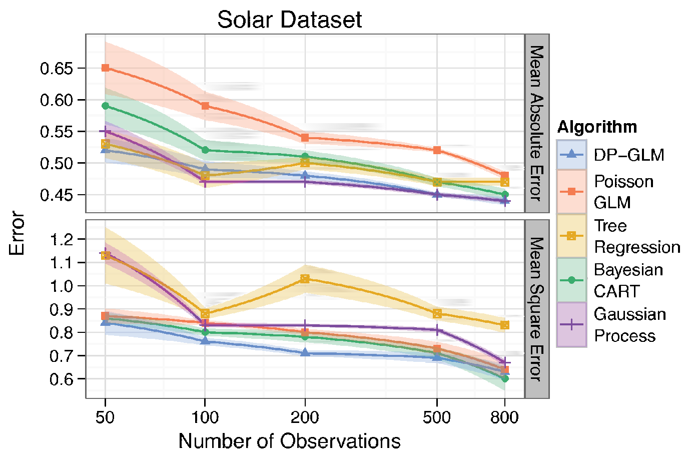


Figure 6: The average mean absolute error (top) and mean squared error (bottom) for tree regression, a Poisson GLM (GLM) and DP-GLM on the Solar data set. Mean $+/-$ one standard deviation are given for each method.

Method	Mean Absolute Error					Mean Squared Error				
	50	100	200	500	800	50	100	200	500	800
DP-GLM	**0.52**	0.49	0.48	**0.45**	**0.44**	**0.84**	**0.76**	**0.71**	**0.69**	0.63
Poisson Regression	0.65	0.59	0.54	0.52	0.48	0.87	0.84	0.80	0.73	0.64
CART	0.53	0.48	0.50	0.47	0.47	1.13	0.88	1.03	0.88	0.83
Bayesian CART	0.59	0.52	0.51	0.47	0.45	0.86	0.80	0.78	0.71	**0.60**
Gaussian Process	0.55	**0.47**	**0.47**	**0.45**	**0.44**	1.14	0.83	0.83	0.81	0.67

Table 3: Mean absolute and square errors for methods on the Solar data set by training data size. The best results for each size of training data are in bold.

Computational details are given in Appendix C.

The only other methods that can handle this data set are CART, Bayesian CART and Poisson regression. GP regression was run with a squared exponential covariance function and Gaussian errors to make use of the ordering in the covariates. The DP-GLM had good performance under both error measures. The high mean squared error values suggests that frequentist CART overfit while the high mean absolute error for Poisson regression suggests that it did not adequately fit nonlinearities. See Figure 6 and Table 3 for results.

6.6 Discussion

The DP-GLM is a relatively strong competitor on all of the data sets. It was more stable than most of its Bayesian competitors (aside from GPs) on the CCS data set. Our results suggest that the DP-GLM would be a good choice for small sample sizes when there is significant prior knowledge; in those cases, it acts as an automatic outlier detector and produces a result that is similar to a Bayesian

GLM. Results from Section 4 suggest that the DP-GLM is not appropriate for problems with high dimensional covariates; in those cases, the covariate posterior swamps the response posterior with poor numerical results.

7. Conclusions and Future Work

We developed the Dirichlet process mixture of generalized linear models (DP-GLM), a flexible Bayesian regression technique. We discussed its statistical and empirical properties; we gave conditions for asymptotic unbiasedness and gave situations in which they hold; finally, we tested the DP-GLM on a variety of data sets against state of the art Bayesian competitors. The DP-GLM was competitive in most setting and provided stable, conservative estimates, even with extremely small sample sizes.

One concern with the DP-GLM is computational efficiency as implemented. All results were generated using MCMC, which does not scale well to large data sets. An alternative implementation using variational inference (Blei and Jordan, 2006), possibly online variational inference (Sato, 2001), would greatly increase computational feasibility for large data sets.

Our empirical analysis of the DP-GLM has implications for regression methods that rely on modeling a joint posterior distribution of the covariates and the response. Our experiments suggest that the covariate posterior can swamp the response posterior, but careful modeling can mitigate the effects for problems with low to moderate dimensionality. A better understanding would allow us to know when and how such modeling problems can be avoided.

Acknowledgments

David M. Blei was supported by ONR 175-6343, NSF CAREER 0745520, AFOSR 09NL202, the Alfred P. Sloan foundation, and a grant from Google. Warren Powell and Lauren Hannah were supported in part by grant AFOSR contract FA9550-08-1-0195 and the National Science Foundation grant CMMI-0856153.

Appendix A.

In the Gibbs sampler, the state is the collection of labels $(z_1,\dots,z_n)$ and parameters $(\theta_1^*,\dots,\theta_K^*)$, where θ_c^* is the parameter associated with cluster c and K is the number of unique labels given $z_{1:n}$. In a collapsed Gibbs sampler, all or part of $(\theta_1^*,\dots,\theta_K^*)$ is eliminated through integration. Let $z_{-i}=(z_1,\dots,z_{i-1},z_{i+1},\dots,z_n)$. A basic inference algorithm is given in Algorithm 1. Convergence criteria for the Gibbs samplers in our numerical examples are given in Appendix C. See Gelman et al. (2004) for a more complete discussion on convergence criteria.

We can sample from the distribution $p(z_i\,|\,D,z_{-i},\theta_{1:K}^*)$ as follows,

$$p(z_i\,|\,D,z_{-i},\theta_{1:K}^*) \propto p(z_i\,|\,z_{-i})p(X_i\,|\,z_{1:n},D,\theta_{1:K}^*)p(Y_i\,|\,X_i,z_{1:n},D,\theta_{1:K}^*). \tag{11}$$

The first part of Equation (11) is the Chinese Restaurant Process posterior value,

$$p(z_i\,|\,z_{-i}) = \begin{cases} \frac{n_{z_j}}{n-1+\alpha} & \text{if } z_i = z_j \text{ for some } j\neq i, \\ \frac{\alpha}{n-1+\alpha} & \text{if } z_i \neq z_j \text{ for all } j\neq i. \end{cases}$$

Algorithm 1: Gibbs Sampling Algorithm for the DP-GLM

```
Require: Starting state $(z_1,\dots,z_n)$, $(\theta^*_1,\dots,\theta^*_K)$, convergence criteria.
 1: repeat
 2:   for $i = 1$ to $n$ do
 3:     Sample $z_i$ from $p(z_i \,|\, D, z_{-i}, \theta^*_{1:K})$.
 4:   end for
 5:   for $c = 1$ to $K$ do
 6:     Sample $\theta^*_c$ given $\{(X_i,Y_i) : z_i = c\}$.
 7:   end for
 8:   if Convergence criteria are met then
 9:     Record $(z_1,\dots,z_n)$ and $(\theta^*_1,\dots,\theta^*_K)$.
10:   end if
11: until $M$ posterior samples obtained.
```

Here n_{z_j} is the number of elements with the label z_j. The second term of Equation (11) is the same as in other Gibbs sampling algorithms. If possible, the component parameters $\theta^*_{1:K}$ can be integrated out (in the case of conjugate base measures and parameters that pertain strictly to the covariates) and $p(X_i \,|\, z_{1:n}, D, \theta^*_{1:K})$ can be replaced with

$$\int p(X_i \,|\, z_{1:n}, D, \theta^*_{1:K}) p(\theta^*_{1:K} \,|\, z_{1:n}) d\theta^*_{1:K}.$$

The third term of Equation (11) is not found in traditional Dirichlet process mixture model samplers. In some cases, this term can also be collapsed, such as Gaussian model with a Normal-Inverse-Gamma base measure. In that case,

$$\begin{aligned}
p(Y_i \,|\, X_i, z_c, D_c) &= \frac{\Gamma((n_n+1)/2)}{\Gamma(n_n/2)} (n_n s_n)^{-1/2} \exp\left(-1/2(n_n+1)\log\left(1+\frac{1}{n_n s_n}(Y_i - m_n)^2\right)\right),\\
\tilde{V} &= \left(V^{-1} + \tilde{X}_c^T \tilde{X}_c\right)^{-1},\\
\hat{m}_n &= \tilde{V}\left(m_0 V^{-1} + \tilde{X}_c^T Y_c\right),\\
m_n &= \tilde{X}_i \hat{m}_n,\\
n_n &= n_{y0} + n_c,\\
s_n^2 &= 4\left(s_{y0}^2 + 1/2\left(m_0 V^{-1} m_0^T + Y_c^T Y_c - \hat{m}_n^T \tilde{V}^{-1} \hat{m}_n\right)\right) / \left((n_{y0}+n_c)\tilde{X}_c \tilde{V} \tilde{X}_c^T\right).
\end{aligned}$$

Here, we define $\tilde{X}_c = \{[1X_j] : z_j = z_c\}$, $Y_c = \{Y_j : z_j = z_c\}$, $\tilde{X}_i = [1X_i]$, n_c is the number of data associated with label z_c and the base measure is define as,

$$\begin{aligned}
\sigma_y^2 &\sim Inverse-Gamma(n_{y0}, s_{y0}^2),\\
\beta \,|\, \sigma_y^2 &\sim N(m_0, \sigma_y^2 V).
\end{aligned}$$

Appendix B.

Proofs for the main theorems.

B.1 Proof of Theorem 1

Both Theorems 1 and 2 rely on a theorem by Schwartz (1965).

Theorem 5 (Schwartz, 1965) *Let Π^f be a prior on $\mathcal{F}$. Then, if Π^f places positive probability on all neighborhoods*

$$\left\{f : \int f_0(x,y)\log\frac{f_0(x,y)}{f(x,y)}dxdy < \delta\right\}$$

for every $\delta > 0$, then Π^f is weakly consistent at f_0.

The proof for Theorem 1 follows closely both Ghosal et al. (1999) and Tokdar (2006).

Proof Without loss of generality, assume $d = 1$. Since f_0 has compact support, there exists an x_0 and a y_0 such that $f_0(x,y) = 0$ for $|x| > x_0$ or $|y| > y_0$. Fix $\varepsilon > 0$. Following Remark 3 of Ghosal et al. (1999), there exist $\sigma_x > 0$ and $\sigma_y > 0$ such that

$$\int_{-x_0}^{x_0}\int_{-y_0}^{y_0} f_0(x,y)\log\frac{f_0(x,y)}{\int_{-x_0}^{x_0}\int_{-y_0}^{y_0}\phi(\frac{x-\theta_x}{\sigma_x})\phi(\frac{y-\theta_y}{\sigma_y})f_0(x,y)d\theta_x d\theta_y} < \varepsilon/2.$$

Let P_0 be a measure on $\mathbb{R}^3 \times \mathbb{R}_+^2$, that is, a measure for $(\mu_x, \beta_0, \beta_1, \sigma_x, \sigma_y)$. Define it such that $dP_0 = f_0 \times \delta_0 \times \delta_{\sigma_x} \times \delta_{\sigma_y}$. Fix a $\lambda > 0$ and $\kappa > 0$. Choose a large compact set K such that $[-x_0, x_0] \times [-y_0, y_0] \times [-y_0, y_0] \times \{\sigma_x\} \times \{\sigma_y\} \subset K$.Let $\mathcal{B} = \{P : |P(K)/P_0(K) - 1| < \kappa\}$. Since the support of $\mathbb{G}_0$ is $\mathbb{R}^3 \times \mathbb{R}_+^2$, $\Pi(\mathcal{B}) > 0$.

Following Ghosal et al. (1999) and Tokdar (2006), it can be shown that there exists a set $\mathcal{C}$ such that $\Pi(\mathcal{B} \cap \mathcal{C}) > 0$ and for every $P \in \mathcal{B} \cap \mathcal{C}$,

$$\int_{-x_0}^{x_0}\int_{-y_0}^{y_0} f_0(x,y)\log\frac{\int_K \phi(\frac{x-\mu_x}{\sigma_x})\phi(\frac{y-\beta_0-\beta_1 x}{\sigma_y})dP_0}{\int_K \phi(\frac{x-\mu_x}{\sigma_x})\phi(\frac{y-\beta_0-\beta_1 x}{\sigma_y})dP} < \frac{\kappa}{1-\kappa} + 2\kappa < \varepsilon/2$$

for a suitable choice of κ. Therefore, for $f = \phi * P$ for every $P \in \mathcal{B} \cap \mathcal{C}$,

$$\begin{aligned}\int f_0(x,y)\log\frac{f_0(x,y)}{f(x,y)}dxdy \leq{}& \int_{-x_0}^{x_0}\int_{-y_0}^{y_0} f_0(x,y)\log\frac{f_0(x,y)}{\int_{-x_0}^{x_0}\int_{-y_0}^{y_0}\phi(\frac{x-\theta_x}{\sigma_x})\phi(\frac{y-\theta_y}{\sigma_y})f_0(x,y)d\theta_x d\theta_y}\\ &+\int_{-x_0}^{x_0}\int_{-y_0}^{y_0} f_0(x,y)\log\frac{\int_K \phi(\frac{x-\mu_x}{\sigma_x})\phi(\frac{y-\beta_0-\beta_1 x}{\sigma_y})dP_0}{\int_K \phi(\frac{x-\mu_x}{\sigma_x})\phi(\frac{y-\beta_0-\beta_1 x}{\sigma_y})dP}\\ &< \varepsilon.\end{aligned}$$

Therefore, Π^f places positive measure on all weak neighborhoods of f_0, and hence satisfies Theorem 5.

■

Proof [Theorem 2] The proof of Theorem 2 follows along the same lines as the proof for Theorem 1. Instead of the continuous response, however, there is a categorical response. The continuity condition on the response probabilities ensures that there exists a $y_0 > 0$ such that there are m continuous functions $b_1(x), \ldots, b_m(x)$ with $|b_i(x)| < y_0$ and

$$\mathbb{P}_{f_0}[Y = i\,|\,X = x] = \frac{\exp(b_i(x))}{\sum_{j=1}^m \exp(b_j(x))}.$$

Using arguments similar to those in the previous proof, there exists $\sigma_x > 0$ such that,

$$\int_{-x_0}^{x_0} f_0(x,i) \log \frac{f_0(x,i)}{\int_{-x_0}^{x_0} \phi(\frac{x-\theta_x}{\sigma_x}) f_0(x) \frac{\exp(b_i(x))}{\sum_{j=1}^m \exp(b_j(x))} d\theta_x} < \varepsilon/2.$$

Define P_0 such that $dP_0 = f_0(x) \times \{\sigma_x\} \times b_1(x) \times \cdots \times b_m(x)$. The rest of the proof follows as previously, with small modifications. ■

B.2 Proof of Theorem 3

We now show pointwise convergence of the conditional densities. The following propositions will be used to prove Theorems 3 and 4.Let $f_n(x,y)$ be the Bayes estimate of the density under Π^f after n observations,

$$f_n(x,y) = \int_{\mathcal{F}} f(x,y) \Pi^f\left(df \,|\, (X_i,Y_i)_{i=1}^n\right).$$

Proposition 6 *Weak consistency of Π^f at f_0 for the Gaussian model and the multinomial model implies that $f_n(x,y)$ converges pointwise to $f_0(x,y)$ and $f_n(x)$ converges pointwise to $f_0(x)$ for (x,y) in the compact support of f_0.*

Proof Both $f_n(x,y)$ and $f_n(x)$ can be written as expectations of bounded functions with respect to the posterior measure. In the Gaussian case, both $f_n(x,y)$ and $f_n(x)$ are absolutely continuous; in the multinomial case, $f_n(x)$ is absolutely continuous while the probability $\mathbb{P}_{f_n}[Y = k \,|\, x]$ is absolutely continuous in x for $k = 1, \dots, K$. Due to absolute continuity, the result holds. ■

This can be used to show that the conditional density estimate converges pointwise to the true conditional density.

Proposition 7 *Let $f_n(x,y)$ an $f_n(x)$ be as in Proposition 6. Then $f_n(y|x)$ converges pointwise to $f_0(y|x)$ for any (x,y) in the compact support of f_0.*

Proof From Proposition 6, $f_n(x,y)$ converges pointwise to $f_0(x,y)$ and $f_n(x)$ converges pointwise to $f_0(x)$. Then,

$$\lim_{n\to\infty} f_n(y|x) = \lim_{n\to\infty} \frac{f_n(x,y)}{f_n(x)} = \frac{f_0(x,y)}{f_0(x)} = f_0(y|x).$$

The denominator value, $f_n(x)$, is greater than 0 almost surely because it is a mixture of Gaussian densities. ■

Now we proceed to the proof of Theorem 3.
Proof [Theorem 3] The conditions for Theorem 3 assure that Propositions 6 and 7 hold. Because of this and the fact that $\mathbb{G}_0$ places positive measure only on densities with a finite expectation, the results hold. ■

B.3 Proof of Theorem 4

The proof follows in the same manner as that for Theorem 3.

Appendix C.

Implementation details.

C.1 CMB Computational Details

The DP-GLM was run on the largest data size tested several times; log posterior probabilities were evaluated graphically, and in each case the posterior probabilities seem to have stabilized well before 1,000 iterations. Therefore, all runs for each sample size were given a 1,000 iteration burn-in with samples taken every 5 iterations until 2,000 iterations had been observed. The scaling parameter α was given a Gamma prior with shape and scale set to 1. The means and variances of each component and all GLM parameters were also given a log-normal hyper distribution. The model was most sensitive to the hyper-distribution on σ_y, the GLM variance. Small values were used ($\log(m_y) \sim N(-3,2)$) to place greater emphasis on response fit. The non-conjugate parameters were updated using the Hamiltonian dynamics method of Neal (2010). Hyperparameters were chosen based on performance on a subset of 100 data points; values were then held fixed all other data sets. This may produce an overly confident error assessment, but the limited size of the data set did not allow a pure training-validation-testing three way partition. A non-conjugate base measure was used on this data set due to small sample sizes and heteroscedasticity. The conjugate measure, a normal-inverse-gamma, assumes a relationship between the variance and the mean,

$$\mu \,|\, \sigma^2, \lambda, \nu \sim N(\lambda, \sigma^2/\nu).$$

Therefore, smaller variances greatly encourage the mean μ to remain in a small neighborhood around around the prior value, λ. Naturally, this property can be overcome with many observations, but it makes strong statements about the mean in situations with few total samples or few samples per cluster due to heteroscedasticity. This model was implemented in Matlab; a run on the largest data set took about 500 seconds.

C.2 CCS Computational Details

Again, the DP-GLM was run on the largest data size tested several times; log posterior probabilities were evaluated graphically, and in each case the posterior probabilities seem to have stabilized well before 1,000 iterations. Therefore, all runs for each sample size were given a 1,000 iteration burn-in with samples taken every 5 iterations until 2,000 iterations had been observed. The scaling parameter α was given a Gamma prior with shape and scale set to 1. The hyperparameters of the conjugate base measure were set manually by trying different settings over four orders of magnitude for each parameter on a single subset of training data. Again, this may produce an overly confident error assessment, but the limited size of the data set did not allow a pure training-validation-testing three way partition. All base measures were conjugate, so the sampler was fully collapsed. α was updated using Hamiltonian dynamics (Neal, 2010). Original results were generated by Matlab; the longest run times were about 1000 seconds. This method has been re-implemented in Java in a highly efficient manner; the longest run times are now under about 10 seconds. Run times would likely be even faster if variational methods were used for posterior sampling (Blei and Jordan, 2006).

C.3 Solar Computational Details

Again, the DP-GLM was run on the largest data set size tested several times; log posterior probabilities were evaluated graphically, and in each case the posterior probabilities seem to have stabilized well before 1,000 iterations. Therefore, all runs for each sample size were given a 1,000 iteration burn-in with samples taken every 5 iterations until 2,000 iterations had been observed. The scaling parameter α was set to 1 and the Dirichlet priors to $Dir(1,1,\dots,1)$. The response parameters were given a Gaussian base distribution with a mean set to 0 and a variance chosen after trying parameters with four orders of magnitude on a fixed training data set. This may produce an overly confident error assessment, but the limited size of the data set did not allow a pure training-validation-testing three way partition. All covariate base measures were conjugate and the β base measure was Gaussian, so the sampler was collapsed along the covariate dimensions and used in the auxiliary component setting of Algorithm 8 of Neal (2000). The β parameters were updated using Metropolis-Hastings. Results were in generated by Matlab; run times were substantially faster than the other methods implemented in Matlab (under 200 seconds).

References

R. P. Adams, I. Murray, and D. J. C. MacKay. Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 9–16. ACM, 2009.

M. Amewou-Atisso, S. Ghosal, J. K. Ghosh, and R. V. Ramamoorthi. Posterior consistency for semi-parametric regression problems. *Bernoulli*, 9(2):291–312, 2003.

C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

A. Barron, M. J. Schervish, and L. Wasserman. The consistency of posterior distributions in nonparametric problems. *The Annals of Statistics*, 27(2):536–561, 1999.

C. L. Bennett, M. Halpern, G. Hinshaw, N. Jarosik, A. Kogut, M. Limon, S. S. Meyer, L. Page, D. N. Spergel, G. S. Tucker, et al. First-year Wilkinson microwave anisotropy probe (WMAP) 1 observations: preliminary maps and basic results. *The Astrophysical Journal Supplement Series*, 148(1):1–27, 2003.

D. Blackwell and J. B. MacQueen. Ferguson distributions via Polya urn schemes. *The Annals Statistics*, 1(2):353–355, 1973.

D. M. Blei and M. I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–144, 2006.

G. Bradshaw. UCI machine learning repository, 1989.

L. Brieman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, New York, NY, 1984.

L. D. Brown. *Fundamentals of Statistical Exponential Families: with Applications in Statistical Decision Theory*. Institute of Mathematical Statistics, Hayward, CA, 1986.

H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian treed models. *Machine Learning*, 48 (1):299–320, 2002.

M. De Iorio, P. Muller, G. L. Rosner, and S. N. MacEachern. An ANOVA model for dependent random measures. *Journal of the American Statistical Association*, 99(465):205–215, 2004.

J. A. Duan, M. Guindani, and A. E. Gelfand. Generalized spatial Dirichlet process models. *Biometrika*, 94(4):809–825, 2007.

D. B. Dunson, N. Pillai, and J. H. Park. Bayesian density regression. *Journal of the Royal Statistical Society Series B, Statistical Methodology*, 69(2):163–183, 2007.

M. D. Escobar. Estimating normal means with a Dirichlet process prior. *Journal of the American Statistical Association*, 89(425):268–277, 1994.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1 (2):209–230, 1973.

A. E. Gelfand, A. Kottas, and S. N. MacEachern. Bayesian nonparametric spatial modeling with Dirichlet process mixing. *Journal of the American Statistical Association*, 100(471):1021–1035, 2005.

A. Gelman, J. B. Carlin, H. S. Stern, and D. S. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, FL, 2004.

S. Ghosal, J. K. Ghosh, and R. V. Ramamoorthi. Posterior consistency of Dirichlet mixtures in density estimation. *The Annals of Statistics*, 27(1):143–158, 1999.

J. K. Ghosh and R. V. Ramamoorthi. *Bayesian Nonparametrics*. Springer-Verlag New York, Inc., New York, NY, 2003.

R. B. Gramacy and H. K. H. Lee. Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

J. E. Griffin and M. F. J. Steel. Order-based dependent Dirichlet processes. *Journal of the American Statistical Association*, 101(473):179–194, 2006.

J. E. Griffin and M. F. J. Steel. Bayesian nonparametric modelling with the Dirichlet process regression smoother. *Statistica Sinica*, 20(4):1507–1527, 2010.

J. G. Ibrahim and K. P. Kleinman. Semiparametric Bayesian methods for random effects models. In *Practical Nonparametric and Semiparametric Bayesian Statistics*, pages 89–114. 1998.

S. N. MacEachern. Estimating normal means with a conjugate style Dirichlet process prior. *Communications in Statistics-Simulation and Computation*, 23(3):727–741, 1994.

S. N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphical Statistics*, 7(2):223–238, 1998.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Boca Raton, FL, 1989.

S. Mukhopadhyay and A. E. Gelfand. Dirichlet process mixed generalized linear models. *Journal of the American Statistical Association*, 92(438):633–639, 1997.

P. Müller, A. Erkanli, and M. West. Bayesian curve fitting using multivariate normal mixtures. *Biometrika*, 83(1):67–79, 1996.

E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9(1):141–142, 1964.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2010.

P. Z. G. Qian, H. Wu, and C. F. J. Wu. Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics*, 50(3):383–396, 2008.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems, 14*.

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

A. Rodrıguez. *Some Advances in Bayesian Nonparametric Modeling*. PhD thesis, Duke University, 2009.

A. Rodriguez, D. B. Dunson, and A. E. Gelfand. Bayesian nonparametric functional data analysis through density estimation. *Biometrika*, 96(1):149–162, 2009.

M. A. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7): 1649–1681, 2001.

L. Schwartz. On Bayes procedures. *Probability Theory and Related Fields*, 4(1):10–26, 1965.

B. Shahbaba and R. M. Neal. Nonlinear models using Dirichlet process mixtures. *Journal of Machine Learning Research*, 10:1829–1850, 2009.

S. Tokdar. Posterior consistency of Dirichlet location-scale mixture of normals in density estimation and regression. *Sankhya: The Indian Journal of Statistics*, 67:90–110, 2006.

S. Walker. New approaches to Bayesian consistency. *The Annals of Statistics*, 32(5):2028–2043, 2004.

S. G. Walker, A. Lijoi, and I. Prunster. On rates of convergence for posterior distributions in infinite-dimensional models. *Annals of Statistics*, 35(2):738, 2007.

G. S. Watson. Smooth regression analysis. *Sankhya: The Indian Journal of Statistics*, 26(4):359–372, 1964.

M. West, P. Muller, and M. D. Escobar. Hierarchical priors and mixture models, with application in regression and density estimation. In *Aspects of Uncertainty: A Tribute to DV Lindley*, pages 363–386. 1994.

I. C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998.

Journal of Machine Learning Research 12 (2011) 1955-1976 Submitted 2/11; Revised 4/11; Published 6/11

Kernel Regression in the Presence of Correlated Errors

Kris De Brabanter KRIS.DEBRABANTER@ESAT.KULEUVEN.BE
Department of Electrical Engineering SCD-SISTA
K.U. Leuven
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium

Jos De Brabanter JOS.DEBRABANTER@ESAT.KULEUVEN.BE
Departement Industrieel Ingenieur - E&A
KaHo Sint Lieven (Associatie K.U. Leuven)
G. Desmetstraat 1
B-9000 Gent, Belgium

Johan A.K. Suykens JOHAN.SUYKENS@ESAT.KULEUVEN.BE
Bart De Moor BART.DEMOOR@ESAT.KULEUVEN.BE
Department of Electrical Engineering SCD-SISTA
K.U. Leuven
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium

Editor: Xiaotong Shen

Abstract

It is a well-known problem that obtaining a correct bandwidth and/or smoothing parameter in non-parametric regression is difficult in the presence of correlated errors. There exist a wide variety of methods coping with this problem, but they all critically depend on a tuning procedure which requires accurate information about the correlation structure. We propose a bandwidth selection procedure based on bimodal kernels which successfully removes the correlation without requiring any prior knowledge about its structure and its parameters. Further, we show that the form of the kernel is very important when errors are correlated which is in contrast to the independent and identically distributed (i.i.d.) case. Finally, some extensions are proposed to use the proposed criterion in support vector machines and least squares support vector machines for regression.

Keywords: nonparametric regression, correlated errors, bandwidth choice, cross-validation, short-range dependence, bimodal kernel

1. Introduction

Nonparametric regression is a very popular tool for data analysis because these techniques impose few assumptions about the shape of the mean function. Hence, they are extremely flexible tools for uncovering nonlinear relationships between variables. Given the data $\{(x_1,Y_1),\dots,(x_n,Y_n)\}$ where $x_i \equiv i/n$ and $x \in [0,1]$ (fixed design). Then, the data can be written as

$$Y_i = m(x_i) + e_i, \qquad i = 1,\dots,n, \tag{1}$$

where $e_i = Y_i - m(x_i)$ satisfies $\mathbf{E}[e] = 0$ and $\mathbf{Var}[e] = \sigma^2 < \infty$. Thus Y_i can be considered as the sum of the value of the regression function at x_i and some error e_i with the expected value zero and the sequence $\{e_i\}$ is a covariance stationary process.

Definition 1 (Covariance Stationarity) *The sequence $\{e_i\}$ is covariance stationary if*

- $\mathbf{E}[e_i] = \mu$ *for all i*
- $\mathbf{Cov}[e_i, e_{i-j}] = \mathbf{E}[(e_i - \mu)(e_{i-j} - \mu)] = \gamma_j$ *for all i and any j.*

Many techniques include a smoothing parameter and/or kernel bandwidth which controls the smoothness, bias and variance of the estimate. A vast number of techniques have been developed to determine suitable choices for these tuning parameters from data when the errors are independent and identically distributed (i.i.d.) with finite variance. More detailed information can be found in the books of Fan & Gijbels (1996), Davison & Hinkley (2003) and Konishi & Kitagawa (2008) and the article by Feng & Heiler (2009). However, all the previous techniques have been derived under the i.i.d. assumption. It has been shown that violating this assumption results in the break down of the above methods (Altman, 1990; Hermann, Gasser & Kneip, 1992; Opsomer, Wand & Yang, 2001; Lahiri, 2003). If the errors are positively (negatively) correlated, these methods will produce a small (large) bandwidth which results in a rough (oversmooth) estimate of the regression function. The focus of this paper is to look at the problem of estimating the mean function m in the presence of correlation, not that of estimating the correlation function itself. Approaches describing the estimation of the correlation function are extensively studied in Hart & Wehrly (1986), Hart (1991) and Park et al. (2006).

Another issue in this context is whether the errors are assumed to be short-range dependent, where the correlation decreases rapidly as the distance between two observations increases or long-range dependent. The error process is said to be short-range dependent if for some $\tau > 0$, $\delta > 1$ and correlation function $\rho(\cdot)$, the spectral density $H(\omega) = \frac{\sigma^2}{2\pi}\sum_{k=-\infty}^{\infty}\rho(k)e^{-i\omega}$ of the errors satisfies (Cox, 1984)

$$H(\omega) \sim \tau\omega^{-(1-\delta)} \text{ as } \omega \to 0,$$

where $A \sim B$ denotes A is asymptotic equivalent to B. In that case, $\rho(j)$ is of order $|j|^{-\delta}$ (Adenstedt, 1974). In case of long-range dependence, the correlation decreases more slowly and regression estimation becomes even harder (Hall, Lahiri & Polzehl, 1995; Opsomer, Wand & Yang, 2001). Here, the decrease is of order $|j|^{-\delta}$ for $0 < \delta \leq 1$. Estimation under long-range dependence has attracted more and more attention in recent years. In many scientific research fields such as astronomy, chemistry, physics and signal processing, the observational errors sometimes reveal long-range dependence. Künsch, Beran & Hampel (1993) made the following interesting remark:

> "*Perhaps most unbelievable to many is the observation that high-quality measurements series from astronomy, physics, chemistry, generally regarded as prototype of i.i.d. observations, are not independent but long-range correlated.*"

Further, since Kulkarni et al. (2002) have proven consistency for the data-dependent kernel estimators, that is, correlated errors and/or correlation among the independent variables, there is no need to alter the kernel smoother by adding constraints. Confirming their results, we show that the problem is due to the model selection criterion. In fact, we will show in Section 3 that there exists a simple multiplicative relation between the bandwidth under correlation and the bandwidth under the i.i.d. assumption.

In the parametric case, ordinary least squares estimators in the presence of autocorrelation are still linear-unbiased as well as consistent, but they are no longer efficient (i.e., minimum variance). As a result, the usual confidence intervals and the test hypotheses cannot be legitimately applied (Sen & Srivastava, 1990).

2. Problems With Correlation

Some quite fundamental problems occur when nonparametric regression is attempted in the presence of correlated errors. For all nonparametric regression techniques, the shape and the smoothness of the estimated function depends on a large extent on the specific value(s) chosen for the kernel bandwidth (and/or regularization parameter). In order to avoid selecting values for these parameters by trial and error, several data-driven methods are developed. However, the presence of correlation between the errors, if ignored, causes breakdown of commonly used automatic tuning parameter selection methods such as cross-validation (CV) or plug-in.

Data-driven bandwidth selectors tend to be “fooled” by the correlation, interpreting it as reflecting the regression relationship and variance function. So, the cyclical pattern in positively correlated errors is viewed as a high frequency regression relationship with small variance, and the bandwidth is set small enough to track the cycles resulting in an undersmoothed fitted regression curve. The alternating pattern above and below the true underlying function for negatively correlated errors is interpreted as a high variance, and the bandwidth is set high enough to smooth over the variability, producing an oversmoothed fitted regression curve.

The breakdown of automated methods, as well as a suitable solution, is illustrated by means of a simple example shown in Figure 1. For 200 equally spaced observations and a polynomial mean function $m(x) = 300x^3(1-x)^3$, four progressively more correlated sets of errors were generated from the same vector of independent noise and added to the mean function. The errors are normally distributed with variance $\sigma^2 = 0.3$ and correlation following an Auto Regressive process of order 1, denoted by AR(1), $\operatorname{corr}(e_i, e_j) = \exp(-\alpha|x_i - x_j|)$ (Fan & Yao, 2003). Figure 1 shows four local linear regression estimates for these data sets. For each data set, two bandwidth selection methods were used: standard CV and a correlation-corrected CV (CC-CV) which is further discussed in Section 3. Table 1 summarizes the bandwidths selected for the four data sets under both methods.

Table 1 and Figure 1 clearly show that when correlation increases, the bandwidth selected by CV becomes smaller and smaller, and the estimates become more undersmoothed. The bandwidths selected by CC-CV (explained in Section 3), a method that accounts for the presence of correlation, are much more stable and result in virtually the same estimate for all four cases. This type of undersmoothing behavior in the presence of positively correlated errors has been observed with most commonly used automated bandwidth selection methods (Altman, 1990; Hart, 1991; Opsomer, Wand & Yang, 2001; Kim et al., 2009).

3. New Developments in Kernel Regression with Correlated Errors

In this Section, we address how to deal with, in a simple but effective way, correlated errors using CV. We make a clear distinction between kernel methods requiring no positive definite kernel and kernel methods requiring a positive definite kernel. We will also show that the form of the kernel,

Correlation level	**Autocorrelation**	**CV**	**CC-CV**
Independent	0	0.09	0.09
$\alpha = 400$	0.14	0.034	0.12
$\alpha = 200$	0.37	0.0084	0.13
$\alpha = 100$	0.61	0.0072	0.13

Table 1: Summary of bandwidth selection for simulated data in Figure 1

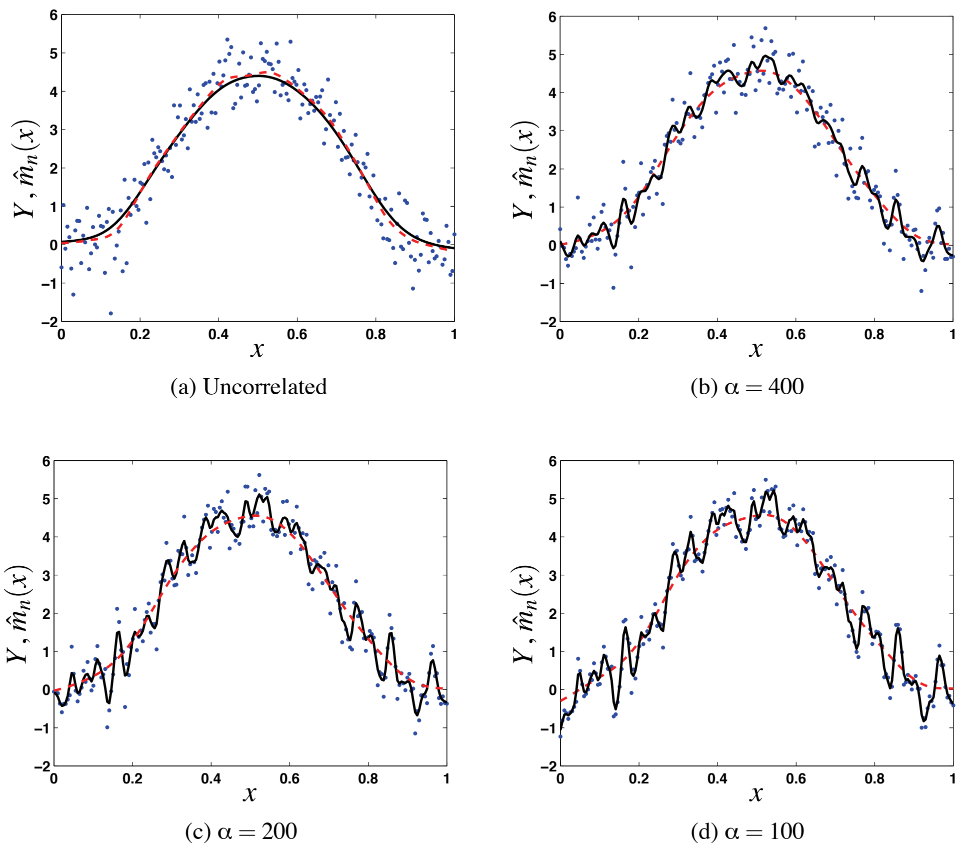


Figure 1: Simulated data with four levels of AR(1) correlation, estimated with local linear regression; full line represents the estimates obtained with bandwidth selected by CV; dashed line represents the estimates obtained with bandwidth selected by our method.

based on the mean squared error, is very important when errors are correlated. This is in contrast with the i.i.d. case where the choice between the various kernels, based on the mean squared error, is not very crucial (Härdle, 1999). In what follows, the kernel K is assumed to be an isotropic kernel.

3.1 No Positive Definite Kernel Constraint

To estimate the unknown regression function m, consider the Nadaraya-Watson (NW) kernel estimator (Nadaraya, 1964; Watson, 1964) defined as

$$\hat{m}_n(x) = \sum_{i=1}^{n} \frac{K(\frac{x-x_i}{h})Y_i}{\sum_{j=1}^{n} K(\frac{x-x_j}{h})},$$

where h is the bandwidth of the kernel K. This kernel can be one of the following kernels: Epanechnikov, Gaussian, triangular, spline, etc. An optimal h can for example be found by minimizing the leave-one-out cross-validation (LCV) score function

$$\text{LCV}(h) = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \hat{m}_n^{(-i)}(x_i;h) \right)^2, \tag{2}$$

where $\hat{m}_n^{(-i)}(x_i;h)$ denotes the leave-one-out estimator where point i is left out from the training. For notational ease, the dependence on the bandwidth h will be suppressed. We can now state the following.

Lemma 2 *Assume that the errors are zero-mean, then the expected value of the LCV score function* (2) *is given by*

$$\mathbf{E}[\text{LCV}(h)] = \frac{1}{n} \mathbf{E} \left[\sum_{i=1}^{n} \left(m(x_i) - \hat{m}_n^{(-i)}(x_i) \right)^2 \right] + \sigma^2 - \frac{2}{n} \sum_{i=1}^{n} \mathbf{Cov} \left[\hat{m}_n^{(-i)}(x_i), e_i \right].$$

Proof: see Appendix A. ■

Note that the last term on the right-hand side in Lemma 2 is in addition to the correlation already included in the first term. Hart (1991) shows, if $n \to \infty$, $nh \to \infty$, $nh^5 \to 0$ and for positively correlated errors, that $\mathbf{E}[\text{LCV}(h)] \approx \sigma^2 + c/nh$ where $c < 0$ and c does not depend on the bandwidth. If the correlation is sufficiently strong and n sufficiently large, $\mathbf{E}[\text{LCV}(h)]$ will be minimized at a value of h that is very near to zero. The latter corresponds to almost interpolating the data (see Figure 1). This result does not only hold for leave-one-out cross-validation but also for Mallow's criterion (Chiu, 1989) and plug-in based techniques (Opsomer, Wand & Yang, 2001). The following theorem provides a simple but effective way to deal with correlated errors. In what follows we will use the following notation

$$k(u) = \int_{-\infty}^{\infty} K(y) e^{-iuy}\, dy$$

for the Fourier Transform of the kernel function K.

Theorem 3 *Assume uniform equally spaced design, $x \in [0,1]$, $\mathbf{E}[e] = 0$, $\mathbf{Cov}[e_i, e_{i+k}] = \mathbf{E}[e_i e_{i+k}] = \gamma_k$ and $\gamma_k \sim k^{-a}$ for some $a > 2$. Assume that*

(C1) K is Lipschitz continuous at $x = 0$;

(C2) $\int K(u)\, du = 1, \lim_{|u|\to\infty} |uK(u)| = 0, \int |K(u)|\, du < \infty, \sup_u |K(u)| < \infty$;

(C3) $\int |k(u)|\, du < \infty$ and K is symmetric.

Assume further that boundary effects are ignored and that $h \to 0$ as $n \to \infty$ such that $nh^2 \to \infty$, then for the NW smoother it follows that

$$\mathbf{E}[\mathrm{LCV}(h)] = \frac{1}{n}\mathbf{E}\left[\sum_{i=1}^{n}\left(m(x_i) - \hat{m}_n^{(-i)}(x_i)\right)^2\right] + \sigma^2 - \frac{4K(0)}{nh - K(0)}\sum_{k=1}^{\infty}\gamma_k + o(n^{-1}h^{-1}). \tag{3}$$

Proof: see Appendix B. ■

Remark 4 *There are no major changes in the proof if we consider other smoothers such as Priestley-Chao and local linear regression. In fact, it is well-known that the local linear estimate is the local constant estimate (Nadaraya-Watson) plus a correction for local slope of the data and skewness of the data point under consideration. Following the steps of the proof of Theorem 3 for the correction factor will yield a similar result.*

From this result it is clear that, by taking a kernel satisfying the condition $K(0) = 0$, the correlation structure is removed without requiring any prior information about its structure and (3) reduces to

$$\mathbf{E}[\mathrm{LCV}(h)] = \frac{1}{n}\mathbf{E}\left[\sum_{i=1}^{n}\left(m(x_i) - \hat{m}_n^{(-i)}(x_i)\right)^2\right] + \sigma^2 + o(n^{-1}h^{-1}). \tag{4}$$

Therefore, it is natural to use a bandwidth selection criterion based on a kernel satisfying $K(0) = 0$, defined by

$$\hat{h}_b = \arg\min_{h \in Q_n} \mathrm{LCV}(h),$$

where Q_n is a finite set of parameters. Notice that if K is a symmetric probability density function, then $K(0) = 0$ implies that K is not unimodal. Hence, it is obvious to use bimodal kernels. Such a kernel gives more weight to observations near to the point x of interest than those that are far from x. But at the same time it also reduces the weight of points which are too close to x. A major advantage of using a bandwidth selection criterion based on bimodal kernels is the fact that is more efficient in removing the correlation than leave-$(2l+1)$-out CV (Chu & Marron, 1991).

Definition 5 (Leave-$(2l+1)$-out CV) *Leave-$(2l+1)$-out CV or modified CV (MCV) is defined as*

$$\mathrm{MCV}(h) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{m}_n^{(-i)}(x_i)\right)^2, \tag{5}$$

where $\hat{m}_n^{(-i)}(x_i)$ is the leave-$(2l+1)$-out version of $m(x_i)$, that is, the observations (x_{i+j}, Y_{i+j}) for $-l \le j \le l$ are left out to estimate $\hat{m}_n(x_i)$.

Taking a bimodal kernel satisfying $K(0) = 0$ results in Equation (4) while leave-$(2l+1)$-out CV with unimodal kernel K, under the conditions of Theorem 3, yields

$$\mathbf{E}[\mathrm{MCV}(h)] = \frac{1}{n}\mathbf{E}\left[\sum_{i=1}^{n}\left(m(x_i) - \hat{m}_n^{(-i)}(x_i)\right)^2\right] + \sigma^2 - \frac{4K(0)}{nh - K(0)}\sum_{k=l+1}^{\infty}\gamma_k + o(n^{-1}h^{-1}).$$

The formula above clearly shows that leave-$(2l+1)$-out CV with unimodal kernel K cannot completely remove the correlation structure. Only the first l elements of the correlation are removed.

Another possibility of bandwidth selection under correlation, not based on bimodal kernels, is to estimate the covariance structure $\gamma_0, \gamma_1, \ldots$ in Equation (3). Although the usual residual-based estimators of the autocovariances $\hat{\gamma}_k$ are consistent, $\sum_{k=1}^{\infty} \hat{\gamma}_k$ is not a consistent estimator of $\sum_{k=1}^{\infty} \gamma_k$ (Simonoff, 1996). A first approach correcting for this, is to estimate $\sum_{k=1}^{\infty} \gamma_k$ by fitting a parametric model to the residuals (and thereby obtaining estimates of γ_k) and use these estimates in Equation (3) together with a univariate kernel. If the assumed parametric model is incorrect, these estimates can be far from the correct ones resulting in a poor choice of the bandwidth. However, Altman (1990) showed that, if the signal to noise ratio is small, this approach results in sufficiently good estimates of correlation for correcting the selection criteria. A second approach, proposed by Hart (1989, 1991), suggests estimating the covariance structure in the spectral domain via differencing the data at least twice. A third approach is to derive an asymptotic bias-variance decomposition under the correlated error assumption of the kernel smoother. In this way and under certain conditions on the correlation function, plug-ins can be derived taking the correlation into account, see Hermann, Gasser & Kneip (1992), Opsomer, Wand & Yang (2001), Hall & Van Keilegom (2003), Francisco-Fernández & Opsomer (2004) and Francisco-Fernández et al. (2005). More recently, Park et al. (2006) proposed to estimate the error correlation nonparametrically without prior knowledge of the correlation structure.

3.2 Positive Definite Kernel Constraint

Methods like support vector machines (SVM) (Vapnik, 1999) and least squares support vector machines (LS-SVM) (Suykens et al., 2002) require a positive (semi) definite kernel (see Appendix C for more details on LS-SVM for regression). However, the following theorem reveals why a bimodal kernel $\tilde{K}$ cannot be directly applied in these methods.

Theorem 6 *A bimodal kernel $\tilde{K}$, satisfying $\tilde{K}(0) = 0$, is never positive (semi) definite.*

Proof: see Appendix D. ■

Consequently, the previous strategy of using bimodal kernels cannot directly be applied to SVM and LS-SVM. A possible way to circumvent this obstacle, is to use the bandwidth $\hat{h}_b$, obtained from the bimodal kernel, as a pilot bandwidth selector for other data-driven selection procedures such as leave-$(2l+1)$-out CV or block bootstrap bandwidth selector (Hall, Lahiri & Polzehl, 1995). Since the block bootstrap in Hall, Lahiri & Polzehl (1995) is based on two smoothers, that is, one is used to compute centered residuals and the other generates bootstrap data, the procedure is computationally costly. Therefore, we will use leave-$(2l+1)$-out CV or MCV which has a lower computational cost. A crucial parameter to be estimated in MCV, see also Chu & Marron (1991), is l. Indeed, the amount of dependence between $\hat{m}_n(x_k)$ and Y_k is reduced as l increases.

A similar problem arises in block bootstrap where the accuracy of the method critically depends on the block size that is supplied by the user. The orders of magnitude of the optimal block sizes are known in some inference problems (see Künsch, 1989; Hall, Horowitz & Jing, 1995; Lahiri, 1999; Bühlmann & Künsch, 1999). However, the leading terms of these optimal block sizes depend on various population characteristics in an intricate manner, making it difficult to estimate these parameters in practice. Recently, Lahiri et al. (2007) proposed a nonparametric plug-in principle to determine the block size.

For $l = 0$, MCV is ordinary CV or leave-one-out CV. One possible method to select a value for l is to use $\hat{h}_b$ as pilot bandwidth selector. Define a bimodal kernel $\tilde{K}$ and assume $\hat{h}_b$ is available, then one can calculate

$$\hat{m}_n(x) = \sum_{i=1}^{n} \frac{\tilde{K}\left(\frac{x-x_i}{\hat{h}_b}\right) Y_i}{\sum_{j=1}^{n} \tilde{K}\left(\frac{x-x_j}{\hat{h}_b}\right)}. \tag{6}$$

From this result, the residuals are obtained by

$$\hat{e}_i = Y_i - \hat{m}_n(x_i), \text{ for } i = 1, \ldots, n$$

and choose l to be the smallest $q \geq 1$ such that

$$|r_q| = \left| \frac{\sum_{i=1}^{n-q} \hat{e}_i \hat{e}_{i+q}}{\sum_{i=1}^{n} \hat{e}_i^2} \right| \leq \frac{\Phi^{-1}(1-\frac{\alpha}{2})}{\sqrt{n}}, \tag{7}$$

where Φ^{-1} denotes the quantile function of the standard normal distribution and α is the significance level, say 5%. Observe that Equation (7) is based on the fact that r_q is asymptotically normal distributed under the centered i.i.d. error assumption (Kendall, Stuart & Ord, 1983) and hence provides an approximate $100(1-\alpha)\%$ confidence interval for the autocorrelation. The reason why Equation (7) can be legitimately applied is motivated by combining the theoretical results of Kim et al. (2004) and Park et al. (2006) stating that

$$\frac{1}{n-q} \sum_{i=1}^{n-q} \hat{e}_i \hat{e}_{i+q} = \frac{1}{n-q} \sum_{i=1}^{n-q} e_i e_{i+q} + O(n^{-4/5}).$$

Once l is selected, the tuning parameters of SVM or LS-SVM can be determined by using leave-$(2l+1)$-out CV combined with a positive definite kernel, for example, Gaussian kernel. We then call Correlation-Corrected CV (CC-CV) the combination of finding l via bimodal kernels and using the obtained l in leave-$(2l+1)$-out CV. Algorithm 1 summarizes the CC-CV procedure for LS-SVM. This procedure can also be applied to SVM for regression.

Algorithm 1 Correlation-Corrected CV for LS-SVM Regression

```
1: Determine $\hat{h}_b$ in Equation (6) with a bimodal kernel by means of LCV
2: Calculate $l$ satisfying Equation (7)
3: Determine both tuning parameters for LS-SVM by means of leave-$(2l+1)$-out CV Equation (5)
   and a positive definite unimodal kernel.
```

3.3 Drawback of Using Bimodal Kernels

Although bimodal kernels are very effective in removing the correlation structure, they have an inherent drawback. When using bimodal kernels to estimate the regression function m, the estimate $\hat{m}_n$ will suffer from increased mean squared error (MSE). The following theorem indicates the asymptotic behavior of the MSE of $\hat{m}_n(x)$ when the errors are covariance stationary.

Theorem 7 (Simonoff, 1996) *Let Equation* (1) *hold and assume that m has two continuous derivatives. Assume also that* $\mathbf{Cov}[e_i, e_{i+k}] = \gamma_k$ *for all k, where* $\gamma_0 = \sigma^2 < \infty$ *and* $\sum_{k=1}^{\infty} k|\gamma_k| < \infty$. *Now, as*

$n \to \infty$ and $h \to 0$, the following statement holds uniformly in $x \in (h, 1-h)$ for the Mean Integrated Squared Error (MISE)

$$\mathrm{MISE}(\hat{m}_n) = \frac{\mu_2^2(K)h^4 \int (m''(x))^2\,dx}{4} + \frac{R(K)[\sigma^2 + 2\sum_{k=1}^{\infty}\gamma_k]}{nh} + o(h^4 + n^{-1}h^{-1}),$$

where $\mu_2(K) = \int u^2 K(u)\,du$ and $R(K) = \int K^2(u)\,du$.

An asymptotic optimal constant or global bandwidth $\hat{h}_{\mathrm{AMISE}}$, for $m''(x) \neq 0$, is the minimizer of the Asymptotic MISE (AMISE)

$$\mathrm{AMISE}(\hat{m}_n) = \frac{\mu_2^2(K)h^4 \int (m''(x))^2\,dx}{4} + \frac{R(K)[\sigma^2 + 2\sum_{k=1}^{\infty}\gamma_k]}{nh},$$

w.r.t. to the bandwidth, yielding

$$\hat{h}_{\mathrm{AMISE}} = \left[\frac{R(K)[\sigma^2 + 2\sum_{k=1}^{\infty}\gamma_k]}{\mu_2^2(K)\int (m''(x))^2\,dx}\right]^{1/5} n^{-1/5}. \tag{8}$$

We see that $\hat{h}_{\mathrm{AMISE}}$ is at least as big as the bandwidth for i.i.d data $\hat{h}_0$ if $\gamma_k \geq 0$ for all $k \geq 1$. The following corollary shows that there is a simple multiplicative relationship between the asymptotic optimal bandwidth for dependent data $\hat{h}_{\mathrm{AMISE}}$ and bandwidth for independent data $\hat{h}_0$.

Corollary 8 *Assume the conditions of Theorem 7 hold, then*

$$\hat{h}_{\mathrm{AMISE}} = \left[1 + 2\sum_{k=1}^{\infty}\rho(k)\right]^{1/5} \hat{h}_0, \tag{9}$$

where $\hat{h}_{\mathrm{AMISE}}$ is the asymptotic MISE optimal bandwidth for dependent data, $\hat{h}_0$ is the asymptotic optimal bandwidth for independent data and $\rho(k)$ denotes the autocorrelation function at lag k, that is, $\rho(k) = \gamma_k/\sigma^2 = \mathbf{E}[e_i e_{i+k}]/\sigma^2$.

Proof: see Appendix E. ■

Thus, if the data are positively autocorrelated ($\rho(k) \geq 0 \quad \forall k$), the optimal bandwidth under correlation is larger than that for independent data. Unfortunately, Equation (9) is quite hard to use in practice since it requires knowledge about the correlation structure and an estimate of the bandwidth $\hat{h}_0$ under the i.i.d. assumption, given correlated data. By taking $\hat{h}_{\mathrm{AMISE}}$ as in Equation (8), the corresponding asymptotic MISE is equal to

$$\mathrm{AMISE}(\hat{m}_n) = cD_K^{2/5} n^{-4/5},$$

where c depends neither on the bandwidth nor on the kernel K and

$$D_K = \mu_2(K)R(K)^2 = \left(\int u^2 K(u)\,du\right)\left(\int K^2(u)\,du\right)^2. \tag{10}$$

It is obvious that one wants to minimize Equation (10) with respect to the kernel function K. This leads to the well-known Epanechnikov kernel K_{epa}. However, adding the constraint $K(0) = 0$ (see Theorem 3) to the minimization of Equation (10) would lead to the following optimal kernel

$$K^{\star}(u) = \begin{cases} K_{\text{epa}}(u), & \text{if } u \neq 0; \\ 0, & \text{if } u = 0. \end{cases}$$

Certainly, this kernel violates assumption (C1) in Theorem 3. In fact, an optimal kernel does not exist in the class of kernels satisfying assumption (C1) and $K(0) = 0$. To illustrate this, note that there exist a sequence of kernels $\{K_{\text{epa}}(u,\varepsilon)\}_{\varepsilon\in]0,1[}$, indexed by ε, such that $K_{\text{epa}}(u)$ converges to $K^{\star}(u)$ and the value of $\int K_{\text{epa}}(u,\varepsilon)^2 du$ decreases to $\int K^{\star}(u)^2 du$ as ε tends to zero. Since an optimal kernel in this class cannot be found, we have to be satisfied with a so-called ε-optimal class of bimodal kernels $\tilde{K}_{\varepsilon}(u)$, with $0 < \varepsilon < 1$, defined as

$$\tilde{K}_{\varepsilon}(u) = \frac{4}{4 - 3\varepsilon - \varepsilon^3} \begin{cases} \frac{3}{4}(1-u^2) I_{\{|u|\leq 1\}}, & |u| \geq \varepsilon; \\ \frac{3}{4}\frac{1-\varepsilon^2}{\varepsilon}|u|, & |u| < \varepsilon. \end{cases}$$

For $\varepsilon = 0$, we define $\tilde{K}_{\varepsilon}(u) = K_{\text{epa}}(u)$. Table 2 displays several possible bimodal kernel functions with their respective D_K value compared to the Epanechnikov kernel. Although it is possible to express the D_K value for $\tilde{K}_{\varepsilon}(u)$ as a function of ε, we do not include it in Table 2 but instead, we graphically illustrate the dependence of D_K on ε in Figure 2a. An illustration of the ε-optimal class of bimodal kernels is shown in Figure 2b.

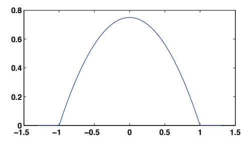
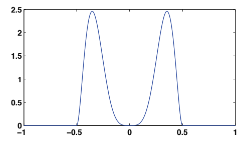
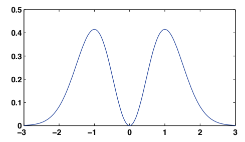
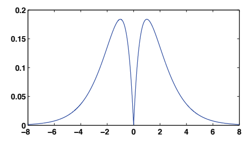
	kernel function	**Illustration**	**$\mathbf{D_K}$**
K_{epa}	$\frac{3}{4}(1-u^2)I_{\{\lvert u\rvert\leq 1\}}$		0.072
$\tilde{K}_1$	$630(4u^2-1)^2u^4 I_{\{\lvert u\rvert\leq 1/2\}}$		0.374
$\tilde{K}_2$	$\frac{2}{\sqrt{\pi}}u^2\exp(-u^2)$		0.134
$\tilde{K}_3$	$\frac{1}{2}\lvert u\rvert\exp(-\lvert u\rvert)$		0.093

Table 2: Kernel functions with illustrations and their respective D_K value compared to the Epanechnikov kernel. I_A denotes the indicator function of an event A.

Remark 9 *We do not consider ε as a tuning parameter but the user can set its value. By doing this one should be aware of two aspects. First, one should choose the value of ε so that its D_K*

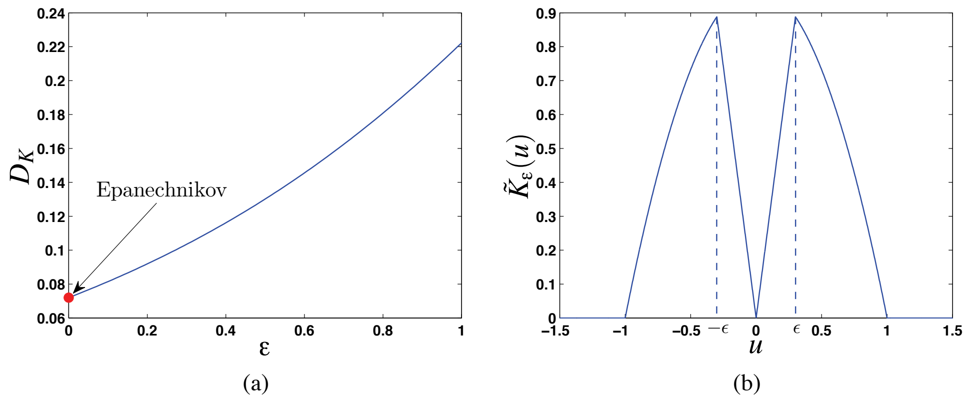


Figure 2: (a) D_K as a function of ε for the ε-optimal class of kernels. The dot on the left side marks the Epanechnikov kernel; (b) Illustration of the ε-optimal class of kernels for $\varepsilon = 0.3$.

value is lower than the D_K value of kernel $\tilde{K}_3$. This is fulfilled when $\varepsilon < 0.2$. Second, by choosing ε extremely small (but not zero) some numerical difficulties may arise. We have experimented with several values of ε and we concluded that the value taken in the remaining of the paper, that is, $\varepsilon = 0.1$ is small enough and it does not show any numerical problems. In theory, there is indeed a difference between kernel $\tilde{K}_3$ and the ε-optimal class of bimodal kernels. However, in practice the difference is rather small. One can compare it with the i.i.d. case where the Epanechnikov kernel is the optimal kernel, but in practice the difference with say a Gaussian kernel is negligible.

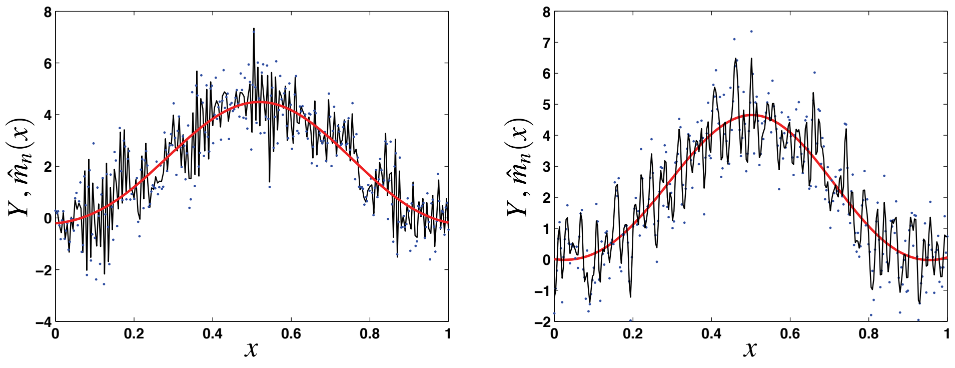
4. Simulations

In this Section, we illustrate the capability of the proposed method on several toy examples corrupted with different noise models as well as a real data set.

4.1 CC-CV vs. LCV with Different Noise Models

In a first example, we compare the finite sample performance of CC-CV (with $\tilde{K}_\varepsilon$ and $\varepsilon = 0.1$ in the first step and the Gaussian kernel in the second step) to the classical leave-one-out CV (LCV) based on the Epanechnikov (unimodal) kernel in the presence of correlation. Consider the following function $m(x) = 300x^3(1-x)^3$ for $0 \le x \le 1$. The sample size is set to $n = 200$. We consider two types of noise models: (i) an AR(5) process $e_j = \sum_{l=1}^{5} \phi_l e_{j-l} + \sqrt{1-\phi_1^2} Z_j$ where Z_j are i.i.d. normal random variables with variance $\sigma^2 = 0.5$ and zero mean. The data is generated according to Equation (1). The errors e_j for $j = 1, \ldots, 5$ are standard normal random variables. The AR(5) parameters are set to $[\phi_1, \phi_2, \phi_3, \phi_4, \phi_5] = [0.7, -0.5, 0.4, -0.3, 0.2]$. (ii) m-dependent models $e_i = r_0\delta_i + r_1\delta_{i-1}$ with $m = 1$ where δ_i is i.i.d. standard normal random variable, $r_0 = \frac{\sqrt{1+2\nu}+\sqrt{1-2\nu}}{2}$ and $r_1 = \frac{\sqrt{1+2\nu}-\sqrt{1-2\nu}}{2}$ for $\nu = 1/2$.

Figure 3 shows typical results of LS-SVM regression estimates for both noise models. Table 3 summarizes the average of the regularization parameters $\hat{\gamma}$, bandwidths $\hat{h}$ and asymptotic squared error, defined as $\text{ASE} = \frac{1}{n}\sum_{i=1}^{n}(m(x_i) - \hat{m}_n(x_i))^2$, for 200 runs for both noise models. By looking at the average ASE, it is clear that the tuning parameters obtained by CC-CV result into better estimates which are not influenced by the correlation. Also notice the small bandwidths and larger regularization constants found by LCV for both noise models. This provides clear evidence that the kernel smoother is trying to model the noise instead of the true underlying function. These findings are also valid if one uses generalized CV or v-fold CV. Figure 4 and Figure 5 show the CV surfaces



(a) AR(5) (b) m-dependence models

Figure 3: Typical results of the LS-SVM regression estimates for both noise models. The thin line represents the estimate with tuning parameters determined by LCV and the bold line is the estimate based on the CC-CV tuning parameters.

	AR(5)		m-dependence models	
	LCV	CC-CV	LCV	CC-CV
av. $\hat{\gamma}$	226.24	2.27	1.05×10^5	6.87
av. $\hat{h}$	0.014	1.01	0.023	1.88
av. ASE	$0.39\ (2.9 \times 10^{-2})$	$0.019\ (9.9 \times 10^{-4})$	$0.90\ (8.2 \times 10^{-2})$	$0.038\ (1.4 \times 10^{-3})$

Table 3: Average of the regularization parameters $\hat{\gamma}$, bandwidths $\hat{h}$ and average ASE for 200 runs for both noise models. The standard deviation is given between parenthesis.

for both model selection methods on the AR(5) noise model corresponding to the model selection of the estimate in Figure 3(a). These plots clearly demonstrate the shift of the tuning parameters. A cross section for both tuning parameters is provided below each surface plot. Also note that the surface of the CC-CV tends to be flatter than LCV and so it is harder to minimize numerically (see Hall, Lahiri & Polzehl, 1995).

4.2 Evolution of the Bandwidth Under Correlation

Consider the same function as in the previous simulation and let $n = 400$. The noise error model is taken to be an AR(1) process with varying parameter $\phi = -0.95, -0.9, \ldots, 0.9, 0.95$. For each ϕ, 100 replications of size n were made to report the average regularization parameter, bandwidth and average ASE for both methods. The results are summarized in Table 4. We used the $\tilde{K}_\varepsilon$ kernel with $\varepsilon = 0.1$ in the first step and the Gaussian kernel in the second step for CC-CV and the Gaussian kernel for classical leave-one-out CV (LCV). The results indicate that the CC-CV method is indeed capable of finding good tuning parameters in the presence of correlated errors. The CC-CV method

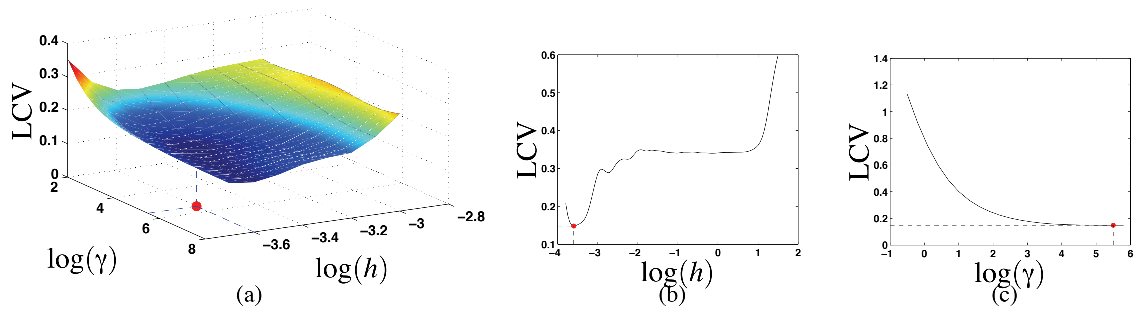


Figure 4: (a) CV surface for LCV; (b) cross sectional view of $\log(h)$ for fixed $\log(\gamma) = 5.5$; (c) cross sectional view of $\log(\gamma)$ for fixed $\log(h) = -3.6$. The dot indicates the minimum of the cost function. This corresponds to the model selection of the wiggly estimate in Figure 3(a).

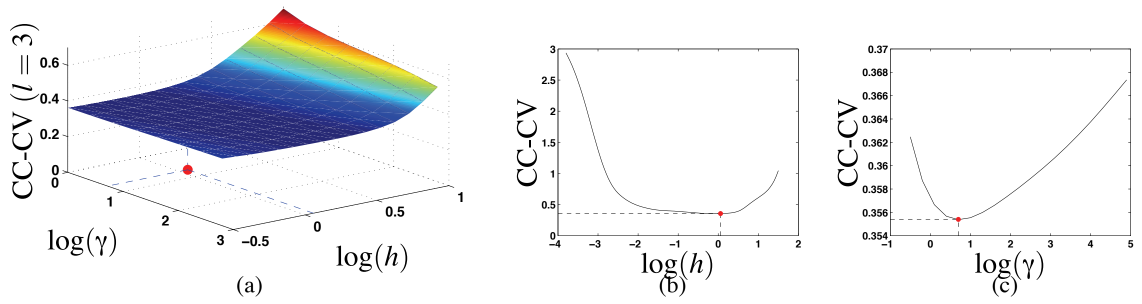


Figure 5: (a) CV surface for CC-CV; (b) cross sectional view of $\log(h)$ for fixed $\log(\gamma) = 0.82$; (c) cross sectional view of $\log(\gamma)$ for fixed $\log(h) = 0.06$. The dot indicates the minimum of the cost function. This corresponds to the model selection of the smooth estimate in Figure 3(a).

outperforms the classical LCV for positively correlated errors, that is, $\phi > 0$. The method is capable of producing good bandwidths which do not tend to very small values as in the LCV case.

In general, the regularization parameter obtained by LCV is larger than the one from CC - CV. However, the latter is not theoretically verified and serves only as a heuristic. On the other hand, for negatively correlated errors ($\phi < 0$), both methods perform equally well. The reason why the effects from correlated errors is more outspoken for positive ϕ than for negative ϕ might be related to the fact that negatively correlated errors are seemingly hard to differentiate form i.i.d. errors in practice.

4.3 Comparison of Different Bimodal Kernels

Consider a polynomial mean function $m(x_k) = 300x_k^3(1-x_k)^3$, $k = 1,\ldots,400$, where the errors are normally distributed with variance $\sigma^2 = 0.1$ and correlation following an AR(1) process, $\text{corr}(e_i,e_j) = \exp(-150|x_i - x_j|)$. The simulation shows the difference in regression estimates (Nadaraya-Watson) based on kernels $\tilde{K}_1$, $\tilde{K}_3$ and $\tilde{K}_\varepsilon$ with $\varepsilon = 0.1$, see Figure 6a and 6b respectively. Due to the larger D_K value of $\tilde{K}_1$, the estimate tends to be more wiggly compared to kernel $\tilde{K}_3$. The difference be-

ϕ	LCV			CC-CV		
	$\hat{\gamma}$	$\hat{h}$	av. ASE	$\hat{\gamma}$	$\hat{h}$	av. ASE
-0.95	14.75	1.48	0.0017	7.65	1.43	0.0019
-0.9	11.48	1.47	0.0017	14.58	1.18	0.0021
-0.8	7.52	1.39	0.0021	18.12	1.15	0.0031
-0.7	2.89	1.51	0.0024	6.23	1.21	0.0030
-0.6	28.78	1.52	0.0030	5.48	1.62	0.0033
-0.5	42.58	1.71	0.0031	87.85	1.75	0.0048
-0.4	39.15	1.55	0.0052	39.02	1.43	0.0060
-0.3	72.91	1.68	0.0055	19.76	1.54	0.0061
-0.2	98.12	1.75	0.0061	99.56	1.96	0.0069
-0.1	60.56	1.81	0.0069	101.1	1.89	0.0070
0	102.5	1.45	0.0091	158.4	1.89	0.0092
0.1	1251	1.22	0.0138	209.2	1.88	0.0105
0.2	1893	0.98	0.0482	224.6	1.65	0.0160
0.3	1535	0.66	0.11	5.18	1.86	0.0161
0.4	482.3	0.12	0.25	667.5	1.68	0.023
0.5	2598	0.04	0.33	541.8	1.82	0.033
0.6	230.1	0.03	0.36	986.9	1.85	0.036
0.7	9785	0.03	0.41	12.58	1.68	0.052
0.8	612.1	0.03	0.45	1531	1.53	0.069
0.9	448.8	0.02	0.51	145.12	1.35	0.095
0.95	878.4	0.01	0.66	96.5	1.19	0.13

Table 4: Average of the regularization parameters, bandwidths and average ASE for 100 runs for the AR(1) process with varying parameter ϕ

tween the regression estimate based on $\tilde{K}_3$ and $\tilde{K}_\varepsilon$ with $\varepsilon = 0.1$ is very small and almost cannot be seen on Figure 6b. For illustration purposes we did not visualize the result based on kernel $\tilde{K}_2$. For the sake of comparison between regression estimates based on $\tilde{K}_1$, $\tilde{K}_2$,$\tilde{K}_3$ and $\tilde{K}_\varepsilon$ with $\varepsilon = 0.1$, we show the corresponding asymptotic squared error (ASE) in Figure 7 based on 100 simulations with the data generation process described as above. The boxplot shows that the kernel $\tilde{K}_\varepsilon$ with $\varepsilon = 0.1$ outperforms the other three.

4.4 Real Life Data Set

We apply the proposed method to a time series of the Beveridge (1921) index of wheat prices from the year 1500 to 1869 (Anderson, 1971). These data are an annual index of prices at which wheat was sold in European markets. The data used for analysis are the natural logarithms of the Beveridge indices. This transformation is done to correct for heteroscedasticity in the original series (no other preprocessing was performed). The result is shown in Figure 8 for LS-SVM with Gaussian kernel. It is clear that the estimate based on classical leave-one-out CV (assumption of no correlation) is very rough. The proposed CC-CV method produces a smooth regression fit. The selected parameters $(\hat{\gamma},\hat{h})$ for LS-SVM are $(15.61, 29.27)$ and $(96.91, 1.55)$ obtained by CC-CV and LCV respectively.

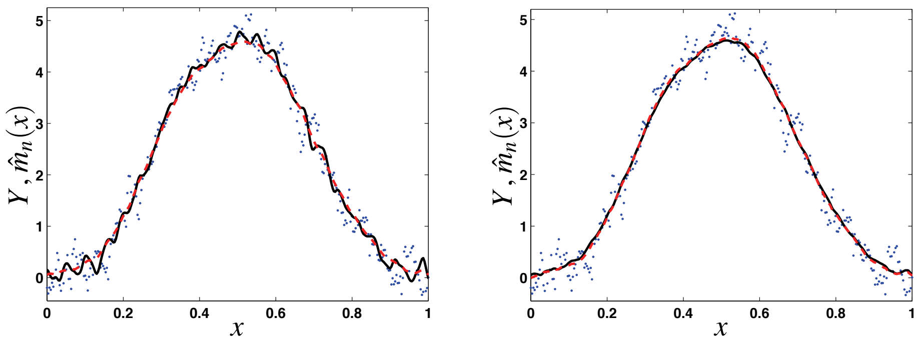


Figure 6: Difference in the regression estimate (Nadaraya-Watson) (a) based on kernel $\tilde{K}_1$ (full line) and $\tilde{K}_3$ (dashed line). Due to the larger D_K value of $\tilde{K}_1$, the estimate tends to be more wiggly compared to $\tilde{K}_3$; (b) based on kernel $\tilde{K}_3$ (full line) and ε-optimal kernel with $\varepsilon = 0.1$ (dashed line).

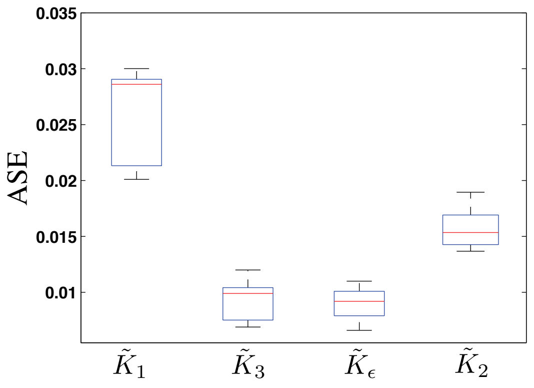


Figure 7: Boxplot of the asymptotic squared errors for the regression estimates based on bimodal kernels $\tilde{K}_1, \tilde{K}_2, \tilde{K}_3$ and $\tilde{K}_\varepsilon$ with $\varepsilon = 0.1$.

5. Conclusion

We have introduced a new type of cross-validation procedure, based on bimodal kernels, in order to automatically remove the error correlation without requiring any prior knowledge about its structure. We have shown that the form of the kernel is very important when errors are correlated. This in contrast with the i.i.d. case where the choice between the various kernels on the basis of the mean squared error is not very important. As a consequence of the bimodal kernel choice the estimate suffers from increased mean squared error. Since an optimal bimodal kernel (in mean squared error sense) cannot be found we have proposed a ε-optimal class of bimodal kernels. Further, we have used the bandwidth of the bimodal kernel as pilot bandwidth selector for leave-$(2l+1)$-out cross-validation. By taking this extra step, methods that require a positive definite kernel (SVM and LS-SVM) can be equipped with this technique of handling data in the presence of correlated errors since they require a positive definite kernel. Also other kernel methods which do not require positive definite kernels can benefit from the proposed method.

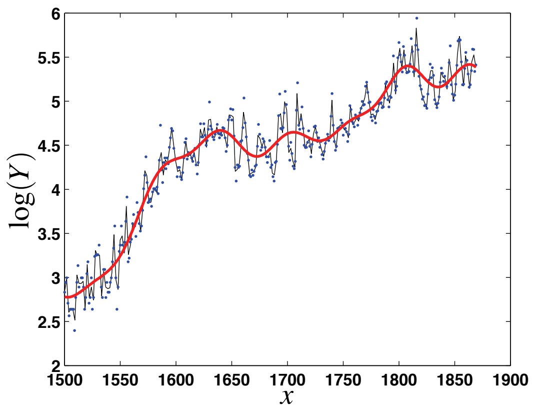


Figure 8: Difference in regression estimates (LS-SVM) for standard leave-one-out CV (thin line) and the proposed method (bold line).

Acknowledgments

The authors would like to thank Prof. László Györfi and Prof. Irène Gijbels for their constructive comments which improved the results of the paper.
Research supported by Research Council KUL: GOA/11/05 Ambiorics, GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC) en PFV/10/002 (OPTEC), IOF-SCORES4CHEM, several PhD/post-doc & fellow grants; Flemish Government: FWO: FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G0321.06 (Tensors), G.0302.07 (SVM/ Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC), IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare, Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011), IBBT, EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), Contract Research: AMINAL, Other: Helmholtz, viCERP, ACCM. BDM is a full professor at the Katholieke Universiteit Leuven, Belgium. JS is a professor at the Katholieke Universiteit Leuven, Belgium.

Appendix A. Proof of Lemma 2

We first rewrite the LCV score function in a more workable form. Since $Y_i = m(x_i) + e_i$

$$\begin{aligned}
\mathrm{LCV}(h) &= \frac{1}{n}\sum_{i=1}^{n}\left[Y_i - \hat{m}_{(-i)}(x_i)\right]^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[m^2(x_i) + 2m(x_i)e_i + e_i^2 - 2Y_i\hat{m}_n^{(-i)}(x_i) + \left(\hat{m}_n^{(-i)}(x_i)\right)^2\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[m(x_i) - \hat{m}_n^{(-i)}(x_i)\right]^2 + \frac{1}{n}\sum_{i=1}^{n} e_i^2 \\
&\quad + \frac{2}{n}\sum_{i=1}^{n}\left[m(x_i) - \hat{m}_n^{(-i)}(x_i)\right] e_i.
\end{aligned}$$

Taking expectations, yields

$$\mathbf{E}[\mathrm{LCV}(h)] = \frac{1}{n}\mathbf{E}\left[\sum_{i=1}^{n}\left(m(x_i) - \hat{m}_n^{(-i)}(x_i)\right)^2\right] + \sigma^2 - \frac{2}{n}\sum_{i=1}^{n}\mathbf{Cov}\left[\hat{m}_n^{(-i)}(x_i), e_i\right].$$

Appendix B. Proof of Theorem 3

Consider only the last term of the expected LCV (Lemma 2), that is,

$$A(h) = -\frac{2}{n}\sum_{i=1}^{n}\mathbf{Cov}\left[\hat{m}_n^{(-i)}(x_i), e_i\right].$$

Plugging in the Nadaraya-Watson kernel smoother for $\hat{m}_n^{(-i)}(x_i)$ in the term above yields

$$A(h) = -\frac{2}{n}\sum_{i=1}^{n}\mathbf{Cov}\left[\sum_{j\neq i}^{n}\frac{K\left(\frac{x_i-x_j}{h}\right)Y_j}{\sum_{l\neq i}^{n}K\left(\frac{x_i-x_l}{h}\right)}, e_i\right].$$

By using the linearity of the expectation operator, $Y_j = m(x_j) + e_j$ and $\mathbf{E}[e] = 0$ it follows that

$$\begin{aligned} A(h) &= -\frac{2}{n}\sum_{i=1}^{n}\sum_{j\neq i}^{n}\mathbf{E}\left[\frac{K\left(\frac{x_i-x_j}{h}\right)Y_j}{\sum_{j\neq i}^{n}K\left(\frac{x_i-x_l}{h}\right)}e_i\right] \\ &= -\frac{2}{n}\sum_{i=1}^{n}\sum_{j\neq i}^{n}\frac{K\left(\frac{x_i-x_j}{h}\right)}{\sum_{j\neq i}^{n}K\left(\frac{x_i-x_l}{h}\right)}\mathbf{E}\left[e_ie_j\right]. \end{aligned}$$

By slightly rewriting the denominator and using the covariance stationary property of the errors (Definition 1), the above equation can be written as

$$A(h) = -\frac{2}{n}\sum_{i=1}^{n}\sum_{j\neq i}^{n}\frac{K\left(\frac{x_i-x_j}{h}\right)}{\sum_{j=1}^{n}K\left(\frac{x_i-x_l}{h}\right) - K(0)}\gamma_{|i-j|}. \tag{11}$$

Let f denote the design density. The first term of the denominator can be written as

$$\begin{aligned} \sum_{j=1}^{n}K\left(\frac{x_i-x_l}{h}\right) &= nh\hat{f}(x_i) \\ &= nhf(x_i) + nh(\hat{f}(x_i) - f(x_i)). \end{aligned}$$

If conditions (C2) and (C3) are fulfilled, f is uniform continuous and $h \to \infty$ as $n \to \infty$ such that $nh^2 \to \infty$, then

$$|\hat{f}(x_i) - f(x_i)| \leq \sup_{x_i}|\hat{f}(x_i) - f(x_i)| \xrightarrow{P} 0 \quad \text{as } n \to \infty,$$

due to the uniform weak consistency of the kernel density estimator (Parzen, 1962). $\xrightarrow{P}$ denotes convergence in probability. Hence, for $n \to \infty$, the following approximation is valid

$$nh\hat{f}(x_i) \approx nhf(x_i).$$

Further, by grouping terms together and using the fact that $x_i \equiv i/n$ (uniform equispaced design) and assume without loss of generality that $x \in [0,1]$, Equation (11) can be written as

$$\begin{aligned} A(h) &= -\frac{2}{n}\sum_{i=1}^{n}\frac{1}{nhf(x_i)-K(0)}\sum_{j\neq i}^{n} K\left(\frac{x_i-x_j}{h}\right)\gamma_{|i-j|} \\ &= -\frac{4}{nh-K(0)}\sum_{k=1}^{n-1}\left(\frac{n-k}{n}\right)K\left(\frac{k}{nh}\right)\gamma_k. \end{aligned}$$

Next, we show that $\sum_{k=1}^{n-1}\left(\frac{n-k}{n}\right)K\left(\frac{k}{nh}\right)\gamma_k = K(0)\sum_{k=1}^{\infty}\gamma_k + o(n^{-1}h^{-1})$ for $n\to\infty$. Since the kernel $K \geq 0$ is Lipschitz continuous at $x=0$

$$[K(0)+C_2x]_+ \leq K(x) \leq K(0)+C_1x,$$

where $[z]_+ = \max(z,0)$. Then, for $K(0)\geq 0$ and $C_1 > C_2$, we establish the following upperbound

$$\begin{aligned} \sum_{k=1}^{n-1}\left(\frac{n-k}{n}\right)K\left(\frac{k}{nh}\right)\gamma_k &\leq \sum_{k=1}^{n-1}\left(1-\frac{k}{n}\right)\left(K(0)+C_1\frac{k}{nh}\right)\gamma_k \\ &\leq \sum_{k=1}^{n-1}K(0)\gamma_k + \sum_{k=1}^{n-1}C_1\frac{k}{nh}\gamma_k. \end{aligned}$$

Then, for $n\to\infty$ and using $\gamma_k \sim k^{-a}$ for $a>2$,

$$C_1\sum_{k=1}^{n-1}\frac{k}{nh}\gamma_k = C_1\sum_{k=1}^{n-1}\frac{k^{1-a}}{nh} = o(n^{-1}h^{-1}).$$

Hence,

$$\sum_{k=1}^{n-1}\left(\frac{n-k}{n}\right)K\left(\frac{k}{nh}\right)\gamma_k \leq K(0)\sum_{k=1}^{\infty}\gamma_k + o(n^{-1}h^{-1}).$$

For the construction of the lower bound, assume first that $C_2 < 0$ and $K(0)\geq 0$ then

$$\sum_{k=1}^{n-1}\left(\frac{n-k}{n}\right)K\left(\frac{k}{nh}\right)\gamma_k \geq \sum_{k=1}^{n-1}\left(1-\frac{k}{n}\right)\left[K(0)+C_2\frac{k}{nh}\right]_+\gamma_k.$$

Since $C_2<0$, it follows that $k \leq \frac{K(0)}{-C_2}nh$ and therefore

$$\sum_{k=1}^{n-1}\left(1-\frac{k}{n}\right)\left[K(0)+C_2\frac{k}{nh}\right]_+\gamma_k = \sum_{k=1}^{\min\left(n-1,\frac{K(0)}{-C_2}nh\right)}\left(1-\frac{k}{n}\right)\left(K(0)+C_2\frac{k}{nh}\right)\gamma_k.$$

Analogous to deriving the upper bound, we obtain for $n\to\infty$

$$\sum_{k=1}^{n-1}\left(\frac{n-k}{n}\right)K\left(\frac{k}{nh}\right)\gamma_k \geq K(0)\sum_{k=1}^{\infty}\gamma_k + o(n^{-1}h^{-1}).$$

In the second case, that is, $C_2>0$, the same lower bound can be obtained. Finally, from the upper and lower bound, for $n\to\infty$, yields

$$\sum_{k=1}^{n-1}\left(\frac{n-k}{n}\right)K\left(\frac{k}{nh}\right)\gamma_k = K(0)\sum_{k=1}^{\infty}\gamma_k + o(n^{-1}h^{-1}).$$

Appendix C. Least Squares Support Vector Machines for Regression

Given a training set defined as $\mathcal{D}_n = \{(x_k, Y_k) : x_k \in \mathbb{R}^d, Y_k \in \mathbb{R}; k = 1, \dots, n\}$. Then least squares support vector machines for regression are formulated as follows (Suykens et al., 2002)

$$\begin{aligned} \min_{w,b,e} \mathcal{J}(w,e) &= \tfrac{1}{2} w^T w + \tfrac{\gamma}{2} \sum_{k=1}^{n} e_k^2 \\ \text{s.t.} \quad Y_k &= w^T \varphi(x_k) + b + e_k, \quad k = 1, \dots, n, \end{aligned} \tag{12}$$

where $e_k \in \mathbb{R}$ are assumed to be i.i.d. random errors with zero mean and finite variance, $\varphi : \mathbb{R}^d \to \mathbb{R}^{n_h}$ is the feature map to the high dimensional feature space (possibly infinite dimensional) and $w \in \mathbb{R}^{n_h}$, $b \in \mathbb{R}$. The cost function $\mathcal{J}$ consists of a residual sum of squares (RSS) fitting error and a regularization term (with regularization parameter γ) corresponding to ridge regression in the feature space with additional bias term.

However, one does not need to evaluate w and φ explicitly. By using Lagrange multipliers, the solution of Equation (12) can be obtained by taking the Karush-Kuhn-Tucker (KKT) conditions for optimality. The result is given by the following linear system in the dual variables α

$$\left(\begin{array}{c|c} 0 & 1_n^T \\ \hline 1_n & \Omega + \frac{1}{\gamma} I_n \end{array}\right) \left(\begin{array}{c} b \\ \hline \alpha \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline Y \end{array}\right),$$

with $Y = (Y_1, \dots, Y_n)^T$, $1_n = (1, \dots, 1)^T$, $\alpha = (\alpha_1, \dots, \alpha_n)^T$ and $\Omega_{kl} = \varphi(x_k)^T \varphi(x_l) = K(x_k, x_l)$, with $K(x_k, x_l)$ positive definite, for $k, l = 1, \dots, n$. According to Mercer's theorem, the resulting LS-SVM model for function estimation becomes

$$\hat{m}_n(x) = \sum_{k=1}^{n} \hat{\alpha}_k K(x, x_k) + \hat{b},$$

where $K(\cdot,\cdot)$ is an appropriately chosen positive definite kernel. In this paper we choose K to be the Gaussian kernel, that is, $K(x_k, x_l) = (2\pi)^{-d/2} \exp\left(\frac{-\|x_k - x_l\|^2}{2h^2}\right)$.

Appendix D. Proof of Theorem 6

We split up the proof in two parts, that is, for positive definite and positive semi-definite kernels. The statement will be proven by contradiction.

- Suppose there exists a positive definite bimodal kernel $\tilde{K}$. This leads to a positive definite kernel matrix Ω. Then, all eigenvalues of Ω are strictly positive and hence the trace of Ω is always larger than zero. However, this is in contradiction with the fact that Ω has all zeros on its main diagonal. Consequently, a positive definite bimodal kernel $\tilde{K}$ cannot exist.

- Suppose there exists a positive semi-definite bimodal kernel $\tilde{K}$. Then, at least one eigenvalue of the matrix Ω is equal to zero (the rest of the eigenvalues is strictly positive). We have now two possibilities, that is, some eigenvalues are equal to zero and all eigenvalues are equal to zero. In the first case, the trace of the matrix Ω is larger than zero and we have again a contradiction. In the second case, the trace of the matrix Ω is equal to zero and also the determinant of Ω equals zero (since all eigenvalues are equal to zero). But the determinant can never be zero since there is no linear dependence between the rows or columns (there is a zero in each row or column).

Appendix E. Proof of Corollary 8

From Equation (8) it follows that

$$\begin{aligned}\hat{h}_{\text{AMISE}} &= \left[\frac{R(K)\sigma^2}{n\mu_2^2(K)\int(m''(x))^2\,dx}+\frac{2R(K)\sum_{k=1}^{\infty}\gamma_k}{n\mu_2^2(K)\int(m''(x))^2\,dx}\right]^{1/5}\\ &= \left[\hat{h}_0^5+\frac{\sigma^2R(K)}{n\mu_2^2(K)\int(m''(x))^2\,dx}\,\frac{2\sum_{k=1}^{\infty}\gamma_k}{\sigma^2}\right]^{1/5}\\ &= \left[1+2\sum_{k=1}^{\infty}\rho(k)\right]^{1/5}\hat{h}_0.\end{aligned}$$

References

R.K. Adenstedt. On large sample estimation for the mean of a stationary sequence. *Ann. Statist.*, 2(6):1095–1107, 1974.

N.S. Altman. Kernel smoothing of data with correlated errors. *J. Amer. Statist. Assoc.*, 85(411):749–759, 1990.

T.W. Anderson. *The Statistical Analysis of Time Series*. Wiley, New York, 1971.

P. Bühlmann and H.R. Künsch. Block length selection in the bootstrap for time series. *Computational Statistics & Data Analysis*, 31(3):295310, 1999

S.-T. Chiu. Bandwidth selection for kernel estimate with correlated noise. *Statist. Probab. Lett.*, 8(4):347–354, 1989.

C.K. Chu and J.S. Marron. Comparison of two bandwidth selectors with dependent errors. *Ann. Statist.*, 19(4):1906–1918, 1991.

D.R. Cox. Long-range dependence: a review. In *Proceedings 50th Anniversary Conference. Statistics: An Appraisal*. pages 55–74, Iowa State Univ. Press.

A.C Davison and D.V. Hinkley. *Bootstrap Methods and their Application* (reprinted with corrections). Cambridge University Press, 2003.

J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications*. Chapmann & Hall, 1996.

J. Fan and Q. Yao. *Nonlinear Time Series: Nonparametric and Parametric Methods*. Springer, 2003.

Y. Feng and S. Heiler. A simple bootstrap bandwidth selector for local polynomial fitting. *J. Stat. Comput. Simul.*, 79(12):1425–1439, 2009.

M. Francisco-Fernández and J.D. Opsomer. Smoothing parameter selection methods for nonparametric regression with spacially correlated errors. *Canad. J. Statist.*, 33(2):279–295, 2004.

M. Francisco-Fernández, M., J.D. Opsomer and J.M. Vilar-Fernández. A plug-in bandwidth selector for local polynomial regression estimator with correlated errors. *J. Nonparametr. Stat.*, 18(1–2):127–151, 2005.

P. Hall, S.N. Lahiri and J. Polzehl. On bandwidth choice in nonparametric regression with both short- and long-range dependent errors. *Ann. Statist.*, 23(6):1921–1936, 1995.

P. Hall, J.L. Horowitz, and B.-Y. Jing. On blocking rules for the bootstrap with dependent data. *Biometrika*, 82(3):561574, 1995

P. Hall and I. Van Keilegom. Using difference-based methods for inference in nonparametric regression with time-series errors. *J. Roy. Statist. Assoc. Ser. B Stat. Methodol.*, 65(2):443–456, 2003.

W. Härdle. *Applied Nonparametric Regression* (Reprinted). Cambridge University Press, 1999.

J.D. Hart and T.E. Wehrly. Kernel regression estimation using repeated measurements data. *J. Amer. Statist. Assoc.*, 81(396):1080–1088, 1986.

J.D. Hart. Differencing as an approximate de-trending device. *Stoch. Processes Appl.*, 31(2):251–259, 1989.

J.D. Hart. Kernel regression estimation with time series errors. *J. Royal Statist. Soc. B*, 53(1):173–187, 1991.

E. Hermann, T. Gasser and A. Kneip. Choice of bandwidth for kernel regression when residuals are correlated. *Biometrika*, 79(4):783–795, 1992.

M.G. Kendall, A. Stuart and J. K. Ord. *The Advanced Theory of Statistics, vol. 3, Design and Analysis, and Time-Series* (4th ed.). Griffin, London, 1983.

T.Y. Kim, D. Kim, B.U. Park and D.G. Simpson. Nonparametric detection of correlated errors. *Biometrika*, 91(2):491–496, 2004.

T.Y. Kim, B.U. Park, M.S. Moon and C. Kim. Using bimodal kernel inference in nonparametric regression with correlated errors. *J. Multivariate Anal.*, 100(7):1487–1497, 2009.

S. Konishi and G. Kitagawa. *Information Criteria and Statistical Modeling*. Springer, 2008.

S.R. Kulkarni, S.E. Posner and S. Sandilya. Data-dependent k_n-NN and kernel estimators consistent for arbitrary processes. *IEEE Trans. Inform. Theory*, 48(10):2785–2788, 2002.

H. Künsch. The jackknife and the bootstrap for general stationary observations. *Ann. Statist.*, 17(3):12171261, 1989.

H. Künsch, J. Beran and F. Hampel. Contrasts under long-range correlations. *Ann. Statist.*, 21(2):943–964, 1993.

S.N. Lahiri. Theoretical comparisons of block bootstrap methods. *Ann. Statist.*, 27(1):386404, 1999.

S.N. Lahiri. *Resampling Methods for Dependent Data*. Springer, 2003.

S.N. Lahiri, K. Furukawa, and Y.-D. Lee. A nonparametric plug-in rule for selecting optimal block lengths for block bootstrap methods. *Statistical Methodology*, 4(3):292321, 2007.

E.A. Nadaraya. On estimating regression. *Theory Probab. Appl.*, 9(1):141–142, 1964.

J. Opsomer, Y. Wang and Y. Yang. Nonparametric regression with correlated errors. *Statist. Sci.*, 16(2):134–153, 2001.

B.U. Park, Y.K. Lee, T.Y. Kim and C. Park. A simple estimator of error correlation in non-parametric regression models. *Scand. J. Statist.*, 33(3):451–462, 2006.

E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):10651076, 1962

A. Sen and M. Srivastava. *Regression Analysis: Theory, Methods and Applications*. Springer, 1990.

J.S. Simonoff. *Smoothing Methods in Statistics*. Springer, 1996.

J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1999.

G.S. Watson. Smooth regression analysis. *Sankhya Ser. A*, 26(4):359–372, 1964.

Journal of Machine Learning Research 12 (2011) 1977-2020 Submitted 10/09; Revised 5/09; Published 6/11

Generalized TD Learning

Tsuyoshi Ueno TSUYOS-U@SYS.I.KYOTO-U.AC.JP
Shin-ichi Maeda ICHI@SYS.I.KYOTO-U.AC.JP
Graduate School of Informatics
Kyoto University
Gokasho, Uji, Kyoto, 611-0011 Japan

Motoaki Kawanabe[*] MOTOAKI.KAWANABE@FIRST.FRAUNHOFER.DE
Fraunhofer FIRST.IDA
Kekulestrasee 7
12489, Berlin, Germany

Shin Ishii ISHII@I.KYOTO-U.AC.JP
Graduate School of Informatics
Kyoto University
Gokasho, Uji, Kyoto, 611-0011 Japan

Editor: Shie Mannor

Abstract

Since the invention of temporal difference (TD) learning (Sutton, 1988), many new algorithms for model-free policy evaluation have been proposed. Although they have brought much progress in practical applications of reinforcement learning (RL), there still remain fundamental problems concerning statistical properties of the value function estimation. To solve these problems, we introduce a new framework, *semiparametric statistical inference*, to model-free policy evaluation. This framework generalizes TD learning and its extensions, and allows us to investigate statistical properties of both of batch and online learning procedures for the value function estimation in a unified way in terms of *estimating functions*. Furthermore, based on this framework, we derive an optimal estimating function with the *minimum asymptotic variance* and propose batch and online learning algorithms which achieve the optimality.

Keywords: reinforcement learning, model-free policy evaluation, TD learning, semiparametirc model, estimating function

1. Introduction

Studies in reinforcement learning (RL) have provided a methodology for optimal control and decision making in various practical applications, for example, job scheduling (Zhang and Dietterich, 1995), backgammon (Tesauro, 1995), elevator dispatching (Crites and Barto, 1996), and dynamic channel allocation (Singh and Bertsekas, 1997). Although the tasks in these studies are large-scale and complicated, RL has achieved good performance which exceeds that of human experts. These successes were attributed to model-free policy evaluation, that is, the value function which evaluates the expected cumulative reward is estimated from a given sample trajectory without specifying the task environment. Since the policy is updated based on the estimated value function, the quality

[*] Also at Berlin Institute of Technology, Computer Science, Machine Learning Group, Franklinstr. 28/29, 10587, Berlin, Germany.

of its estimation directly affects policy improvement. Hence, it is important for research in RL to develop efficient model-free policy evaluation techniques.

This article introduces a novel framework, *semiparametric statistical inference*, to model-free policy evaluation. This framework generalizes previously developed model-free algorithms, which include temporal difference learning and its extensions, and moreover, enables us to investigate the statistical properties of these algorithms, which have not been yet elucidated.

The overall framework can be summarized as follows. We focus on the policy evaluation like in previous studies (Singh and Dayan, 1998; Mannor et al., 2004; Grunëwälder and Obermayer, 2006; Mannor et al., 2007); then we deal with the Markov Reward Process (MRP), in which the initial, transition, and the reward probabilities are assumed to be unknown. From a sample trajectory given by MRP, the value function is estimated without directly identifying those probabilities. Central to our proposed framework is the notion of *semiparametric statistical models* which include not only parameters of interest but also additional nuisance parameters with possibly infinite degrees of freedom. We specify the MRP as a semiparametric model, where only the value function is modeled parametrically with a smaller number of parameters than necessary, while the other unspecified part of MRP corresponds to the nuisance parameters. For estimating the parameters of interest in such models, *estimating functions* provide a well-established toolbox: they give consistent estimators (M-estimators) regardless of the nuisance parameters (Godambe, 1960, 1991; Huber and Ronchetti, 2009; van der Vaart, 2000). In this sense, the semiparametric inference is a promising approach to model-free policy evaluation.

Our contributions are summarized as follows:

(a) A set of all estimating functions is shown explicitly: the set constitutes a general class of consistent estimators (Theorem 4). Furthermore, by applying the asymptotic analysis, we derive the asymptotic estimation variance of general estimating functions (Lemma 3) and the optimal estimating function that yields the *minimum asymptotic variance* of estimation (Theorem 6).

(b) We discuss two types of learning algorithms based on estimating functions. One is the class of batch algorithms which obtain estimators in one shot by using all samples in the given trajectory such as least squares temporal difference (LSTD) learning (Bradtke and Barto, 1996). The other is the class of online algorithms which update the estimators step-by-step such as temporal difference (TD) learning (Sutton, 1988). In the batch algorithm, we assume that the value function is represented as a parametrically linear function and derive a new least squares-type algorithm, *gLSTD* learning, which achieves the minimum asymptotic variance (Algorithm 1).

(c) Following previous work (Amari, 1998; Murata and Amari, 1999; Bottou and LeCun, 2004, 2005), we examine the convergence of statistical deviations of the online algorithms. We then show that the online algorithms can achieve the same asymptotic performance as their batch counterparts if the parameters controlling learning processes are appropriately tuned (Lemma 9 and Theorem 10). We derive the optimal choice of the estimating function and construct the online learning algorithm that achieves the minimum estimation error asymptotically (Algorithm 2). We also propose an acceleration of TD learning, which is called *accelerated TD learning* (Algorithm 3).

(d) We then show that our proposed framework generalizes almost all of the conventional model-free policy evaluation algorithms, such as TD learning, TD(λ) learning (Sutton, 1988; Sutton

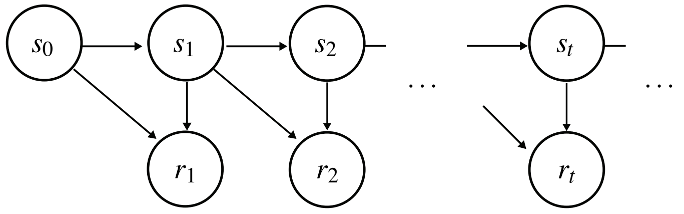


Figure 1: Graphical model for infinite horizon MRP. s and r denote state variable and reward, respectively.

and Barto, 1998), Bellman residual (RG) learning (Baird, 1995), LSTD learning (Bradtke and Barto, 1996), LSTD(λ) learning (Boyan, 2002), least squares policy evaluation (LSPE) learning (Nedić and Bertsekas, 2003), and incremental LSTD (iLSTD) learning (Geramifard et al., 2006, 2007) (Table 1).

We compare the performance of the proposed online algorithms with a couple of well-established algorithms in simple numerical experiments and show that the results support our theoretical findings.

The rest of this article is organized as follows. First, we give background of MRP and define the semiparametric statistical model for estimating the value function (Section 2). After providing a short overview of estimating functions (Section 3), we present the main contribution, fundamental statistical analysis based on the estimating function theory (Section 4). Then, we explain the construction of practical learning algorithms, derived from estimating functions, as both batch and online algorithms (Section 5). Furthermore, relations of our proposed methods to current algorithms in RL are discussed (Section 6). Finally, we report our experimental results (Section 7), and discuss open questions and future direction of this study (Section 8).

2. Markov Reward Processes

Figure 1 shows a graphical model for an infinite horizon MRP[1] which is defined by the initial state probability $p(s_0)$, the state transition probability $p(s_t|s_{t-1})$ and the reward probability $p(r_t|s_t,s_{t-1})$. State variable s is an element of a finite set S and reward variable $r \in R$ can be either discrete or continuous. The joint distribution of a sample trajectory $Z_T \equiv \{s_0, s_1, r_1 \cdots, s_T, r_T\}$ of the MRP is described as

$$p(Z_T) = p(s_0)\prod_{t=1}^{T} p(r_t|s_t,s_{t-1})p(s_t|s_{t-1}). \tag{1}$$

We further impose the following assumptions on MRPs.

Assumption 1 *Under $p(s_t|s_{t-1})$, the MRP has a unique invariant stationary distribution $\mu(s)$.*

Assumption 2 *For any time t, reward r_t is uniformly bounded.*

1. In this study, we only consider MRPs; however, extension to Markov Decision Processes (MDPs) is straightforward as long as considering the policy evaluation problem (hence the policy is fixed).

Before introducing the statistical framework, we begin by confirming that the value function estimation can be interpreted as estimation of certain statistics of MRP (1).

Proposition 1 *(Bertsekas and Tsitsiklis, 1996) Consider a conditional probability of* $\{r_t, s_t\}$ *given* s_{t-1},

$$p(r_t, s_t|s_{t-1}) = p(r_t|s_t, s_{t-1})p(s_t|s_{t-1}).$$

Then, there is such a function V *that*

$$\mathbb{E}\left[r_t|s_{t-1}\right] = V(s_{t-1}) - \gamma\mathbb{E}[V(s_t)|s_{t-1}] \tag{2}$$

holds for any state $s_{t-1} \in S$*, where* $\gamma \in [0,1)$ *is a constant called discount factor. Here,* $\mathbb{E}\left[\cdot|s\right]$ *denotes the conditional expectation for the given state* s*. The function* V *that satisfies Equation (2) is unique and found to be the value function:*

$$V(s) \equiv \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t \,\middle|\, s_0 = s\right]. \tag{3}$$

We assume throughout this article that the value function can be represented by a certain parametric function, even a nonlinear function with respect to its parameter.

Assumption 3 *The value function given by Equation (3) is represented by a parametric function* $g(s, \boldsymbol{\theta})$*:*

$$V(s) = g(s, \boldsymbol{\theta}).$$

Here, $g : S \times \Theta \mapsto \mathbb{R}$ *and* $\boldsymbol{\theta} \in \Theta$ *is a certain parameter in a parameter space* $\Theta \subseteq \mathbb{R}^m$*. Also, the dimension of the parameter* $\boldsymbol{\theta}$ *is smaller than that of the state space:* $m < |S|$*. Moreover,* $g(s, \boldsymbol{\theta})$ *is assumed to be twice continuously differentiable with respect to* $\boldsymbol{\theta}$*.*

Under Assumption 3, $p(r_t|s_{t-1})$ is partially parametrized by $\boldsymbol{\theta}$, through its conditional mean

$$\mathbb{E}[r_t|s_{t-1}] = g(s_{t-1}, \boldsymbol{\theta}) - \gamma\mathbb{E}[g(s_t, \boldsymbol{\theta})|s_{t-1}]. \tag{4}$$

Our objective is to find out such a value of the parameter $\boldsymbol{\theta}$ that function $g(s, \boldsymbol{\theta})$ satisfies Equation (4), that is, it coincides with the true value function.

To specify the probabilistic model (1) altogether, we usually need extra parameters other than $\boldsymbol{\theta}$. Let $\boldsymbol{\xi}_0$ and $\boldsymbol{\xi}_s$ be such additional parameters that $p(s_0, \boldsymbol{\xi}_0)$ and $p(r, s|s; \boldsymbol{\theta}, \boldsymbol{\xi}_s)$ can completely represent the initial and transition distributions, respectively. In such a case, the joint distribution of the trajectory Z_T is expressed as

$$p(Z_T; \boldsymbol{\theta}, \boldsymbol{\xi}) = p(s_0; \boldsymbol{\xi}_0) \prod_{t=1}^{T} p(r_t, s_t|s_{t-1}; \boldsymbol{\theta}, \boldsymbol{\xi}_s), \tag{5}$$

where $\boldsymbol{\xi} \equiv (\boldsymbol{\xi}_0, \boldsymbol{\xi}_s)$.

Since it is in general quite difficult to know the complexity of the target system, we attempt to estimate the parameter $\boldsymbol{\theta}$ representing the value function beside the presence of the extra $\boldsymbol{\xi}$ which may have innumerable degrees of freedom. Statistical models which contain such (possibly infinite-dimensional) nuisance parameters ($\boldsymbol{\xi}$) in addition to the parameter of interest ($\boldsymbol{\theta}$), are called semi-parametric (Bickel et al., 1998; Amari and Kawanabe, 1997; van der Vaart, 2000). We emphasize that the nuisance parameters are necessary only for developing theoretical frameworks. In actual estimation procedures of the parameter $\boldsymbol{\theta}$, same as in other model-free policy evaluation algorithms, we neither define them concretely, nor estimate them. This can be achieved by using estimating functions which is a well-established technique to obtain a consistent estimator of the parameter without estimating the nuisance parameters (Godambe, 1960, 1991; Amari and Kawanabe, 1997; Huber and Ronchetti, 2009). The advantages of considering such semiparametric models behind the model-free policy evaluation are:

(a) we can characterize all possible model-free algorithms,

(b) we can discuss asymptotic properties of the estimators in a unified way and obtain the optimal one with the *minimum estimation error*.

We review the estimating function method in the next section.

3. Estimating Functions in Semiparametric Models

We begin with a short overview of the estimating function theory in the independent and identically distributed (i.i.d.) case and then discuss the MRP case in the next section. We consider a general semiparametric model $p(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\xi})$, where $\boldsymbol{\theta}$ is an m-dimensional parameter of interest and $\boldsymbol{\xi}$ is a nuisance parameter which can have infinite degrees of freedom. An m-dimensional vector function $\boldsymbol{f}$ of $\boldsymbol{x}$ and $\boldsymbol{\theta}$ is called an *estimating function* (Godambe, 1960, 1991) when it satisfies the following conditions for any $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$ for sufficiently large values of T;

$$\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}[\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})] = \mathbf{0}, \tag{6}$$

$$\det|\mathbf{A}| \neq 0, \ \text{ where } \mathbf{A} = \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[\partial_{\boldsymbol{\theta}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})\right], \tag{7}$$

$$\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[||\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})||^2\right] < \infty, \tag{8}$$

where $\partial_{\boldsymbol{\theta}} = \partial/\partial\boldsymbol{\theta}$ is the partial derivative with respect to $\boldsymbol{\theta}$, and $\det|\cdot|$ and $||\cdot||$ denote the determinant and the Euclidean norm, respectively. Here $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}[\cdot]$ means the expectation over $\boldsymbol{x}$ with respect to the distribution $p(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\xi})$ and we further remark that the parameter $\boldsymbol{\theta}$ in $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})$ and $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}[\cdot]$ must be the same.

Suppose i.i.d. samples $\{\boldsymbol{x}_1,\cdots,\boldsymbol{x}_T\}$ are generated from the model $p(\boldsymbol{x};\boldsymbol{\theta}^*,\boldsymbol{\xi}^*)$. If there is an estimating function $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})$, we can obtain an estimator $\hat{\boldsymbol{\theta}}_T$ which has good asymptotic properties, by solving the following estimating equation:

$$\sum_{t=1}^{T} \boldsymbol{f}(\boldsymbol{x}_t,\hat{\boldsymbol{\theta}}_T) = \mathbf{0}. \tag{9}$$

A solution of the estimating Equation (9) is called an *M-estimator* in statistics (Huber and Ronchetti, 2009; van der Vaart, 2000). The M-estimator is consistent, that is, it converges to the true value *regardless of the nuisance parameter* $\boldsymbol{\xi}^*$.[2] Moreover, it is normally distributed, that is,

2. In this study, 'consistency' means 'local consistency' as well as in the previous works (Amari and Kawanabe, 1997; Amari and Cardoso, 2002; Kawanabe and Müller, 2005).

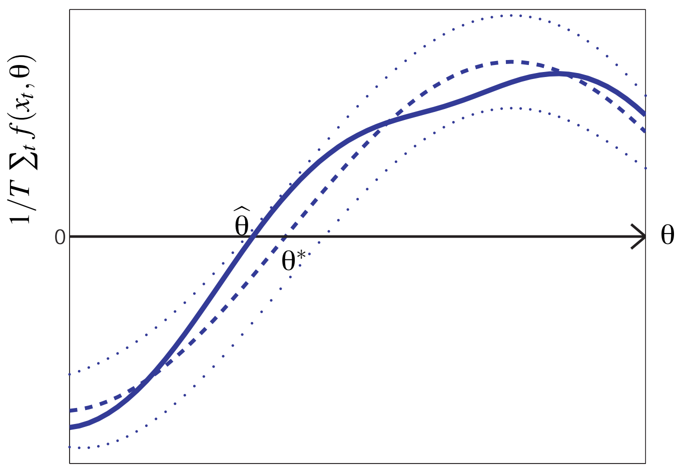


Figure 2: An illustrative plot of $1/T\sum_t f(x_t,\theta)$ as function of θ (solid line). Due to the effect of finite samples, the function is slightly apart from its expectation $\mathbb{E}_{\theta^*,\xi^*}[f(x,\theta)]$ (dashed line) which takes 0 at $\theta=\theta^*$ because of condition (6). Condition (7) means that the expectation (dashed line) has a non-zero slope around θ^*, which ensures the local uniqueness of the zero crossing point. On the other hand, condition (8) guarantees that its standard deviation, shown by the two dotted lines, shrinks in the order of $1/\sqrt{T}$, thus we can expect to find asymptotically at least one solution $\hat{\theta}_T$ of estimating Equation (9) near the true value θ^*. This situation holds regardless of the value of the true nuisance parameter ξ^*.

$\hat{\boldsymbol{\theta}}_T \sim \mathcal{N}(\boldsymbol{\theta}^*, \mathrm{Av})$, when the sample size T approaches infinity. The matrix Av, which is called the asymptotic variance, can be calculated by

$$\mathrm{Av} \equiv \mathrm{Av}(\hat{\boldsymbol{\theta}}_T) = \frac{1}{T}\mathbf{A}^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta}^*)\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta}^*)^\top\right](\mathbf{A}^\top)^{-1},$$

where $\mathbf{A}=\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\partial_{\boldsymbol{\theta}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta}^*)]$, and the symbol $\top$ denotes the matrix transpose. Note that the matrix Av depends on $(\boldsymbol{\theta}^*,\boldsymbol{\xi}^*)$, but not on the samples $\{\boldsymbol{x}_1,\cdots,\boldsymbol{x}_T\}$. We illustrate in Figure 2 the left side of the estimating Equation (9) normalized by the sample size T to explain why an M-estimator has good properties and to show the meaning of conditions (6)-(8).

4. Estimating Functions in MRP Model

The notion of estimating functions has been extended to be applicable to Markov time-series (Godambe, 1985, 1991; Wefelmeyer, 1996; Sørensen, 1999). We need a similar extension to enable it to be applied to MRPs. For convenience, we write the triplet at time t as $z_t \equiv \{s_{t-1}, s_t, r_t\} \in S^2 \times R$ and the trajectory up to time t as $Z_t \equiv \{s_0, s_1, r_1, \ldots, s_t, r_t\} \in S^{t+1}\times R^t$.

Let us consider an m-dimensional vector-valued function of the form $\boldsymbol{f}_T : S^{T+1}\times R^T \times \Theta \mapsto \mathbb{R}^m$:

$$\boldsymbol{f}_T(Z_T,\boldsymbol{\theta}) = \sum_{t=1}^{T}\psi_t(Z_t,\boldsymbol{\theta}).$$

This is similar to the left side of (9) in the i.i.d. case, but now each term $\psi_t : S^{t+1} \times R^t \times \Theta \mapsto \mathbb{R}^m$ depends also on previous observations, that is, a function of the sequence up to time t. If the sequence of the functions $\{\psi_t\}$ satisfies the following properties for any $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$, the function $\boldsymbol{f}_T$ becomes an estimating function for T sufficiently large (Godambe, 1985, 1991).

$$\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\psi_t(Z_t,\boldsymbol{\theta})|Z_{t-1}] = \mathbf{0}, \ \forall t, \tag{10}$$

$$\det|\mathbf{A}| \neq 0, \ \text{ where } \ \mathbf{A} \equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}[\partial_{\boldsymbol{\theta}}\psi_t(Z_t,\boldsymbol{\theta})], \tag{11}$$

$$\lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[\|\psi_t(Z_t,\boldsymbol{\theta})\|^2\right] < \infty. \tag{12}$$

Note that the estimating function $\boldsymbol{f}_T(Z_T,\boldsymbol{\theta})$ satisfies the martingale properties because of condition (10). Therefore, it is called a *martingale estimating function* in the literature (Godambe, 1985, 1991; Wefelmeyer, 1996; Sørensen, 1999).[3] Although time-series estimating functions can be defined in a more general form, the above definition is sufficient for our theoretical consideration.

4.1 Characterizing Class of Estimating Functions

In this section, we characterize possible estimating functions in MRPs. Let $\varepsilon : S^2 \times R \times \Theta \mapsto \mathbb{R}^1$ be the so-called temporal difference (TD) error, that is,

$$\varepsilon_t \equiv \varepsilon(z_t,\boldsymbol{\theta}) \equiv g(s_{t-1},\boldsymbol{\theta}) - \gamma g(s_t,\boldsymbol{\theta}) - r_t.$$

From Equation (4), its conditional expectation $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\varepsilon_t|Z_{t-1}] = \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\varepsilon_t|s_{t-1}]$ is equal to 0 for any t. Furthermore, this zero-mean property holds even when multiplied by any weight function $\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})$, which depends on past observations and the parameter, that is,

$$\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\varepsilon(z_t,\boldsymbol{\theta})|Z_{t-1}] = \boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\varepsilon(z_t,\boldsymbol{\theta})|Z_{t-1}] = \mathbf{0}, \tag{13}$$

for any t. We can obtain a class of possible estimating functions $\boldsymbol{f}_T(Z_T,\boldsymbol{\theta})$ in MRPs from this observation if we impose some regularity conditions summarized in Assumption 4.

Assumption 4

(a) Function $\boldsymbol{w}_t : S^{t+1} \times R^t \times \Theta \mapsto \mathbb{R}^m$ can be twice continuously differentiable with respect to parameter $\boldsymbol{\theta}$ for any t, and $\lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}[|\partial_{\boldsymbol{\theta}}\boldsymbol{w}_t(Z_t,\boldsymbol{\theta})|] < \infty$ for any $\boldsymbol{\theta}$.

(b) There exists a limit of matrix $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta})\}^\top]$, and the matrix $\lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta})\}^\top]$ is nonsingular for any $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$.

(c) $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}[\|\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\varepsilon(z_t,\boldsymbol{\theta})\|^2]$ is finite for any t, $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$.

3. Strictly speaking, strict consistency of M-estimator given by function $\boldsymbol{f}(Z_T,\boldsymbol{\theta})$ requires some additional conditions. To show consistency rigorously, we have to impose further conditions for exchange between limit and expectation operators in the neighborhood of the true parameter (more detailed discussion is shown in Theorem 3.6 in Sørensen 1999). In this article, for the sake of readability, we do not show such strict discussion.

Lemma 2 *Suppose that random sequence Z_T is generated from a distribution of semiparametric model $\{p(Z_T;\boldsymbol{\theta},\boldsymbol{\xi})\,|\,\boldsymbol{\theta},\boldsymbol{\xi}\}$ defined by Equation (5). If the conditions in Assumptions 1-4 are satisfied, then*

$$\boldsymbol{f}_T(Z_T,\boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}) \equiv \sum_{t=1}^{T} \boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\varepsilon(z_t,\boldsymbol{\theta}) \tag{14}$$

becomes an estimating function.

The proof is given in Appendix C. From Lemma 2, we can obtain an M-estimator $\hat{\boldsymbol{\theta}}_T : S^{T+1}\times R^T \mapsto \mathbb{R}^m$ by solving the estimating equation

$$\sum_{t=1}^{T} \boldsymbol{\psi}_t(Z_t,\hat{\boldsymbol{\theta}}_T) = \mathbf{0}. \tag{15}$$

Practical procedures for finding the solution of the estimating Equation (15) will be discussed in Section 5. The estimator derived from the estimating Equation (15) has an asymptotic variance summarized in the following lemma.

Lemma 3 *Suppose that random sequence Z_T is generated from distribution $p(Z_T;\boldsymbol{\theta}^*,\boldsymbol{\xi}^*)$. If the conditions in Assumptions 1-4 are satisfied, then the M-estimator derived from Equation (15) has asymptotic estimation variance*

$$\mathrm{Av} = \mathrm{Av}(\hat{\boldsymbol{\theta}}_T) = \frac{1}{T}\mathbf{A}^{-1}\boldsymbol{\Sigma}\left(\mathbf{A}^\top\right)^{-1}, \tag{16}$$

where $\mathbf{A} = \mathbf{A}(\boldsymbol{\theta}^*,\boldsymbol{\xi}^*) = \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\left\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\right\}^\top\right]$,
$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\boldsymbol{\theta}^*,\boldsymbol{\xi}^*) = \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\varepsilon(z_t,\boldsymbol{\theta}^*)^2\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)^\top\right]$.

The proof is given in Appendix D. Interestingly, for the MRP model, we can specify all possible estimating functions. More specifically, the converse of Lemma 2 also holds; any martingale estimating functions for MRP must take the form (14).

Theorem 4 *Suppose that the conditions in Assumptions 1-4 are satisfied. Then, any martingale estimating function $\boldsymbol{f}_T(Z_T,\boldsymbol{\theta}) = \sum_{t=1}^{T}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta})$ in the semiparametric model $\{p(Z_T;\boldsymbol{\theta},\boldsymbol{\xi})\,|\,\boldsymbol{\theta},\boldsymbol{\xi}\}$ of MRP can be expressed as*

$$\boldsymbol{f}_T(Z_T,\boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\varepsilon(z_t,\boldsymbol{\theta}). \tag{17}$$

The proof is given in Appendix E.

4.2 Optimal Estimating Function

Since Theorem 4 has specified the set of all martingale estimating functions, we can now discuss the optimal estimating function among them which gives an M-estimator with the *minimum asymptotic variance*. The weight function $\boldsymbol{w}_t(Z_{t-1},\boldsymbol{\theta})$ may depend not only on the current state s_t and the parameter $\boldsymbol{\theta}$, but also on the previous states and rewards. However, we do not need to consider such weight functions, as Lemma 5 shows.

Lemma 5 *Let $\boldsymbol{w}_t(Z_t, \boldsymbol{\theta})$ be any weight function that depends on the current and previous observations and the parameter, and satisfies the conditions in Assumption 4. Then, there is necessarily a weight function depending only on the current state and the parameter whose corresponding estimator has the minimum asymptotic variance among all possible weight functions.*

The proof is given in Appendix F.

We next discuss the optimal weight function of Equation (14) in terms of asymptotic variance, which corresponds to the optimal estimating function.

Theorem 6 *Suppose that random sequence Z_T is generated from distribution $p(Z_T; \boldsymbol{\theta}^*, \boldsymbol{\xi}^*)$. If the conditions in Assumptions 1-4 are satisfied, an optimal estimating function with minimum asymptotic estimation variance is given by*

$$\boldsymbol{f}_T^*(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\psi}_t^*(z_t, \boldsymbol{\theta}) \equiv \sum_{t=1}^{T} \boldsymbol{w}_{t-1}^*(s_{t-1}, \boldsymbol{\theta}^*)\varepsilon(z_t, \boldsymbol{\theta}), \tag{18}$$

where

$$\boldsymbol{w}_{t-1}^*(s_{t-1}, \boldsymbol{\theta}^*) \equiv \mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}_s^*}[\varepsilon(z_t, \boldsymbol{\theta}^*)^2 | s_{t-1}]^{-1} \mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}} \varepsilon(z_t, \boldsymbol{\theta}^*) | s_{t-1}].$$

The proof is given in Appendix G. Note that weight function $\boldsymbol{w}_{t-1}^*(Z_{t-1}, \boldsymbol{\theta}^*)$ depends on true parameter $\boldsymbol{\theta}^*$ (unknown) and requires the expectation with respect to $p(r_t, s_t | s_{t-1}; \boldsymbol{\theta}^*, \boldsymbol{\xi}_s^*)$, which is also unknown. Therefore, we need to approximate the true parameter and the expectation, which will be explained in a later section.

The asymptotic variance of the optimal estimating function can be calculated from Lemma 3 and Theorem 6.

Lemma 7 *The minimum asymptotic variance is given by*

$$\mathrm{Av} = \mathrm{Av}(\hat{\boldsymbol{\theta}}_T) = \frac{1}{T}\mathbf{Q}^{-1},$$

where $\mathbf{Q} \equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}} \boldsymbol{\psi}_t^*(z_t, \boldsymbol{\theta}^*)\right] = \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}^*}\left[\boldsymbol{\psi}_t^*(z_t, \boldsymbol{\theta}^*) \boldsymbol{\psi}_t^*(z_t, \boldsymbol{\theta}^*)^\top\right].$

The proof is given in Appendix H. We here note that positive definite matrix $\mathbf{Q}$ is similar to the *Fisher information matrix*, which is well-known in asymptotic estimation theory. However, the information associated with this matrix $\mathbf{Q}$ is generally smaller than the Fisher information because we sacrifice statistical efficiency for robustness against the nuisance parameter (Amari and Kawanabe, 1997; Amari and Cardoso, 2002). In other words, the estimator derived from the estimating function (18) does not achieve the statistical lower bound, that is, the Cramèr-Rao lower bound.[4]

5. Learning Algorithms

In this section, we present two kinds of practical algorithms to obtain the solution of the estimating Equation (15): one is the batch learning procedure and the other is the online learning procedure. In Section 5.1, we discuss batch learning and derive new least squares-type algorithms like LSTD and

4. If one wants more efficient estimators, it is necessary to identify the target MRP, including the nuisance parameters.

LSTD(λ) to determine the parameter $\boldsymbol{\theta}$ under the assumption that the value function is represented as a parametrically linear function. In Section 5.2, we then study convergence issues of online learning. We first analyze the sufficient condition of the convergence of the estimation and the convergence rate of various online procedures without the constraint of linear parametrization. This theoretical consideration allows us to obtain a new online learning algorithm that asymptotically converges faster than current online algorithms.

5.1 Batch Learning

Let $g(s,\boldsymbol{\theta})$ be a linear parametric function of features:

$$V(s_t) \equiv \boldsymbol{\phi}(s_t)^\top \boldsymbol{\theta}, \tag{19}$$

where $\boldsymbol{\phi} : S \mapsto \mathbb{R}^m$ is a feature vector and $\boldsymbol{\theta} \in \Theta$ is a parameter vector. Then, estimating Equation (14) is given as

$$\sum_{t=1}^{T} \boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}) \left\{ (\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t))^\top \hat{\boldsymbol{\theta}}_T - r_t \right\} = \mathbf{0}.$$

If the weight function does not depend on parameter $\boldsymbol{\theta}$, the estimator $\hat{\boldsymbol{\theta}}_T$ can be analytically obtained as

$$\hat{\boldsymbol{\theta}}_T = \left\{ \sum_{t=1}^{T} \boldsymbol{w}_{t-1}(Z_{t-1})(\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t))^\top \right\}^{-1} \left\{ \sum_{t=1}^{T} \boldsymbol{w}_{t-1}(Z_{t-1}) r_t \right\},$$

where $\boldsymbol{w}_t : S^{t+1} \times R^t \mapsto \mathbb{R}^m$ is a function which depends only on the previous observations. Note that when the weight function $\boldsymbol{w}(Z_t)$ is set to $\boldsymbol{\phi}(s_t)$, this estimator is equivalent to that of the LSTD learning.

We now derive a new least-squares learning algorithm, *generalized least squares temporal difference (gLSTD)*, which achieves minimum estimation of asymptotic variance in linear estimations of value functions. If weight function $\boldsymbol{w}_t^*(Z_t,\boldsymbol{\theta}^*)$ defined in Theorem 6 is known, an estimator of the estimating function (18) can be obtained as

$$\hat{\boldsymbol{\theta}}_T = \left\{ \sum_{t=1}^{T} \boldsymbol{w}_{t-1}^*(Z_{t-1},\boldsymbol{\theta}^*)(\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t))^\top \right\}^{-1} \left\{ \sum_{t=1}^{T} \boldsymbol{w}_{t-1}^*(Z_{t-1},\boldsymbol{\theta}^*) r_t \right\},$$

by recalling that $\boldsymbol{w}_{t-1}^*(Z_{t-1},\boldsymbol{\theta}^*) = \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}]^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t)|s_{t-1}]$. Obviously, we do not know $\boldsymbol{w}_{t-1}^*(Z_{t-1},\boldsymbol{\theta}^*)$ because the definition of $\boldsymbol{w}_{t-1}^*(Z_{t-1},\boldsymbol{\theta}^*)$ contains the residual at the true parameter, $\varepsilon(z_t,\boldsymbol{\theta}^*)$, and unknown conditional expectations, $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}]$ and $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t)|s_{t-1}]$. Therefore, we replace the true residual $\varepsilon(z_t,\boldsymbol{\theta}^*)$ with that of the LSTD estimator and approximate the expectations $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}]^{-1}$ and $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t)|s_{t-1}]$ by using function approximations

$$\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}]^{-1} \approx v(s_{t-1},\boldsymbol{\alpha}),$$
$$\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t)|s_{t-1}] \approx \boldsymbol{\zeta}(s_{t-1},\boldsymbol{\beta}),$$

Algorithm 1 gLSTD learning

```
for $t = 1, 2, \cdots$ do
    Obtain sample $z_t = \{s_{t-1}, s_t, r_t\}$
end for

Set constant $k$ to a sufficiently large value
for $t = 1, 2, \cdots$ do
    Calculate LSTD estimator $\hat{\boldsymbol{\theta}}^{\mathrm{LSTD}}$ based on sample
    $\{z_1, \cdots, z_{t-1}\} \cup \{z_{t+k}, \cdots, z_T\}$
    Calculate its residual $\hat{\varepsilon}_t$
        $\hat{\varepsilon}_t \leftarrow (\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t))^\top \hat{\boldsymbol{\theta}}^{\mathrm{LSTD}} - r_t$
    Calculate conditional expectations $v(s_{t-1}, \boldsymbol{\alpha}), \boldsymbol{\zeta}(s_{t-1}, \boldsymbol{\beta})$ by means of function approximations
    based on sample $\{z_1, \cdots, z_{t-1}\} \cup \{z_{t+k}, \cdots, z_T\}$
    Obtain the weight function
        $\hat{\boldsymbol{w}}^*_{t-1} \leftarrow v(s_{t-1}, \boldsymbol{\alpha})^{-1} \boldsymbol{\zeta}(s_{t-1}, \boldsymbol{\beta})$
end for

Obtain the gLSTD estimator
    $\hat{\boldsymbol{\theta}}^{\mathrm{gLSTD}}_T \leftarrow [\sum_{t=1}^T \hat{\boldsymbol{w}}^*_{t-1} (\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t))^\top]^{-1} [\sum_{t=1}^T \hat{\boldsymbol{w}}^*_{t-1} r_t]$
```

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are adjustable parameters for function approximators $v(s_{t-1}, \boldsymbol{\alpha})$ and $\boldsymbol{\zeta}(s_{t-1}, \boldsymbol{\beta})$, respectively. The estimation of conditional expectations is a simpler problem than that of the conditional probability itself. Also note that if the weight function is approximated by using past observations Z_{t-1}, condition (13) still holds regardless of the approximation accuracy of the weight function, implying the consistency of gLSTD. This is because any function that only depends on past observations can be employed as a weight function. This favorable characteristic is consistent regardless of the accuracy of approximation and allows us to use any approximation techniques (e.g., sparse regression, kernel regression, or neural networks) without particular constraints. Algorithm 1 demonstrates the pseudo-code of gLSTD learning. We introduce constant non-negative integer k to Algorithm 1 to enhance the efficient use of samples. LSTD estimator $\hat{\boldsymbol{\theta}}^{\mathrm{LSTD}}$ can be obtained in an unbiased manner by using future trajectory $\{z_{t+k}, \cdots, z_T\}$ for sufficiently large positive integer k, because the MRPs defined in Equation (1) satisfy geometrically uniform mixing, implying the exponential decay of the correlation between the statistics of s_t and s_{t+k}. Although k must be infinite to guarantee consistency in a strict sense, it could be a certain moderate integer when we consider the trade-off between the accuracy of function approximations and consistency. There are also some computational difficulties in Algorithm 1 with a large k value, because we must store the sample trajectory in memory to estimate weight function $\hat{\boldsymbol{w}}^*_t$ at each time t. Thus, in the simulations in Sections 7 and 8, we set k to zero; both the LSTD estimator and conditional expectations are calculated by using whole samples. Although this simplified implementation in fact violates the condition of consistency, it works well in practice.

5.2 Online Learning

Online learning procedures in the field of RL are often preferred to batch learning ones because they require less memory and can be adapted even to time-variant situations. Here, an online estimator of $\boldsymbol{\theta}$ at time t is denoted as $\hat{\boldsymbol{\theta}}_t$. Suppose that sequence $\{\boldsymbol{\psi}_1(Z_1,\boldsymbol{\theta}),\cdots,\boldsymbol{\psi}_T(Z_T,\boldsymbol{\theta})\}$ forms a martingale estimating function for MRP. Then, an online update rule can simply be given by

$$\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} - \eta_t \boldsymbol{\psi}_t(Z_t,\hat{\boldsymbol{\theta}}_{t-1}), \tag{20}$$

where η_t denotes a nonnegative scalar stepsize. In fact, there are other online update rules derived from the same estimating function $\boldsymbol{f}_t(Z_t,\boldsymbol{\theta}) = \sum_{i=1}^t \boldsymbol{\psi}_i(Z_i,\boldsymbol{\theta})$ as

$$\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} - \eta_t \mathbf{R}(\hat{\boldsymbol{\theta}}_{t-1})\boldsymbol{\psi}_t(Z_t,\hat{\boldsymbol{\theta}}_{t-1}), \tag{21}$$

where $\mathbf{R}(\boldsymbol{\theta})$ denotes an $m\times m$ nonsingular matrix only depending on $\boldsymbol{\theta}$ (Amari, 1998). This variation results from the fact that function $\mathbf{R}(\boldsymbol{\theta})\sum_{i=1}^t \boldsymbol{\psi}_i(Z_i,\boldsymbol{\theta})$ yields the same roots as its original for any $\mathbf{R}(\boldsymbol{\theta})$. This equivalence guarantees that both learning procedures, (20) and (21), have the same equilibrium, while their dynamics may be different, that is, even if the original algorithm (20) is unstable around the required solution, it can be stabilized by introducing appropriate $\mathbf{R}(\boldsymbol{\theta})$ into (21).

We will discuss the convergence of the online learning algorithm (21) in the next two subsections.

5.2.1 Convergence to True Value

We will now discuss the convergence of online learning (21) to the true parameter $\boldsymbol{\theta}^*$. For the sake of simplicity, we will focus on local convergence, that is, initial estimator $\hat{\boldsymbol{\theta}}_0$ is confined in the neighborhood of the true parameter, which is assumed to be a unique solution in the neighborhood. Now let us introduce sufficient conditions for convergence.

Assumption 5

(a) For any t, $(\hat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^*)^\top \mathbf{R}(\hat{\boldsymbol{\theta}}_t)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)|s_t]$ is nonnegative.

(b) For any t, there exists such nonnegative constants c_1 and c_2 that

$$\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\left\|\mathbf{R}(\hat{\boldsymbol{\theta}}_t)\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)\right\|^2 \Big| s_t\right] \le c_1 + c_2\left\|\hat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^*\right\|^2.$$

Condition (a) assumes that the opposite of gradient $\mathbf{R}(\hat{\boldsymbol{\theta}}_t)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)|s_t]$ must point toward the true parameter $\boldsymbol{\theta}^*$ at each time t. Then, the following theorem guarantees the convergence of $\hat{\boldsymbol{\theta}}_t$ to $\boldsymbol{\theta}^*$.

Theorem 8 *Suppose that random sequence Z_T is generated from distribution $p(Z_T;\boldsymbol{\theta}^*,\boldsymbol{\xi}^*)$. Also, suppose that the conditions in Assumptions 1-5 hold. If stepsizes $\{\eta_t\}$ are all positive and satisfy $\sum_{t=1}^\infty \eta_t = \infty$ and $\sum_{t=1}^\infty \eta_t^2 < \infty$, then the online algorithm (21) almost surely converges to true parameter $\boldsymbol{\theta}^*$.*

The proof is given in Appendix I. Theorem 8 ensures that even if the original online learning algorithm (20) does not converge to the true parameter, we can construct an online learning algorithm with local consistency by appropriately choosing matrix $\mathbf{R}(\boldsymbol{\theta})$.

5.2.2 CONVERGENCE RATE

The convergence speed of an online algorithm could generally be slower than that of its batch counterpart that tries to solve the estimating equation using all available samples. However, if we set matrix $\mathbf{R}(\boldsymbol{\theta})$ and stepsizes $\{\eta_t\}$ appropriately, then it is possible to achieve the same convergence speed as that of the batch algorithm (Amari, 1998; Murata and Amari, 1999; Bottou and LeCun, 2004, 2005). Following the discussion on the previous work (Bottou and LeCun, 2004, 2005), we elucidate the convergence speed of online learning for estimating the value function in this section. Throughout the following discussion, the notion of *stochastic orders* plays a central role. Appendix A briefly describes the definition of stochastic orders and their properties. Then, we characterize the learning process for the batch algorithm.

Lemma 9 *Let $\tilde{\boldsymbol{\theta}}_t$ and $\tilde{\boldsymbol{\theta}}_{t-1}$ be solutions to estimating equations $(1/t)\sum_{i=1}^{t}\psi_i(Z_i,\tilde{\boldsymbol{\theta}}_t)=\mathbf{0}$ and $(1/(t-1))\sum_{i=1}^{t-1}\psi_i(Z_i,\tilde{\boldsymbol{\theta}}_{t-1})=\mathbf{0}$, respectively. We assume that the conditions in Assumptions 2-4 are satisfied. Also we assume that $\tilde{\boldsymbol{\theta}}_t$ is uniformly bounded for any t, and matrix $\tilde{\mathbf{R}}_t(\tilde{\boldsymbol{\theta}}_{t-1})\equiv(1/t)\sum_{i=1}^{t}\partial_{\boldsymbol{\theta}}\psi_i(Z_i,\tilde{\boldsymbol{\theta}}_{t-1})$ is nonsingular for any t. Then, we have*

$$\tilde{\boldsymbol{\theta}}_t=\tilde{\boldsymbol{\theta}}_{t-1}-\frac{1}{t}\tilde{\mathbf{R}}_t^{-1}(\tilde{\boldsymbol{\theta}}_{t-1})\psi_t(Z_t,\tilde{\boldsymbol{\theta}}_{t-1})+O_p\left(\frac{1}{t^2}\right), \tag{22}$$

where the definition of $O_p(\cdot)$ is given in Appendix A.

The proof is given in Appendix J. Note that Equation (22) defines the sequence of $\tilde{\boldsymbol{\theta}}_t$ as a recursive stochastic process that is essentially the same as online learning (21) for the same $\mathbf{R}$. In other words, Lemma 9 indicates that online algorithms can converge with the same convergence speed as their batch counterparts through an appropriate choice of matrix $\mathbf{R}$. Finally, the following theorem addresses the convergence speed of the (stochastic) learning process such as that in Equation (22).

Theorem 10 *Suppose that random sequence Z_T is generated from distribution $p(Z_T;\boldsymbol{\theta}^*,\boldsymbol{\xi}^*)$, and then consider the following learning process*

$$\hat{\boldsymbol{\theta}}_t=\hat{\boldsymbol{\theta}}_{t-1}-\frac{1}{t}\hat{\mathbf{R}}_t^{-1}\psi_t(Z_t,\hat{\boldsymbol{\theta}}_{t-1})+O_p\left(\frac{1}{t^2}\right), \tag{23}$$

where $\hat{\mathbf{R}}_t\equiv\{(1/t)\sum_{i=1}^{t}\partial_{\boldsymbol{\theta}}\psi_i(Z_i,\hat{\boldsymbol{\theta}}_{i-1})\}$. Assume that:

(a) For any t, $\hat{\boldsymbol{\theta}}_t$ is uniformly bounded.

(b) $\hat{\mathbf{R}}_t^{-1}$ can be written as $\hat{\mathbf{R}}_t^{-1}=\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\hat{\mathbf{R}}_t^{-1}|Z_{t-1}]+o_p(1/t)$.

(c) $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\partial_{\boldsymbol{\theta}}\psi_t(Z_t,\boldsymbol{\theta}^*)\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top]$ can be written as $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\partial_{\boldsymbol{\theta}}\psi_t(Z_t,\boldsymbol{\theta}^*)\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top]=\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\partial_{\boldsymbol{\theta}}\psi_t(Z_t,\boldsymbol{\theta}^*)]\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top]+o(1/t)$.

(d) For any t, $\hat{\mathbf{R}}_t$ is a nonsingular matrix.

Also assume that the conditions in Assumptions 1-4 are satisfied. If learning process (23) almost surely converges to the true parameter, then the convergence rate is given as

$$\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\|\hat{\boldsymbol{\theta}}_t-\boldsymbol{\theta}^*\|^2\right]=\frac{1}{t}\mathrm{tr}\left\{\mathbf{A}^{-1}\boldsymbol{\Sigma}(\mathbf{A}^{-1})^\top\right\}+o\left(\frac{1}{t}\right), \tag{24}$$

where $\mathbf{A} = \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]$ *and*
$\boldsymbol{\Sigma} = \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)^\top]$.

The proof is given in Appendix K. Note that this convergence rate (24) is neither affected by the third term of (23) nor by the $o_p(1/t)$ term in matrix $\hat{\mathbf{R}}_t^{-1}$.

5.2.3 GENERALIZED TD LEARNING

We now present the online learning procedure that yields the minimum estimation error. Roughly speaking, this is given by estimating function $\boldsymbol{f}_T^*(Z_T,\boldsymbol{\theta})$ in Theorem 6 with the best (i.e., with the fastest convergence) choice of the nonsingular matrix in Theorem 10:

$$\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} - \frac{1}{t}\hat{\mathbf{Q}}_t^{-1}\boldsymbol{\psi}^*(z_t,\hat{\boldsymbol{\theta}}_{t-1}), \tag{25}$$

where $\hat{\mathbf{Q}}_t^{-1} = \{(1/t)\sum_{i=1}^t \partial_{\boldsymbol{\theta}}\boldsymbol{\psi}^*(z_i,\hat{\boldsymbol{\theta}}_{i-1})\}^{-1}$ and $\boldsymbol{\psi}^*(z_t,\boldsymbol{\theta})$ have been defined by Equation (18). If learning equation (25) satisfies conditions in Assumptions 1-5 and Theorem 10, then it converges to the true parameter with the minimum estimation error, $(1/t)\mathbf{Q}^{-1}$. However, this is impractical as learning rule (25) contains unknown parameters and quantities. For practical implementation, we need to evaluate $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}]$ and $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]$ by using function approximations, whereas standard online learning procedures do not maintain statistics as a time series to avoid increasing the amount of memory. Therefore, we apply online functional approximations to these.

Let $v(s_t,\boldsymbol{\alpha}_t)$and $\boldsymbol{\zeta}(s_t,\boldsymbol{\beta}_t)$ be the approximations of $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon(z_{t+1},\boldsymbol{\theta}_t)^2|s_t]$ and $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_{t+1},\boldsymbol{\theta}_t)|s_t]$, respectively. Here, $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ are adjustable parameters, and they are adjusted in an online manner;

$$\hat{\boldsymbol{\alpha}}_t = \hat{\boldsymbol{\alpha}}_{t-1} - \eta_t^{\boldsymbol{\alpha}}\partial_{\boldsymbol{\alpha}}v(s_{t-1},\hat{\boldsymbol{\alpha}}_{t-1})\{v(s_{t-1},\hat{\boldsymbol{\alpha}}_{t-1}) - \varepsilon(z_t,\hat{\boldsymbol{\theta}}_{t-1})^2\}$$
$$\hat{\boldsymbol{\beta}}_t = \hat{\boldsymbol{\beta}}_{t-1} - \eta_t^{\boldsymbol{\beta}}\partial_{\boldsymbol{\beta}}\boldsymbol{\zeta}(s_{t-1},\hat{\boldsymbol{\beta}}_{t-1})\{\boldsymbol{\zeta}(s_{t-1},\hat{\boldsymbol{\beta}}_{t-1}) - \partial_{\boldsymbol{\theta}}\varepsilon(z_t,\hat{\boldsymbol{\theta}}_{t-1})\},$$

where $\eta_t^{\boldsymbol{\alpha}}$ and $\eta_t^{\boldsymbol{\beta}}$ are stepsizes. By using these parametrized functions, we can replace $\boldsymbol{\psi}_t^*(z_t,\hat{\boldsymbol{\theta}}_{t-1})$ and $\hat{\mathbf{Q}}_t^{-1}$ by

$$\boldsymbol{\psi}_t^*(z_t,\hat{\boldsymbol{\theta}}_{t-1}) = v(s_{t-1},\hat{\boldsymbol{\alpha}}_{t-1})^{-1}\boldsymbol{\zeta}(s_{t-1},\hat{\boldsymbol{\beta}}_{t-1})\varepsilon(z_t,\hat{\boldsymbol{\theta}}_{t-1})$$
$$\hat{\mathbf{Q}}_t^{-1} = \left\{\frac{1}{t}\sum_{i=1}^t v(s_{i-1},\hat{\boldsymbol{\alpha}}_{i-1})^{-1}\boldsymbol{\zeta}(s_{i-1},\hat{\boldsymbol{\beta}}_{i-1})\partial_{\boldsymbol{\theta}}\varepsilon(z_i,\hat{\boldsymbol{\theta}}_{i-1})^\top\right\}^{-1}. \tag{26}$$

Note that update (26) can be done in an online manner by applying the well-known matrix inversion lemma (Horn and Johnson, 1985);

$$\hat{\mathbf{Q}}_t^{-1} = \frac{1}{(1-\varepsilon_t)}\hat{\mathbf{Q}}_{t-1}^{-1} - \frac{\varepsilon_t}{1-\varepsilon_t}\frac{\hat{\mathbf{Q}}_{t-1}^{-1}\hat{\boldsymbol{w}}_{t-1}^*\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\hat{\boldsymbol{\theta}}_{t-1})^\top\hat{\mathbf{Q}}_{t-1}^{-1}}{(1-\varepsilon_t)+\varepsilon_t\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\hat{\boldsymbol{\theta}}_{t-1})^\top\hat{\mathbf{Q}}_{t-1}^{-1}\hat{\boldsymbol{w}}_{t-1}^*}, \tag{27}$$

where $\varepsilon_t \equiv 1/t$ and $\hat{\boldsymbol{w}}_{t-1}^* \equiv v(s_{t-1},\hat{\boldsymbol{\alpha}}_{t-1})^{-1}\boldsymbol{\zeta}(s_{t-1},\hat{\boldsymbol{\beta}}_{t-1})$. Following Amari et al. (2000), we additionally simplify update equation (27) as

$$\hat{\mathbf{Q}}_t^{-1} = (1+\varepsilon_t)\hat{\mathbf{Q}}_{t-1}^{-1} - \varepsilon_t\hat{\mathbf{Q}}_{t-1}^{-1}\hat{\boldsymbol{w}}_{t-1}^*\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\hat{\boldsymbol{\theta}}_{t-1})^\top\hat{\mathbf{Q}}_{t-1}^{-1}, \tag{28}$$

Algorithm 2 Optimal TD Learning

```
Initialize $\hat{\alpha}_0, \hat{\beta}_0, \hat{\theta}_0, \hat{\mathbf{Q}}_0^{-1} = \varepsilon \mathbf{I}_m, a_1, a_2$
{$\varepsilon$ and $\mathbf{I}_m$ denote a small constant and an $m \times m$ identical matrix.}

for $t = 1, 2, \cdots$ do
   Obtain a new sample, $z_t = \{s_{t-1}, s_t, r_t\}$

   Calculate the weight function, $\hat{\boldsymbol{w}}_{t-1}^*$
    $\hat{\alpha}_t \leftarrow \hat{\alpha}_{t-1} - \eta_t^{\alpha} \partial_{\alpha} v(s_{t-1}, \hat{\alpha}_{t-1}) \{ v(s_{t-1}, \hat{\alpha}_{t-1}) - \varepsilon(z_t, \hat{\theta}_{t-1})^2 \}$
    $\hat{\beta}_t \leftarrow \hat{\beta}_{t-1} - \eta_t^{\beta} \partial_{\beta} \zeta(s_{t-1}, \hat{\beta}_{t-1}) \{ \zeta(s_{t-1}, \hat{\beta}_{t-1}) - \partial_{\theta} \varepsilon(z_t, \hat{\theta}_{t-1}) \}$
    $\hat{\boldsymbol{w}}_{t-1}^* \leftarrow v(s_{t-1}, \hat{\alpha}_{t-1})^{-1} \zeta(s_{t-1}, \hat{\beta}_{t-1})$

   Update $\hat{\mathbf{Q}}_t^{-1}$ by using Equation (28)
    $\hat{\mathbf{Q}}_t^{-1} \leftarrow (1 + (1/t)) \hat{\mathbf{Q}}_{t-1}^{-1} - (1/t) \hat{\mathbf{Q}}_{t-1}^{-1} \hat{\boldsymbol{w}}_{t-1}^* \partial_{\theta} \varepsilon(z_t, \hat{\theta}_{t-1})^{\top} \hat{\mathbf{Q}}_{t-1}^{-1}$

   Update the parameter,
    $\tau \leftarrow \min(a_1, a_2/t)$
    $\hat{\theta}_t \leftarrow \hat{\theta}_{t-1} - (1/\tau) \hat{\mathbf{Q}}_t^{-1} \hat{\boldsymbol{w}}_{t-1}^* \varepsilon(z_t, \hat{\theta}_{t-1})$
end for
```

which can be obtained because ε_t is small. We call this procedure *optimal TD learning* and its pseudo-code is summarized in Algorithm 2.[5]

5.2.4 ACCELERATED TD LEARNING

TD learning is a traditional online approach to model-free policy evaluation and has been one of the most important algorithms in the RL field. Although TD learning is widely used because of its simplicity, it is known that it converges rather slowly. This section discusses TD learning from the viewpoint of the method of estimating functions and proposes a new online algorithm that can achieve faster convergence than standard TD learning.

To simplify the following discussion, we have assumed that $g(s, \boldsymbol{\theta})$ is a linear function as in Equation (19) with which we can solve the linear estimating equation using both batch and online procedures. When weight function $\boldsymbol{w}_t(Z_t, \boldsymbol{\theta})$ in Equation (13) is set to $\boldsymbol{\phi}(s_t)$, the online and batch procedures correspond to the TD and LSTD algorithms, respectively. Note that both TD and LSTD share the same estimating function. Therefore, from Lemma 9 and Theorem 10, we can theoretically construct accelerated TD learning, which converges at the same speed as LSTD learning.

Here, we consider the following learning equation:

$$\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} - \frac{1}{t} \hat{\mathbf{R}}_t^{-1} \boldsymbol{\phi}(s_{t-1}) \varepsilon(z_t, \hat{\boldsymbol{\theta}}_{t-1}), \tag{29}$$

where $\hat{\mathbf{R}}_t^{-1} = \{(1/t) \sum_{i=1}^t \boldsymbol{\phi}(s_{i-1})(\boldsymbol{\phi}(s_{i-1}) - \gamma \boldsymbol{\phi}(s_i))^{\top}\}^{-1}$. Since $\hat{\mathbf{R}}_t^{-1}$ converges to $\mathbf{A}^{-1} = \lim_{t \to \infty} \mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}^*}[\boldsymbol{\phi}(s_{t-1})(\boldsymbol{\phi}(s_{t-1}) - \gamma \boldsymbol{\phi}(s_t))^{\top}]^{-1}$ and $\mathbf{A}^{-1}$ must be a positive definite matrix (see Lemma 6.4 in Bertsekas and Tsitsiklis 1996), online algorithm (29) also almost surely converges to the true parameter. Then, if $\hat{\mathbf{R}}_t$ satisfies the conditions in Theorem 10, it can achieve the *same*

5. Since the online approximation of the weight function only depends on past observations, optimal TD learning is necessarily consistent even when the online approximation of the weight function is inaccurate.

Algorithm 3 Accelerated-TD Learning

```
Initialize $\hat{\theta}_0, \hat{\mathbf{R}}_0^{-1} = \varepsilon \mathbf{I}_m, a_1, a_2$
{$\varepsilon$ and $\mathbf{I}_m$ denote a small constant and an $m \times m$ identical matrix.}
for $t = 1, 2, \cdots$ do
  Obtain a new sample, $z_t = \{s_{t-1}, s_t, r_t\}$
  Update $\hat{\mathbf{R}}_t^{-1}$
   $\hat{\mathbf{R}}_t^{-1} \leftarrow (1+(1/t))\hat{\mathbf{R}}_{t-1}^{-1} - (1/t)\hat{\mathbf{R}}_{t-1}^{-1}\partial_{\boldsymbol{\theta}} g(s_{t-1}, \hat{\boldsymbol{\theta}}_{t-1})\partial_{\boldsymbol{\theta}}\varepsilon(z_t, \hat{\boldsymbol{\theta}}_{t-1})\hat{\mathbf{R}}_{t-1}^{-1}$
  Update the parameter,
   $\tau \leftarrow \min(a_1, a_2/t)$
   $\hat{\boldsymbol{\theta}}_t \leftarrow \hat{\boldsymbol{\theta}}_{t-1} - (1/\tau)\hat{\mathbf{R}}_t^{-1}\partial_{\boldsymbol{\theta}} g(s_{t-1}, \hat{\boldsymbol{\theta}}_{t-1})\varepsilon(z_t, \hat{\boldsymbol{\theta}}_{t-1})$
end for
```

convergence rate as LSTD. We call this procedure *Accelerated-TD learning*. We present an implementation of Accelerated-TD learning in Algorithm 3.

6. Related Work

This section discusses the relation between current major RL algorithms and the proposed ones from the viewpoint of estimating functions. Theorem 4 describes the broadest class of estimating functions that lead to unbiased estimators. Therefore, almost all the current value-based RL methods, in which consistency is assured, can be viewed as instances of the method of estimating functions.

For simplicity, let $g(s, \boldsymbol{\theta})$ be a linear function, that is, the value function can be represented as in Equation (19). We have two ways of solving such a linear estimating equation. The first is a batch procedure:

$$\hat{\boldsymbol{\theta}}_T = \left[\sum_{t=1}^{T} \boldsymbol{w}_{t-1}\left(\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t)\right)^\top\right]^{-1}\left[\sum_{t=1}^{T} \boldsymbol{w}_{t-1} r_t\right].$$

and the second is an online procedure:

$$\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} - \eta_t \hat{\mathbf{R}}_t \boldsymbol{w}_{t-1}\varepsilon(z_t, \hat{\boldsymbol{\theta}}_{t-1}),$$

where $\boldsymbol{w}_t$ is a weight function at time t. By choosing both weight function $\boldsymbol{w}_{t-1}$ and the learning procedure, we can derive various RL algorithms. Let $\boldsymbol{f}_T^{\mathrm{TD}}$, $\boldsymbol{f}_T^{\mathrm{TD}(\lambda)}$, $\boldsymbol{f}_T^{\mathrm{RG}}$and $\boldsymbol{f}_T^*$ be the estimating functions that are defined as

$$\boldsymbol{f}_T^{\mathrm{TD}} \equiv \boldsymbol{f}_T^{\mathrm{TD}}(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\phi}(s_{t-1})\varepsilon(z_t, \boldsymbol{\theta}),$$

$$\boldsymbol{f}_T^{\mathrm{TD}(\lambda)} \equiv \boldsymbol{f}_T^{\mathrm{TD}(\lambda)}(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^{T}\sum_{i=1}^{t} (\gamma\lambda)^{t-i}\boldsymbol{\phi}(s_{i-1})\varepsilon(z_t, \boldsymbol{\theta}),$$

$$\boldsymbol{f}_T^{\mathrm{RG}} \equiv \boldsymbol{f}_T^{\mathrm{RG}}(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}_s^*}[\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t)|s_{t-1}]\varepsilon(z_t, \boldsymbol{\theta}),$$

$$\boldsymbol{f}_T^* \equiv \boldsymbol{f}_T^*(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}_s^*}[(\varepsilon(z_t, \boldsymbol{\theta}^*)^2|s_{t-1}]^{-1}\mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}_s^*}[\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t)|s_{t-1}]\varepsilon(z_t, \boldsymbol{\theta}).$$

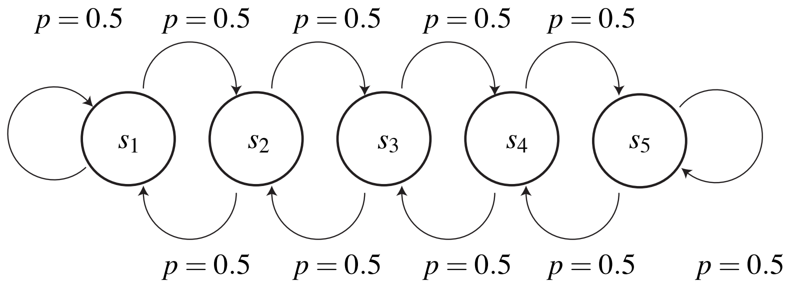


Figure 3: A five-states MRP.

Here, we remark that TD-based algorithms (TD Sutton and Barto, 1998, NTD Bradtke and Barto, 1996, LSTD Bradtke and Barto, 1996, LSPE Nedić and Bertsekas, 2003, GTD Sutton et al., 2009b, GTD2, TDC Sutton et al., 2009a and iLSTD Geramifard et al., 2006), TD (λ)-based algorithms (TD (λ) Sutton and Barto, 1998, NTD (λ) Bradtke and Barto, 1996, LSTD (λ) Boyan, 2002, LSPE (λ) Nedić and Bertsekas, 2003, and iLSTD (λ) Geramifard et al., 2007) and RG (Baird, 1995) originated from the estimating functions $\boldsymbol{f}_T^{\mathrm{TD}}$, $\boldsymbol{f}_T^{\mathrm{TD}(\lambda)}$ and $\boldsymbol{f}_T^{\mathrm{RG}}$, respectively. It should be noted that GTD, GTD2, GTDc, iLSTD, and iLSTD (λ) are specific online implementations for solving corresponding estimating equations; however, these algorithms can also be interpreted as instances of the method of estimating functions we propose. We have briefly summarized the relation between the current learning algorithms and the proposed algorithms in Table 1.

The asymptotic behavior of model-free policy evaluation has been analyzed within special contexts; Konda (2002) derived the asymptotic variance of LSTD (λ) and revealed that the convergence rate of TD (λ) was worse than that of LSTD (λ). Yu and Bertsekas (2006) derived the convergence rate of LSPE (λ) and found that it had the same convergence rate as LSTD (λ). Because these results can be seen in Lemma 3 and Theorem 8, our proposed framework generalizes previous asymptotic analyses to provide us with a methodology that can be more widely applied to carry out asymptotic analyses.

7. Simulation Experiment

In order to validate our theoretical developments, we compared the performance (statistical error) of the proposed algorithms (gLSTD, Accelerated-TD and Optimal-TD algorithms) with those of the online and batch baselines: TD algorithm (Sutton and Barto, 1998) and LSTD algorithm (Bradtke and Barto, 1996), respectively, in a very simple problem. An MRP trajectory was generated from a simple Markov random walk on a chain with five states ($s=1,\cdots,5$) as depicted in Figure 3. At each time t, the state changes to either of its left (-1) or right ($+1$) with equal probability of 0.5. A reward function was set as a deterministic function of the state:
$r=[0.6594,-0.3870,-0.9742,-0.9142,0.9714]$[6] and the discount factor was set to 0.95. The value function was approximated by a linear function with three-dimensional basis functions, that is, $V(s)\approx\sum_{n=1}^{3}\theta_n\phi_n(s)$. The basis functions $\phi_n(s)$ were generated according to a diffusion model (Mahadevan and Maggioni, 2007); basis functions were given based on the minor eigenvectors of the

6. This reward function was prepared as follows. We first set the true value function by choosing the basis function and generating the parameter randomly, then the reward function was set so that it satisfied the Bellman equation.

Online Learning: $\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} - \eta_t \hat{\mathbf{R}}_t \boldsymbol{w}_{t-1}(Z_{t-1})\varepsilon(z_t, \hat{\boldsymbol{\theta}}_t)$

• $\boldsymbol{f}_T^{\mathrm{TD}}(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^T \boldsymbol{\phi}(s_{t-1})\varepsilon(z_t, \boldsymbol{\theta})$	
• TD (Sutton, 1988)	$\hat{\mathbf{R}}_t = \mathbf{R} = \mathbf{I}$
• NTD (Bradtke and Barto, 1996)	$\hat{\mathbf{R}}_t = \{(1/t)\sum_{i=1}^t \boldsymbol{\phi}(s_i)^\top \boldsymbol{\phi}(s_i)\}^{-1}\mathbf{I}$
• LSPE (Nedić and Bertsekas, 2003)	$\hat{\mathbf{R}}_t = \{(1/t)\sum_{i=1}^t \boldsymbol{\phi}(s_i)\boldsymbol{\phi}(s_i)^\top\}^{-1}$
• GTD (Sutton et al., 2009b)	See Equations (9) and (10) in the literature
• GTD2 (Sutton et al., 2009a)	See Equations (8) and (9) in the literature
• TDC (Sutton et al., 2009a)	See Equations (9) and (10) in the literature
• iLSTD (Geramifard et al., 2006)	See Algorithm 3 in the literature
• *Accelerated-TD Learning*	$\hat{\mathbf{R}}_t = \{(1/t)\sum_{i=1}^t \boldsymbol{\phi}(s_{i-1})(\boldsymbol{\phi}(s_{i-1}) - \gamma\boldsymbol{\phi}(s_i))^\top\}^{-1}, \quad \eta_t = 1/t$
• $\boldsymbol{f}_T^{\mathrm{TD}(\lambda)}(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^T \sum_{i=1}^t (\gamma\lambda)^{t-i}\boldsymbol{\phi}(s_{i-1})\varepsilon(z_t, \boldsymbol{\theta})$	
• TD(λ) (Sutton, 1988)	$\hat{\mathbf{R}}_t = \mathbf{R} = \mathbf{I}$
• NTD(λ) (Bradtke and Barto, 1996)	$\hat{\mathbf{R}}_t = \{(1/t)\sum_{i=1}^t \boldsymbol{\phi}^\top(s_i)\boldsymbol{\phi}(s_i)\}^{-1}\mathbf{I}$
• LSPE(λ) (Nedić and Bertsekas, 2003)	$\hat{\mathbf{R}}_t = \{(1/t)\sum_{i=1}^t \boldsymbol{\phi}(s_i)\boldsymbol{\phi}(s_i)^\top\}^{-1}$
• iLSTD(λ) (Geramifard et al., 2007)	See Algorithm 2 in the literature
• $\boldsymbol{f}_T^{\mathrm{RG}}(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^T \left(\boldsymbol{\phi}(s_{t-1}) - \gamma\mathbb{E}_{\boldsymbol{\theta}^*, \boldsymbol{\xi}_s^*}[\boldsymbol{\phi}(s_t)\|s_{t-1}]\right)\varepsilon(z_t, \boldsymbol{\theta})$	
• RG (Baird, 1995)	$\hat{\mathbf{R}} = \mathbf{R} = \mathbf{I}$
• $\boldsymbol{f}_T^*(Z_T, \boldsymbol{\theta})$ given by Equation (18)	
• *Optimal-TD Learning*	$\hat{\mathbf{R}}_t = \hat{\mathbf{Q}}_t^{-1}, \eta_t = 1/t$

Batch Learning: $\hat{\boldsymbol{\theta}}_T = \left[\sum_{t=1}^T \boldsymbol{w}_{t-1}(Z_{t-1})(\boldsymbol{\phi}(s_{t-1}) - \gamma\boldsymbol{\phi}(s_t))^\top\right]^{-1}\left[\sum_{t=1}^T \boldsymbol{w}(Z_{t-1})r_t\right]$

• $\boldsymbol{f}_T^{\mathrm{TD}}(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^T \boldsymbol{\phi}(s_{t-1})\varepsilon(z_t, \boldsymbol{\theta})$
• LSTD (Bradtke and Barto, 1996)
• $\boldsymbol{f}_T^{\mathrm{TD}(\lambda)}(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^T \sum_{i=1}^t (\gamma\lambda)^{t-i}\boldsymbol{\phi}(s_{i-1})\varepsilon(z_t, \boldsymbol{\theta})$
• LSTD(λ) (Boyan, 2002)
• $\boldsymbol{f}_T^*(Z_T, \boldsymbol{\theta})$ given by Equation (18)
• *gLSTD*

Table 1: Relation between the current learning and the proposed algorithms. $\mathbf{I}$ denotes an $m \times m$ identity matrix.

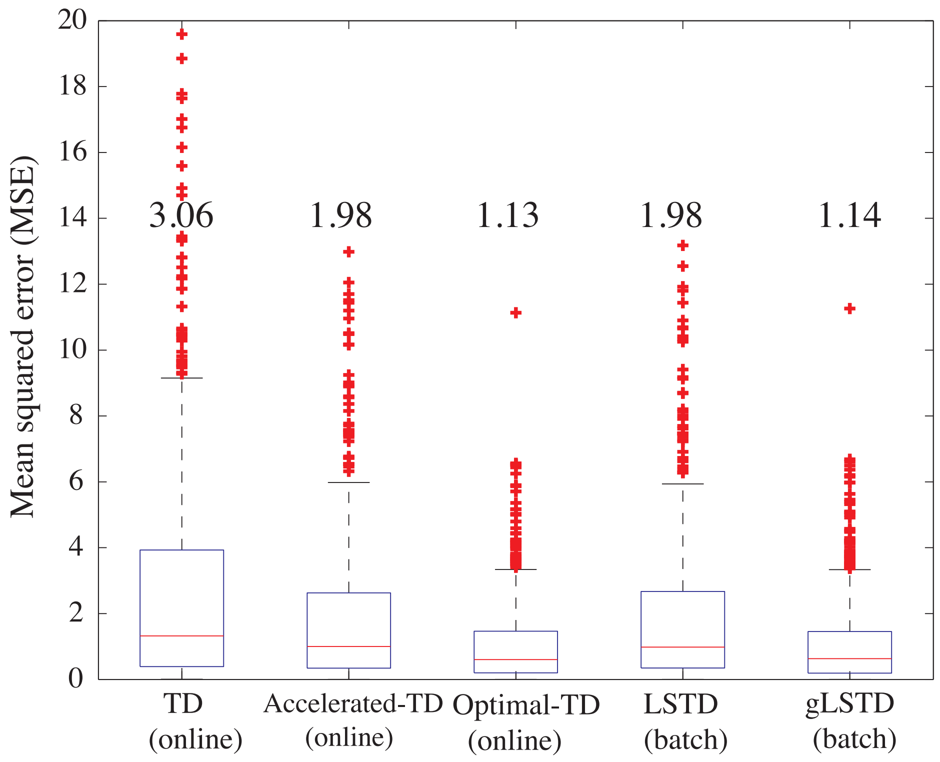


Figure 4: Boxplots of MSE both of the online (TD, Accelerated-TD and Optimal-TD) and batch (LSTD and gLSTD) algorithms. The center line, and the upper and lower sides of each box denote the median of MSE, and the upper and lower quartiles, respectively. The number above each box is the average MSE.

graph Laplacian on an undirected graph constructed by the state transition. The basis functions actually used in this simulation were $\phi(s_1) = [1, -0.6015, 0.5117]^\top$, $\phi(s_2) = [1, -0.3717, -0.1954]^\top$, $\phi(s_3) = [1, 0, -0.6325]^\top$, $\phi(s_4) = [1, 0.3717, -0.1954]^\top$, and $\phi(s_5) = [1, 0.6015, 0.5117]^\top$. In general, there is no guarantee that the true value function is included in the space spanned by the generated basis functions. In our example, however, the true value function can be represented faithfully by the basis vectors above.

We first generated $M = 500$ trajectories (episodes) each of which consisted of $T = 500$ random walk steps. The value function was estimated for each episode. We evaluated the mean squared error (MSE) between the true value function and the estimated value function, evaluated over the five states.

Figure 4 shows the boxplots of the MSE of the value functions estimated by the proposed (Accelerated-TD, Optimal-TD and gLSTD) and baseline (TD and LSTD) algorithms, in which the MSEs of all 500 episodes are shown by box-plots. For this example, the conditional expectations both in Optimal-TD and gLSTD can be calculated by sample average in each state, because there were only five states. In the online algorithms (TD, Accelerated-TD, and Optimal-TD), we used some batch procedures to obtain initial estimates of the parameters, as is often done in many online procedures. More specifically, the first 10 steps in each episode were used to obtain initial estimators in a batch manner and the online algorithm started after the 10 steps.

In the proposed online algorithms (Accelerated-TD and Optimal-TD), the stepsizes were decreased as simple as $1/t$. On the other hand, the convergence of TD learning was too slow in the simple $1/t$ setting due to fast decay of the stepsizes; this slow convergence was also observed when employing a certain well-chosen constant stepsize. Therefore, we adopt an ad-hoc adjustment for the stepsizes as $1/\tau$, where $\tau = \alpha_0(n_0+1)/(n_0+t)$. The best α_0 and n_0 have been selected by searching the sets of $\alpha_0 \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$ and $n_0 \in \{10, 50, 100, 150, 200, 250, 300, 400, 500, 1000\}$, so that α_0 and n_0 are selected as 0.3 and 200, respectively.

As shown in Figure 4, the Optimal-TD and gLSTD algorithms achieved the minimum MSE among the online and batch algorithms, respectively. The MSEs by these two methods were comparable.[7] It should be noted that the Accelerated-TD algorithm performed significantly better than the ordinary TD algorithm, showing the matrix $\mathbf{R}$ was effective for accelerating the convergence of the online procedure as expected by our theoretical analysis.

Figure 5 shows how the estimation error of the estimator ($\hat{\boldsymbol{\theta}}_T$) behaves as the learning proceeds, both for online (upper panel) and batch (lower panel) learning algorithms. X-axis and y-axis denote the number of learning steps and the estimation error, that is, the MSE between the true parameter and estimated parameter, average over 500 runs, respectively. The theoretical results, dotted and solid lines, exhibit good accordance with the simulation results, crosses and circles, respectively, as expected. Although our theoretical methods were mostly based on asymptotic analysis, they were supported by simulation results even in the cases of relatively small number of samples.

8. Discussion and Future Work

The contributions of this study are to present a new semiparametric approach to the model-free policy evaluation, which generalizes most of the current policy evaluation methods, and to clarify statistical properties of the policy evaluation problem. On the other hand, our framework to evaluate the policy evaluation has been restricted to situations in which the function approximation is faithful, that is, there is no model misspecification for the value function; we have not referred to statistical behaviors of our proposed algorithms in misspecified cases. In fact, the proposed algorithms may not better than current algorithms when the choice of parametric function g or the preparation of basis functions for approximating the value function introduces bias. Also, it is unsure whether our proposed online algorithms converge or not in misspecified cases. Figure 6 shows an example where the proposed algorithms (Optimal-TD and gLSTD) fail to obtain the best estimation accuracy. Here, an MRP trajectory was generated from an Markov random chain on the same dynamics as in Section 7. Rewards $+1$ and -1 were given when arriving at states '1' and '20', respectively, and the discounted factor was set at 0.98. Under this setting, we generated $M = 500$ trajectories (episodes) each of which consisted of $T = 1000$ random walk steps. We tested two linear function approximations with eight-dimensional and four-dimensional basis functions, respectively, which were also generated by the diffusion model. The former basis functions cause a tiny bias which can be ignored, whereas the latter ones make a significant bias. The upper and lower panels in Figure 6 show the

7. In a particular implementation of the gLSTD algorithm (Algorithm 1) here, we used the whole sample trajectory to approximate the weight function $\boldsymbol{w}_t^*$, that is, $k = 0$, implying gLSTD does not necessarily hold consistency. Based on good agreement of the results between gLSTD and Optimal-TD, however, we can speculate that the approximation of the weight function using the whole sample trajectory did not affect the estimation accuracy so much.

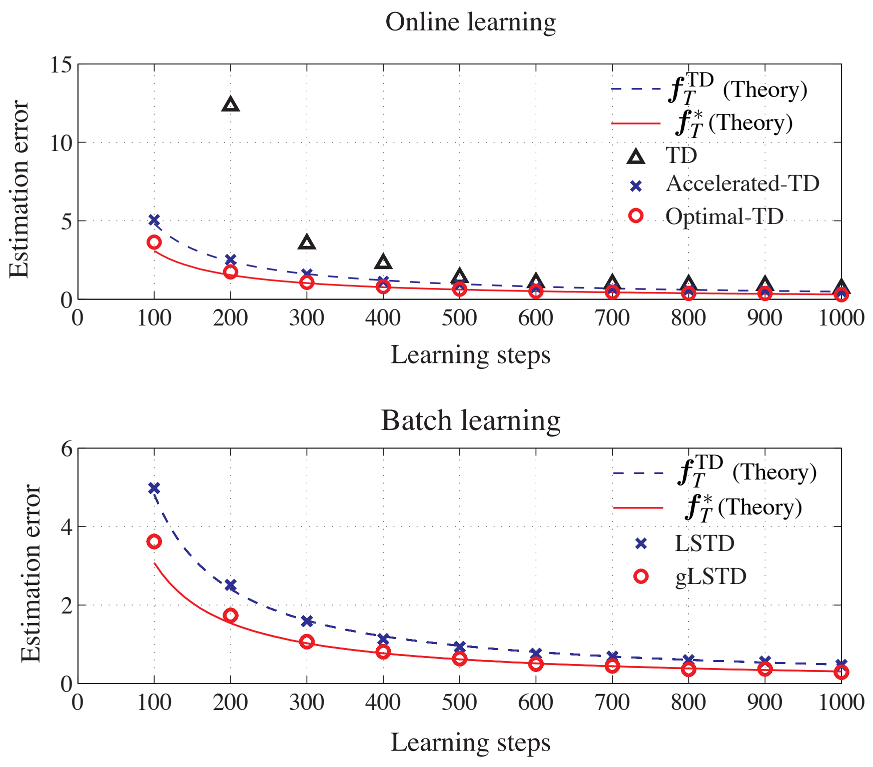


Figure 5: 500 learning runs by varying the initial conditions were performed. (Upper panel) Triangles ($\triangle$), crosses ($\times$) and circles ($\circ$) denote the simulation results for TD, Accelerated-TD and Optimal-TD, respectively. They were averaged over the 500 runs. The dotted and solid lines show the theoretical results discussed in Lemma 3 for estimating functions $\boldsymbol{f}_T^{\mathrm{TD}}$ and $\boldsymbol{f}_T^*$ described in Section 6. (Lower panel) Crosses ($\times$) and circles ($\circ$) denote the simulation results for LSTD and gLSTD, respectively.

MSEs of the value functions estimated by the proposed (Accelerated-TD, Optimal-TD and gLSTD) and baseline (TD and LSTD) algorithms employing eight-dimensional and four-dimensional basis functions, respectively. For scheduling of stepsizes in the online algorithms, we followed the same procedures as in Section 7. In the well-specified case (upper panel), the proposed algorithms achieved the smaller MSEs than the baseline algorithms as expected by our analysis, while in the misspecified case (lower panel), our proposed algorithms were inferior to the baseline algorithms. These results indicate the limitation of our analysis. When the bias-variance trade-off cannot be ignored, it is not sufficient to consider solely the asymptotic variance. Therefore, we need to analyze a *risk* $\mathcal{R}(\hat{\boldsymbol{\theta}}_T)$ which represents the deviation between the estimated value function and the true value function. Also, it is an important future work to construct good parametric representations (e.g., basis functions in linear cases) which attain small modeling biases. Furthermore, it is necessary to extend our convergence analysis for online learning algorithms to applicable to misspecified cases.

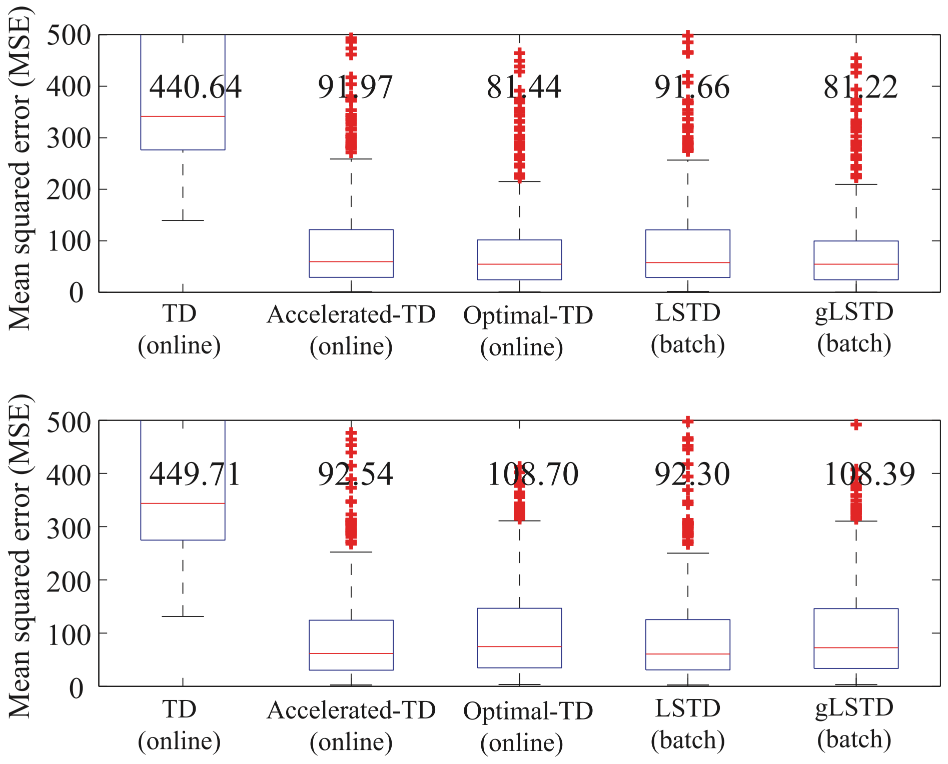


Figure 6: Boxplots of MSE for both of the online (TD, Accelerated-TD and Optimal-TD) and batch (LSTD and gLSTD) algorithms on a twenty states Markov random walk problem. (Upper panel) Simulation results on the function approximation with eight-dimensional diffusion basis functions. (Lower panel) Simulation results on the function approximation with four-dimensional diffusion basis functions.

8.1 Asymptotic Analysis in Misspecified Situations

First, let us revisit the asymptotic variance of the estimating function (15). In misspecified cases, estimating function (14) does not necessarily satisfy the martingale property, then its asymptotic variance can no longer be calculated by Equation (16). However, by introducing a notion of *uniform mixing*, the asymptotic variance can be correctly evaluated, even in misspecified cases.

To clarify the following discussion, we only consider the class of estimators given by the following estimating function $\boldsymbol{f}_T : S^{T+1} \times R^T \times \Theta \mapsto \mathbb{R}^m$:

$$\boldsymbol{f}_T(Z_T, \boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\psi}_t(Z_t, \boldsymbol{\theta}) \equiv \sum_{t=1}^{T} \boldsymbol{w}_{t-1}(Z_{t-1}) \varepsilon(z_t, \boldsymbol{\theta}). \tag{30}$$

Note that the class of estimators characterized by the above estimating function (30) is general enough for our theoretical consideration because it leads to almost all of the major algorithms for model-free policy evaluation that have been proposed so far (see Table 1). Now we demonstrate with Lemma 11 that the asymptotic variance of the estimators $\hat{\boldsymbol{\theta}}_T$ given by the estimating equation

$$\sum_{t=1}^{T} \boldsymbol{\psi}_t(Z_t, \hat{\boldsymbol{\theta}}) = \mathbf{0}. \tag{31}$$

Lemma 11 *Suppose that the random sequence Z_T is generated from the distribution $p(Z_T)$ defined by Equation (1). Assume that:*

(a) There exists such a parameter value $\boldsymbol{\theta} \in \Theta$ that

$$\lim_{t\to\infty} \mathbb{E}\left[\boldsymbol{\psi}_t(Z_t, \boldsymbol{\theta})\right] = \mathbf{0},$$

where that $\mathbb{E}[\cdot]$ denotes the expectation with respect to $p(Z_T)$, and $\hat{\boldsymbol{\theta}}_T$ converges to the parameter $\boldsymbol{\theta}$ in probability.[8]

(b) There exists a limit of matrix $\mathbb{E}\left[\boldsymbol{w}_{t-1}(Z_{t-1})\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t, \boldsymbol{\theta})\}^\top\right]$ and $\lim_{t\to\infty} \mathbb{E}\left[\boldsymbol{w}_{t-1}(Z_{t-1})\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t, \boldsymbol{\theta})\}^\top\right]$ is nonsingular.

(c) $\mathbb{E}\left[\|\boldsymbol{w}_{t-1}(Z_{t-1})\varepsilon(z_t, \boldsymbol{\theta})\|^2\right]$ is finite for any t.

Then, the estimator derived from estimating Equation (31) has the asymptotic variance

$$\widetilde{\mathrm{Av}} \equiv \widetilde{\mathrm{Av}}(\hat{\boldsymbol{\theta}}_T) \equiv \mathbb{E}\left[(\hat{\boldsymbol{\theta}}_T - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}_T - \boldsymbol{\theta})^\top\right] = \frac{1}{T}\mathbf{A}^{-1}\boldsymbol{\Sigma}\left(\mathbf{A}^\top\right)^{-1}, \tag{32}$$

where

$$\mathbf{A} \equiv \mathbf{A}(\boldsymbol{\theta}) \equiv \lim_{t\to\infty} \mathbb{E}\left[\boldsymbol{w}_{t-1}\left\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t, \boldsymbol{\theta})\right\}^\top\right],$$
$$\boldsymbol{\Sigma} \equiv \boldsymbol{\Sigma}(\boldsymbol{\theta}) \equiv \lim_{t\to\infty} \mathbb{E}\left[\varepsilon(z_t, \boldsymbol{\theta})^2 \boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right] + \lim_{t\to\infty} 2\sum_{t'=1}^{\infty} \mathrm{cov}\left[\varepsilon(z_t, \boldsymbol{\theta})\boldsymbol{w}_{t-1}, \varepsilon(z_{t+t'}, \boldsymbol{\theta})\boldsymbol{w}_{t+t'-1}\right].$$

Here, $\boldsymbol{w}_t$ and $\mathrm{cov}[\cdot,\cdot]$ denote the abbreviation of $\boldsymbol{w}_t(Z_t)$ and the covariance function, respectively.

The proof is given in Appendix L. Since this proof required the central limit theorem under uniform mixing condition, we briefly review the notion and properties of uniform mixing in Appendix B. We note that the infinite sum of covariance in Equation (32) becomes zero when the parametric representation of the value function is faithful. This implies that Lemma 11 generalizes the result of Lemma 3.

Furthermore, we can derive the upper bound of the asymptotic variance (32).

Lemma 12 *There exists such a positive constant Υ that*

$$\frac{1}{T}\mathbf{A}^{-1}\boldsymbol{\Sigma}\left(\mathbf{A}^\top\right)^{-1} \preceq \frac{\Upsilon}{T}\mathbf{A}^{-1}\boldsymbol{\Sigma}_0\left(\mathbf{A}^\top\right)^{-1}$$

holds, where $\boldsymbol{\Sigma}_0 \equiv \lim_{t\to\infty} \mathbb{E}\left[\varepsilon(z_t, \boldsymbol{\theta})^2 \boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]$.

8. We can show the stochastic convergence of the estimator to the parameter $\boldsymbol{\theta}$ by imposing further mild conditions to $\boldsymbol{f}_T$. The proof can be obtained by following the procedure used in Theorem 3.6 in Sørensen (1999).

The proof is given in Appendix M. This lemma addresses that the estimators, which we have proposed so far, minimize the upper bound of the asymptotic variance in misspecified cases.

Lemma 11 allows us to see the asymptotic behavior of the risk, like done by the previous work in a different context; Liang and Jordan (2008) evaluated the quality of probabilistic model-based predictions in a structured prediction task. They analyzed the expected log-loss (risk) of composite likelihood estimators and compared it with those of generative, discriminative and pseudo-likelihood estimators, both when the probabilistic models are well-specified and misspecified. Since composite likelihood estimators are in the class of M-estimator, we will be able to evaluate the risk of various estimators by performing a similar analysis to Liang and Jordan (2008). Konishi and Kitagawa (1996) introduced generalized information criterion (GIC) which could be applied to evaluate statistical models constructed by various types of estimation procedures. GIC is the generalization of the well-known Akaike information criterion (AIC) (Akaike, 1974) and provided an unbiased estimator for the expected log-loss (risk) of statistical models obtained by M-estimators. Therefore, it may be possible to select a good model from a set of potential models by constructing an information criterion for model-free policy evaluation based on the analysis in Konishi and Kitagawa (1996).

8.2 Online Learning Procedures in Large Scale Situations

In both Optimal-TD and Accelerated-TD learning, it is necessary to maintain the inverse of the scaling matrix $\hat{\mathbf{R}}_t$. Since this matrix inversion operation costs $O(m^2)$ in each step, maintaining the inverse matrix becomes expensive when the dimensionality of the parameters increases. An efficient implementation in such a large-scale setting is to use a coarsely-represented scale matrix, for example, a diagonal or a block diagonal matrix. An appropriate setting still ensures the convergence rate of $O(1/t)$ without losing the computational efficiency. Le Roux et al. (2008) presented an interesting implementation of natural gradient learning (Amari, 1998) for large-scale settings, which was called “TONGA”. TONGA uses a low-rank approximation of the scaling matrix and casted both problems of finding the low-rank approximation and computing the gradient onto a lower-dimensional space, thereby attaining a lower computational complexity. Therefore, by applying such an idea to our proposed algorithms, we can improve the computational complexity without sacrificing the fast convergence.

9. Conclusions

We introduced a framework of semiparametric statistical inference for value function estimation which can be applied to analyzing both batch learning and online learning procedures. Based on this framework, we derived the general form of estimating functions for model-free value function estimation in MRPs, which provides a statistical basis to many batch and online learning algorithms available currently for policy evaluation. Moreover, we found an optimal estimating function, which yields the minimum asymptotic estimation variance amongst the general class, and presented new learning algorithms based on it as both batch and the online procedures. Using a simple MRP problem, we confirmed the validity of our analysis; actually, our proposed algorithms showed reasonably good performance.

Acknowledgments

This study was partly supported by Grant-in-Aid for Scientific Research (B) (No.21300113) from Japan Society for the Promotion of Science (JSPS) and the Federal Ministry of Economics and Technology of Germany (BMWi) under the project THESEUS, grant 01MQ07018. Tsuyoshi Ueno was also supported by JSPS Fellowship for Young Scientist.

Appendix A. Stochastic Order Symbols

The stochastic order symbols O_p and o_p are useful when evaluating the rate of convergence by means of asymptotic theory. Let n denote the number of observations. The stochastic order symbols are defined as follows.

Definition 13 *Let $\{X_n\}$ and $\{R_n\}$ denote a sequence of random variables and a sequence of real numbers, respectively. Then $X_n = o_p(R_n)$ if and only if X_n/R_n converges in probability to* 0 *when $n \to \infty$.*

Definition 14 *Let $\{X_n\}$ and $\{R_n\}$ denote a sequence of random variables and a sequence of real numbers, respectively. Then $X_n = O_p(R_n)$ if and only if X_n/R_n is bounded in probability when $n \to \infty$. "Bounded in probability" means that there exist a constant C_ε and a natural number $n_0(\varepsilon)$ such that for any $\varepsilon > 0$ and $n > n_0(\varepsilon)$,*

$$\mathbf{P}\{|X_n| \leq C_\varepsilon\} \geq 1 - \varepsilon$$

holds.

Most properties of the usual orders also apply to stochastic orders. For instance,

$$\begin{aligned} o_p(1) + o_p(1) &= o_p(1), \\ o_p(1) + O_p(1) &= O_p(1), \\ O_p(1)o_p(1) &= o_p(1), \\ (1 + o_p(1))^{-1} &= O_p(1), \\ o_p(R_n) &= R_n o_p(1), \\ O_p(R_n) &= R_n O_p(1), \\ o_p(O_p(1)) &= o_p(1). \end{aligned}$$

Moreover, by taking the expectation, the stochastic order symbol $o_p(\cdot)$ reduces to the usual order symbol $o(\cdot)$.

Remark 15 *Let $\{X_n\}$ and $\{R_n\}$ denote a sequence of random variables which satisfies $X_n = o_p(1)$ and a sequence of real numbers, respectively. Let $Y_n = X_n R_n$ denote a random variable which satisfies $Y_n = o_p(R_n)$. If the sequence of random variable Y_n is asymptotically uniformly integrable, then, the expectation of the random variables Y_n has the same normal order, $\mathbb{E}[Y_n] = o(R_n)$.*

This remark can be shown from Theorem 2.20 in van der Vaart (2000). Note that the sequence of real numbers $\{R_n\}$ which appears in Definition 13 and 14 corresponds to the *convergence rate*; then $Y_n = o_p(R_n)$ and $Y_n = O_p(R_n)$ mean that the sequence Y_n converges in probability to zero and is bounded in probability, respectively, at the rate of R_n.

Appendix B. Uniform Mixing and Central Limit Theorem

The notion of *mixing* is important when analyzing the rate of convergence in the stochastic processes which do not satisfy the martingale condition. There are several different definitions for mixing. In this section, we will especially focus on *uniform mixing* which is defined as follows.

Definition 16 *Let* $\mathrm{Y} \equiv \{Y_t : t = 1, 2, \dots\}$ *be a strictly stationary process*[9] *on a probabilistic space* $(\Omega, \mathcal{F}, P)$ *and* $\mathcal{F}_k^m$ *be* σ*-algebra generated by* $\{Y_k, \cdots, Y_m\}$*. Then, the process* Y *is said to be uniform mixing (*ϕ*-mixing) if* $\varphi(t) \to 0$ *as* $t \to \infty$ *where*

$$\varphi(t) \equiv \sup_{A \in \mathcal{F}_1^k, B \in \mathcal{F}_{k+t}^\infty} |P(B|A) - P(B)|, \;\; P(A) \neq 0.$$

The function $\varphi(t)$ is called *mixing coefficient*. If the mixing coefficient $\varphi(t)$ converges to zero as fast as exponential, then Y is called *geometrically uniform mixing*.

Definition 17 *Suppose that* Y *is a strictly stationary process. If there exist some constants* $C > 0$ *and* $\rho \in [0, 1)$ *such that*

$$\varphi(t) < C\rho^t,$$

then Y *is said to be geometrically uniform mixing.*

Let f be a Borel function on the state space and define $f_T = 1/T \sum_{t=1}^T f(Y_t)$. We now consider the conditions under which the central limit theorem holds for f_T.

Lemma 18 *(Ibragimov and Linnik, 1971, Theorem 18.5.2.) Suppose that* $\{\mathrm{Y}_T\}$ *is a strictly stationary process with geometrically uniform mixing. If* $\lim_{t\to\infty} \mathbb{E}[\|f(Y_t)\|^2]$ *is finite, then the central limit theorem holds for* f*, that is,*

$$\sqrt{T}\left(f_T - \lim_{t\to\infty} \mathbb{E}[f(Y_t)]\right) \xrightarrow{d} \mathcal{N}(0, \sigma^2),$$

as $T \to \infty$ *where* $\sigma^2 \equiv \lim_{t\to\infty} \mathbb{E}[f(Y_t)^2] + 2 \lim_{t\to\infty} \sum_{t'=1}^\infty \mathrm{cov}\,[f(Y_t), f(Y_{t+t'})]$.

Note that, unlike the i.i.d. or the martingale case, the variance of the asymptotic distribution involves the correlation between different times. Generally, such time dependency makes finding an exact relationship difficult; however, it may be easy to evaluate the upper bound of the time-dependent covariance.

Lemma 19 *(Ibragimov and Linnik, 1971, Theorem 17.2.3.) Suppose that* Y *is a strictly stationary process with uniform mixing. Let* f *and* g *be measurable functions with respect to* $\mathcal{F}_1^k$ *and* $\mathcal{F}_{k+t}^\infty$*, respectively. If* f *and* g *satisfy*

$$\mathbb{E}\,[|f|^p] < \infty, \;\; \mathbb{E}\,[|g|^q] < \infty,$$

where $p, q > 1$*,* $p + q = 1$*, then*

$$|\mathbb{E}\,[fg] - \mathbb{E}\,[f]\,\mathbb{E}\,[g]| \leq 2\varphi(t)^{1/p}\mathbb{E}\,[|f|^p]^{1/p}\,\mathbb{E}\,[|g|^q]^{1/q}.$$

Finally in this section, we consider the conditions that Markov processes satisfy the uniform mixing condition.

9. In a strictly stationary stochastic process, joint probability distribution is consistent when shifted in time.

Lemma 20 *(Bradley, 2005, Theorem 3.1) Suppose that* Y *is a strictly stationary, finite state Markov process. Then the following statements are equivalent:*

(a) Y *is irreducible and aperiodic.*

(b) Y *is ergodic.*

(c) Y *is geometrically uniform mixing.*

Note that if a finite state Markov process has an unique and invariant stationary distribution, it implies ergodicity. Then Lemma 20 addresses that such Markov process is uniform mixing.

Appendix C. Proof of Lemma 2

Proof Condition corresponding to (12) is satisfied by condition (c) in Assumption 4. Also, condition (10) is satisfied by Equation (13). From condition (a) in Assumption 4, the expectation of the derivative of the function $\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\varepsilon(z_t,\boldsymbol{\theta})$ can be expressed as

$$\begin{aligned}
&\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[\partial_{\boldsymbol{\theta}}\left\{\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\varepsilon(z_t,\boldsymbol{\theta})\right\}\right]\\
&=\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[\partial_{\boldsymbol{\theta}}\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\varepsilon(z_t,\boldsymbol{\theta})\right]+\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta})\right]\\
&=\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[\partial_{\boldsymbol{\theta}}\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\underbrace{\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}\left[\varepsilon(z_t,\boldsymbol{\theta})|Z_{t-1}\right]}_{=0}\right]+\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta})\right]\\
&=\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}}\left[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta})\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta})\right],
\end{aligned}$$

where we have used the fact in Equation (13). Therefore, using condition (b) in Assumption 4, we can show that condition (11) is satisfied. ■

Appendix D. Proof of Lemma 3

Proof By performing a Taylor series expansion of estimating Equation (15) around the true parameter $\boldsymbol{\theta}^*$, we obtain

$$\mathbf{0}=\sum_{t=1}^{T}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)+\sum_{t=1}^{T}\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)(\hat{\boldsymbol{\theta}}_T-\boldsymbol{\theta}^*)+O_p\left(\left\|\hat{\boldsymbol{\theta}}_T-\boldsymbol{\theta}^*\right\|^2\right).$$

Here, high order terms of the above equation are in total represented as $O_p(\|\hat{\boldsymbol{\theta}}_T-\boldsymbol{\theta}^*\|^2)$ because of Assumption 3, that is, the twice differentiable condition for the function $g(s,\boldsymbol{\theta})$. By applying the law of large numbers (ergodic pointwise theorem) (Billingsley, 1995, Theorem 24.1) to $(1/T)\sum_{t=1}^{T}\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)$ and the martingale central limit theorem (Billingsley,

1961) to $(1/\sqrt{T})\sum_{t=1}^{T}\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)$, we have

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*) &= \frac{1}{T}\sum_{t=1}^{T}\partial_{\boldsymbol{\theta}}\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\varepsilon(z_t,\boldsymbol{\theta}^*) + \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*) \\
&\xrightarrow{a.s.} \underbrace{\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}}\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\varepsilon(z_t,\boldsymbol{\theta}^*)\right]}_{=\mathbf{0}} \\
&\quad + \underbrace{\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top\right]}_{=\mathbf{A}} \\
\frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*) &= \frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\varepsilon(z_t,\boldsymbol{\theta}^*) \\
&\xrightarrow{d} \mathcal{N}\left(\mathbf{0}, \underbrace{\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\varepsilon(z_t,\boldsymbol{\theta}^*)^2\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)^\top\right]}_{=\boldsymbol{\Sigma}}\right).
\end{aligned}$$

By neglecting higher order terms, we obtain

$$\sqrt{T}(\hat{\boldsymbol{\theta}}_T - \boldsymbol{\theta}^*) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{A}^{-1}\boldsymbol{\Sigma}(\mathbf{A}^\top)^{-1}\right).$$

Then, $\hat{\boldsymbol{\theta}}_T$ is Gaussian distributed: $\hat{\boldsymbol{\theta}}_T \sim \mathcal{N}(\boldsymbol{\theta}^*, \mathrm{Av})$, where the asymptotic variance Av is given by Equation (16). ■

Appendix E. Proof of Theorem 4

Proof From Equation (2), for any t, the value function $V(s_t) = g(s_t,\boldsymbol{\theta})$ must satisfy

$$\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[r_{t+1}|s_t] = g(s_t,\boldsymbol{\theta}) - \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[g(s_{t+1},\boldsymbol{\theta})|s_t],$$

regardless of the nuisance parameter $\boldsymbol{\xi}$. Then, the TD error
$\varepsilon(z_{t+1},\boldsymbol{\theta}) = g(s_t,\boldsymbol{\theta}) - \gamma g(s_{t+1},\boldsymbol{\theta}) - r_{t+1}$ must satisfy $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\varepsilon(z_{t+1},\boldsymbol{\theta})|Z_t] = 0$ for any t and $\boldsymbol{\xi}$. Also, from the condition of martingale estimating functions, for any time t, the estimating function must satisfy

$$\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\boldsymbol{f}_{t+1}(Z_{t+1},\boldsymbol{\theta}) - \boldsymbol{f}_t(Z_t,\boldsymbol{\theta})|Z_t] = \mathbf{0}, \tag{33}$$

regardless of the nuisance parameter $\boldsymbol{\xi}$. If we can show from Equation (33) that
$\boldsymbol{f}_{t+1}(Z_{t+1},\boldsymbol{\theta}) - \boldsymbol{f}_t(Z_t,\boldsymbol{\theta}) = \boldsymbol{w}_t(Z_t,\boldsymbol{\theta})\varepsilon(z_{t+1},\boldsymbol{\theta})$ holds, $\boldsymbol{f}_T(Z_T,\boldsymbol{\theta})$ must have the form (17) by induction. Since this statement can be considered component-wise, we will prove the similar claim for scalar functions, that is,

$$\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[h(Z_{t+1},\boldsymbol{\theta})|Z_t] = 0,\ \forall\boldsymbol{\xi}_s \quad\Longrightarrow\quad h(Z_{t+1},\boldsymbol{\theta}) = w(Z_t,\boldsymbol{\theta})\varepsilon(z_{t+1},\boldsymbol{\theta}), \tag{34}$$

in the following two steps.

1. We first prove in a *constructive* manner that any simple function $h(z_{t+1},\boldsymbol{\theta})$ which depends only on z_{t+1} and satisfy $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[h(z_{t+1},\boldsymbol{\theta})|s_t]=0$ and $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\{h(z_{t+1},\boldsymbol{\theta})\}^2]<\infty$ for any t, $\boldsymbol{\theta}$ and $\boldsymbol{\xi}_s$ can be expressed as $h(z_{t+1},\boldsymbol{\theta})=w(s_t,\boldsymbol{\theta})\varepsilon(z_{t+1},\boldsymbol{\theta})$, where $w(s_t,\boldsymbol{\theta})$ is a function of s_t.

2. Our claim (34) for general function $h(Z_{t+1},\boldsymbol{\theta})$ is derived from the fact shown in the previous step, because for each fixed Z_t this problem boils down to the simple case above.

To prove the simple case first, for arbitrary fixed s_t and $\boldsymbol{\theta}$, we consider the set $\mathcal{M}(s_t,\boldsymbol{\theta})$ of all probability distributions of r_{t+1} and s_{t+1} with each of which the expectation of the TD error $\varepsilon(z_{t+1},\boldsymbol{\theta})$ vanishes. In the following discussion, s_t is treated as a *fixed constant*. In our semiparametric case, this set can be expressed as the set of all conditional distributions of r_{t+1} and s_{t+1} for given s_t which has value function $g(s_t,\boldsymbol{\theta})$ with the fixed $\boldsymbol{\theta}$, that is,

$$\begin{aligned}\mathcal{M}(s_t,\boldsymbol{\theta}) \equiv \{\, & p(r_{t+1},s_{t+1}|s_t;\boldsymbol{\theta},\boldsymbol{\xi}_s)\,|\, s_t,\boldsymbol{\theta}\text{: fixed},\\ & \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\varepsilon(z_{t+1},\boldsymbol{\theta})|s_t]=g(s_t,\boldsymbol{\theta})-\gamma\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[g(s_{t+1},\boldsymbol{\theta})|s_t]-\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[r_{t+1}|s_t]=0\,\},\end{aligned}$$

where the nuisance parameter $\boldsymbol{\xi}_s$ is designed so that it becomes bijective with the distributions in $\mathcal{M}(s_t,\boldsymbol{\theta})$. We remark that the set $\mathcal{M}(s_t,\boldsymbol{\theta})$ and the domain of the nuisance parameter $\boldsymbol{\xi}_s$ depend on s_t and $\boldsymbol{\theta}$.

Suppose that there exists a function of $h(z_{t+1})$ which satisfies $\mathbb{E}_p[h(z_{t+1})]=0$ and $\mathbb{E}_p[\{h(z_{t+1})\}^2]<\infty$ for any $p(r_{t+1},s_{t+1})\in\mathcal{M}(s_t,\boldsymbol{\theta})$. Then, because of the linearity and continuity of the integral operator, the unbiasedness condition can be extended to any function $q(r_{t+1},s_{t+1})$ which belongs to the closed span $\overline{\mathcal{M}}(s_t,\boldsymbol{\theta})$ of $\mathcal{M}(s_t,\boldsymbol{\theta})$:

$$\sum_{s_{t+1}}\int h(z_{t+1})\,q(r_{t+1},s_{t+1})\,dr_{t+1}=0. \tag{35}$$

It is also easy to show that $\overline{\mathcal{M}}(s_t,\boldsymbol{\theta})$ contains any functions (i.e., even without satisfying the non-negativity constraint of probabilities) which satisfy the condition

$$\sum_{s_{t+1}}\int \varepsilon(z_{t+1})\,q(r_{t+1},s_{t+1})\,dr_{t+1}=0. \tag{36}$$

Indeed, we can always construct a linear representation of such a function $q(r_{t+1},s_{t+1})$ with four probability distributions in $\mathcal{M}(s_t,\boldsymbol{\theta})$ which take positive values only in two regions out of $\{(r_{t+1},s_{t+1})\,|\,\varepsilon\geq 0,\ q\geq 0\}$, $\{(r_{t+1},s_{t+1})\,|\,\varepsilon\geq 0,\ q<0\}$, $\{(r_{t+1},s_{t+1})\,|\,\varepsilon<0,\ q\geq 0\}$ and $\{(r_{t+1},s_{t+1})\,|\,\varepsilon<0,\ q<0\}$.

Now, we take a distribution $p(r_{t+1},s_{t+1})$ in $\mathcal{M}(s_t,\boldsymbol{\theta})$ which is positive over its domain[10] and consider its perturbation

$$\widetilde{q}(r_{t+1},s_{t+1})\equiv p(r_{t+1},s_{t+1})\left\{1+\delta h(z_{t+1})-\delta\frac{\mathbb{E}_p[h(z_{t+1})\varepsilon(z_{t+1})]}{\mathbb{E}_p[\varepsilon(z_{t+1})^2]}\varepsilon(z_{t+1})\right\},$$

where $\delta>0$ is a small constant and $\mathbb{E}_p$ denotes the expectation over r_{t+1} and s_{t+1} with respect to $p(r_{t+1},s_{t+1})$. This function $\widetilde{q}(r_{t+1},s_{t+1})$ does not necessarily belong to the model $\mathcal{M}(s_t,\boldsymbol{\theta})$, but

10. If there exists a region where all distributions in $\mathcal{M}(s_t,\boldsymbol{\theta})$ take 0, it is impossible to characterize the functional form of $h(z_{t+1})$ in that region. For simplicity, however, we do not consider such a pathological case in this proof.

is an element of its closed span $\overline{\mathcal{M}}(s_t,\boldsymbol{\theta})$, because it also satisfies the condition (36). Therefore, Equation (35) must hold for this perturbed function $\widetilde{q}$, leading to

$$\sum_{s_{t+1}} \int h(z_{t+1})\, \widetilde{q}(r_{t+1},s_{t+1})\, dr_{t+1} = \delta \left\{ \mathbb{E}_p[h(z_{t+1})^2] - \frac{(\mathbb{E}_p[h(z_{t+1})\varepsilon(z_{t+1})])^2}{\mathbb{E}_p[\varepsilon(z_{t+1})^2]} \right\} = 0. \tag{37}$$

From Cauchy-Schwarz's inequality, this equation holds if and only if $h(z_{t+1}) \propto \varepsilon(z_{t+1})$ and otherwise $\mathbb{E}_{\widetilde{q}}[h(z_{t+1})]$, the left-hand-side of Equation (37), becomes strictly positive, which contradicts the fact (35). Since the whole argument holds for any s_t and $\boldsymbol{\theta}$, the first claim is proved.

In the general case for the function of Z_{t+1}, we just show that any function $h(Z_{t+1},\boldsymbol{\theta})$ which satisfies $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[h(Z_{t+1},\boldsymbol{\theta})|Z_t] = 0$ and $\mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\xi}_s}[\{h(Z_{t+1},\boldsymbol{\theta})\}^2] < \infty$ for any s_t, $\boldsymbol{\theta}$ and $\boldsymbol{\xi}_s$ can be expressed as $h(Z_{t+1},\boldsymbol{\theta}) = w_t(Z_t,\boldsymbol{\theta})\varepsilon(z_{t+1},\boldsymbol{\theta})$, where $w_t(Z_t,\boldsymbol{\theta})$ is a function of Z_t and $\boldsymbol{\theta}$.

For arbitrary fixed Z_t, $h(Z_{t+1},\boldsymbol{\theta})$ can be regarded as a function of r_{t+1} and s_{t+1}. Therefore, the problem reduces to the case that the function only depends on z_{t+1}, so that we can say that $h(r_{t+1},s_{t+1},Z_t,\boldsymbol{\theta}) \propto \varepsilon(r_{t+1},s_{t+1},s_t)$. Since this relationship holds for any Z_t, we conclude that the function $h(Z_{t+1},\boldsymbol{\theta})$ must have the form $w_t(Z_t,\boldsymbol{\theta})\varepsilon(z_{t+1},\boldsymbol{\theta})$. ■

Appendix F. Proof of Lemma 5

Proof We show that the conditional expectation $\tilde{\boldsymbol{w}}_t(s_t,\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\xi}_s}[\boldsymbol{w}_t(Z_t,\boldsymbol{\theta})|s_t]$, which depends only on the current state s_t and the parameter $\boldsymbol{\theta}$, gives an equally good estimator or better estimator than those by the original weight function $\boldsymbol{w}_t(Z_t,\boldsymbol{\theta})$. As shown in Equation (16), the asymptotic variance of the estimator $\hat{\boldsymbol{\theta}}_w$ with $\boldsymbol{w}_t(Z_t,\boldsymbol{\theta})$ is given by

$$\mathrm{Av}(\hat{\boldsymbol{\theta}}_w) \equiv \frac{1}{T}\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w \left(\mathbf{A}_w^{-1}\right)^\top,$$

where $\mathbf{A}_w = \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{w}_{t-1}\left\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\right\}^\top\right] \equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\left\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\right\}^\top\right]$ and $\boldsymbol{\Sigma}_w = \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right] \equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)^\top\right]$. Here, $\boldsymbol{w}_t$ is an abbreviation of $\boldsymbol{w}_t(Z_t,\boldsymbol{\theta}^*)$. Similarly, the asymptotic variance of the estimator $\hat{\boldsymbol{\theta}}_{\tilde{w}}$ with $\tilde{\boldsymbol{w}}_{t-1}(s_{t-1},\boldsymbol{\theta})$ is given by

$$\mathrm{Av}(\hat{\boldsymbol{\theta}}_{\tilde{w}}) \equiv \frac{1}{T}\mathbf{A}_{\tilde{w}}^{-1}\boldsymbol{\Sigma}_{\tilde{w}} \left(\mathbf{A}_{\tilde{w}}^{-1}\right)^\top,$$

where $\mathbf{A}_{\tilde{w}} \equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\tilde{\boldsymbol{w}}_{t-1}\left\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\right\}^\top\right] \equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\tilde{\boldsymbol{w}}_{t-1}(s_{t-1},\boldsymbol{\theta}^*)\left\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\right\}^\top\right]$ and $\boldsymbol{\Sigma}_{\tilde{w}} \equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\tilde{\boldsymbol{w}}_{t-1}\tilde{\boldsymbol{w}}_{t-1}^\top\right] \equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\tilde{\boldsymbol{w}}_{t-1}(s_{t-1},\boldsymbol{\theta}^*)\tilde{\boldsymbol{w}}_{t-1}(s_{t-1},\boldsymbol{\theta}^*)^\top\right]$. Here, $\tilde{\boldsymbol{w}}_t$ is an abbreviation of $\tilde{\boldsymbol{w}}_t(s_t,\boldsymbol{\theta})$. The matrices $\mathbf{A}_w$ and $\boldsymbol{\Sigma}_w$ can be calculated as

$$\begin{aligned}
\mathbf{A}_w &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{w}_{t-1}\left\{\partial_{\boldsymbol{\theta}}\varepsilon_t(z_t,\boldsymbol{\theta}^*)\right\}^\top\right] \\
&= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\left\{\partial_{\boldsymbol{\theta}}\varepsilon_t(z_t,\boldsymbol{\theta}^*)\right\}^\top\right] = \mathbf{A}_{\tilde{w}}, \\
\boldsymbol{\Sigma}_w &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\Big[(\varepsilon_t^*)^2\left(\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]+\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right) \\
&\qquad\qquad \left(\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]+\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)^\top\Big] \\
&= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]^\top\right] \\
&\quad + \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\left(\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)^\top\right] \\
&\quad + \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\left(\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)\left(\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)^\top\right] \\
&\quad + \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\left(\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)\left(\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)^\top\right] \\
&= \boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}} + \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\left(\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)\left(\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)^\top\right] \\
&= \boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}} + \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\left(\boldsymbol{w}_{t-1}-\tilde{\boldsymbol{w}}_{t-1}\right)\left(\boldsymbol{w}_{t-1}-\tilde{\boldsymbol{w}}_{t-1}\right)^\top\right],
\end{aligned}$$

where we have used $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|z_{t-1}\right] = \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right] = \tilde{\boldsymbol{w}}_{t-1}$ and

$$\begin{aligned}
&\lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\left(\boldsymbol{w}_{t-1}-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]^\top\right] \\
&= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[(\varepsilon_t^*)^2\left(\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]-\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)\left(\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{w}_{t-1}|s_{t-1}\right]\right)^\top\right] = \mathbf{0}.
\end{aligned}$$

This implies that

$$\mathrm{Av}(\hat{\boldsymbol{\theta}}_w) = \frac{1}{T}\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w\left(\mathbf{A}_w^{-1}\right)^\top \succeq \frac{1}{T}\mathbf{A}_{\tilde{w}}^{-1}\boldsymbol{\Sigma}_{\tilde{w}}\left(\mathbf{A}_{\tilde{w}}^{-1}\right)^\top = \mathrm{Av}(\hat{\boldsymbol{\theta}}_{\tilde{w}}),$$

where $\succeq$ denotes the semipositive definiteness of the subtraction. ■

Appendix G. Proof of Theorem 6

Proof As shown in Equation (16), the asymptotic variance of the estimator $\hat{\boldsymbol{\theta}}_w$ is given by

$$\mathrm{Av} = \frac{1}{T}\mathbf{A}_w\boldsymbol{\Sigma}_w(\mathbf{A}_w^{-1})^\top,$$

where

$$\begin{aligned}
\mathbf{A}_w &\equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top], \\
\boldsymbol{\Sigma}_w &\equiv \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)\boldsymbol{w}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)^\top].
\end{aligned}$$

For the sake of expression simplicity, the weight function $\boldsymbol{w}_t(Z_t,\boldsymbol{\theta}^*)$ the TD error $\varepsilon(z_t,\boldsymbol{\theta}^*)$ are abbreviated as $\boldsymbol{w}_t$ and ε_t, respectively; we rewrite $\mathbf{A}_w$ and Σ as

$$\mathbf{A}_w = \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{w}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top\right],$$
$$\boldsymbol{\Sigma}_w = \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\varepsilon_t^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right].$$

We first derive the weight function that minimizes the trace of the asymptotic variance, that is,

$$\boldsymbol{w}_{t-1}^* = \underset{\boldsymbol{w}_{t-1}}{\operatorname{argmin}}\, F(\boldsymbol{w}_{t-1})$$
$$\text{where}\ \ F(\boldsymbol{w}_{t-1}) = \operatorname{tr}\{\operatorname{Av}(\boldsymbol{w}_{t-1})\}.$$

Let $\boldsymbol{\delta}_{t-1}\equiv\boldsymbol{\delta}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)$ be an arbitrary function of Z_{t-1} and $\boldsymbol{\theta}^*$. We consider how much a functional $F(\boldsymbol{w}_{t-1})$ changes when we make a small change $h\boldsymbol{\delta}_{t-1}$ to the weight function $\boldsymbol{w}_{t-1}$. For notational convenience, we define $G(h;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})\equiv F(\boldsymbol{w}_{t-1}+h\boldsymbol{\delta}_{t-1})$.[11] If the function $G(h;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})$ is twice differentiable with respect to h, then we have

$$G(h;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1}) = G(0;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1}) + h\,\partial_h G(h;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})|_{h=0} + O(h^2),$$

where ∂_h denotes the partial derivative with respect to h. Since the functional $F(\boldsymbol{w}_{t-1})$ is stationary for tiny variation in the function $\boldsymbol{w}_{t-1}$, the weight function $\boldsymbol{w}_{t-1}^*$ which minimizes the asymptotic estimation variance must satisfy

$$\partial_h G(h;\boldsymbol{w}_{t-1}^*,\boldsymbol{\delta}_{t-1})\big|_{h=0} = 0,$$

for arbitrary choice of $\boldsymbol{\delta}_{t-1}$.

The definition of derivative says

$$\partial_h G(h;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})|_{h=0} = \lim_{\lambda\to 0}\frac{G(\lambda;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1}) - G(0;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})}{\lambda}.$$

The numerator of the above equation is written as

$$\begin{aligned} &G(\lambda;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1}) - G(0;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})\\ &= \operatorname{tr}\left[\mathbf{A}_{w+\lambda\delta}^{-1}\boldsymbol{\Sigma}_{w+\lambda\delta}(\mathbf{A}_{w+\lambda\delta}^{-1})^\top\right] - \operatorname{tr}\left[\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w(\mathbf{A}_w^{-1})^\top\right], \end{aligned} \tag{38}$$

where

$$\begin{aligned} \mathbf{A}_{w+\lambda\delta} &\equiv \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[(\boldsymbol{w}_{t-1}+\lambda\boldsymbol{\delta}_{t-1})\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\\ &= \mathbf{A}_w + \lambda\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top] \end{aligned} \tag{39}$$

$$\begin{aligned} \boldsymbol{\Sigma}_{w+\lambda\delta} &\equiv \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\varepsilon_t^2(\boldsymbol{w}_{t-1}+\lambda\boldsymbol{\delta}_{t-1})(\boldsymbol{w}_{t-1}+\lambda\boldsymbol{\delta}_{t-1})^\top]\\ &= \boldsymbol{\Sigma}_w + \lambda\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\varepsilon_t^2(\boldsymbol{\delta}_{t-1}\boldsymbol{w}_{t-1}^\top + \boldsymbol{w}_{t-1}\boldsymbol{\delta}_{t-1}^\top)] + O(\lambda^2). \end{aligned} \tag{40}$$

11. We used this notation to emphasize that $G(h;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})$ is a function of h, while $\boldsymbol{w}_{t-1}$ and $\boldsymbol{\delta}_{t-1}$ are regarded as auxiliary variables.

By using the matrix inversion lemma (Horn and Johnson, 1985), $\mathbf{A}_{w+\lambda\delta}^{-1}$ can be written as

$$\begin{aligned}\mathbf{A}_{w+\lambda\delta}^{-1} &= \left(\mathbf{A}_w + \lambda \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\right)^{-1} \\ &= \mathbf{A}_w^{-1} - \lim_{t\to\infty}\mathbf{A}_w^{-1}\left(\mathbf{I}+\lambda\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\mathbf{A}_w^{-1}\right)^{-1}\lambda\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\mathbf{A}_w^{-1}.\end{aligned}$$

The matrix $\left(\mathbf{I}+\lambda\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\mathbf{A}_w^{-1}\right)^{-1}$ can be calculated as

$$\left(\mathbf{I}+\lambda\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\mathbf{A}_w^{-1}\right)^{-1} = \left(\mathbf{I}-\lambda\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\mathbf{A}_w^{-1}\right) + O(\lambda^2),$$

because of

$$\left(\mathbf{I}+\lambda\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\mathbf{A}_w^{-1}\right)\left(\mathbf{I}-\lambda\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top]\mathbf{A}_w^{-1}\right) = \mathbf{I}+O\left(\lambda^2\right).$$

Thus we obtain

$$\mathbf{A}_{w+\lambda\delta}^{-1} = \mathbf{A}_w^{-1} - \lambda\lim_{t\to\infty}\mathbf{A}_w^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top\right]\mathbf{A}_w^{-1} + O(\lambda^2), \tag{41}$$

where high order terms are summarized as $O(\lambda^2)$.

Substituting Equations (39)-(41) to Equation (38), we have

$$\begin{aligned}&\mathrm{tr}\left[\mathbf{A}_{w+\lambda\delta}^{-1}\boldsymbol{\Sigma}_{w+\lambda\delta}(\mathbf{A}_{w+\lambda\delta}^{-1})^\top\right] - \mathrm{tr}\left[\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w(\mathbf{A}_w^{-1})^\top\right] \\ &= -\lambda\lim_{t\to\infty}\mathrm{tr}\left[\mathbf{A}_w^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top\right]\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w\left(\mathbf{A}_w^{-1}\right)^\top\right] \\ &\quad -\lambda\lim_{t\to\infty}\mathrm{tr}\left[\mathbf{A}_w^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\boldsymbol{\delta}_{t-1}^\top\right]\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w\left(\mathbf{A}_w^{-1}\right)^\top\right] \\ &\quad +\lambda\lim_{t\to\infty}\mathrm{tr}\left[\mathbf{A}_w^{-1}\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\varepsilon_t^2(\boldsymbol{\delta}_{t-1}\boldsymbol{w}_{t-1}^\top+\boldsymbol{w}_{t-1}\boldsymbol{\delta}_{t-1}^\top)]\left(\mathbf{A}_w^{-1}\right)^\top\right] + O(\lambda^2) \\ &= -2\lambda\lim_{t\to\infty}\mathrm{tr}\left[\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w(\mathbf{A}_w^{-1})^\top\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\boldsymbol{\delta}_{t-1}^\top\right](\mathbf{A}_w^{-1})^\top\right] \\ &\quad +2\lambda\lim_{t\to\infty}\mathrm{tr}\left[\mathbf{A}_w^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\varepsilon_t^2\boldsymbol{w}_{t-1}\boldsymbol{\delta}_{t-1}^\top](\mathbf{A}_w^{-1})^\top\right] + O(\lambda^2).\end{aligned}$$

This gives the partial derivative $\partial_h G(h;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})$ as

$$\begin{aligned}&\partial_h G(h;\boldsymbol{w}_{t-1},\boldsymbol{\delta}_{t-1})|_{h=0} \\ &= -2\lim_{t\to\infty}\mathrm{tr}\left[\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w(\mathbf{A}_w^{-1})^\top\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\boldsymbol{\delta}_{t-1}^\top\right](\mathbf{A}_w^{-1})^\top\right] \\ &\quad +2\lim_{t\to\infty}\mathrm{tr}\left[\mathbf{A}_w^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\varepsilon_t^2\boldsymbol{w}_{t-1}\boldsymbol{\delta}_{t-1}^\top](\mathbf{A}_w^{-1})^\top\right] \\ &= -2\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\delta}_{t-1}^\top(\mathbf{A}_w^{-1})^\top\mathbf{A}_w^{-1}\boldsymbol{\Sigma}_w(\mathbf{A}_w^{-1})^\top\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|Z_{t-1}]\right] \\ &\quad +2\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\delta}_{t-1}^\top(\mathbf{A}_w^{-1})^\top\mathbf{A}_w^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon_t^2|Z_{t-1}]\boldsymbol{w}_{t-1}\right] \\ &= 2\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\delta}_{t-1}^\top(\mathbf{A}_w^{-1})^\top\mathbf{A}_w^{-1}\left\{\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon_t^2|s_{t-1}]\boldsymbol{w}_{t-1} - \boldsymbol{\Sigma}_w(\mathbf{A}_w^{-1})^\top\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]\right\}\right].\end{aligned}$$

By applying the condition that the deviation becomes $\mathbf{0}$ for any function $\boldsymbol{\delta}_{t-1}(Z_{t-1},\boldsymbol{\theta}^*)$, the optimal weight function is obtained as

$$\boldsymbol{w}^*_{t-1} = \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}[(\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}]^{-1}\boldsymbol{\Sigma}_w(\mathbf{A}_w^{-1})^\top\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}].$$

Because any estimating function is invariant to transformation applied by any regular matrix, the optimal estimating function is restricted as

$$\boldsymbol{w}^*_{t-1} = \boldsymbol{w}^*_{t-1}(Z_{t-1},\boldsymbol{\theta}^*) = \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}\left[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}\right]^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}],$$

or its transformation applied by any regular matrix.

Now, we confirm that the estimator obtained by Equation (18) yields the minimum asymptotic variance. Substituting $\boldsymbol{w}^*_{t-1}$ to the matrix $\mathbf{A}_w$, some calculations in Appendix H lead us to

$$\mathbf{A}_{w^*} = \boldsymbol{\Sigma}_{w^*} = \mathbf{Q},$$

where

$$\mathbf{Q} \equiv \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}[\varepsilon_t^2|s_t]^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]^\top\right].$$

We consider how much the asymptotic variance Av changes when we make a small change $\boldsymbol{\delta}_{t-1} \equiv h\boldsymbol{\delta}_{t-1}$ on $\boldsymbol{w}^*_{t-1}$. The matrices at $\boldsymbol{w}^*_{t-1}+\boldsymbol{\delta}_{t-1}$ become

$$\begin{aligned}\mathbf{A}_{w^*+\delta} &= \mathbf{Q} + \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}[\boldsymbol{\delta}_{t-1}\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)^\top],\\ \boldsymbol{\Sigma}_{w^*+\delta} &= \mathbf{Q} + \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\boldsymbol{\delta}_{t-1}^\top\right]\\ &\quad + \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top\right] + \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\varepsilon_t^2\boldsymbol{\delta}_{t-1}\boldsymbol{\delta}_{t-1}^\top\right].\end{aligned}$$

Therefore,

$$\begin{aligned}&\mathbf{A}^{-1}_{w^*+\delta}\boldsymbol{\Sigma}_{w^*+\delta}\left(\mathbf{A}^{-1}_{w^*+\delta}\right)^\top - \mathbf{A}^{-1}_{w^*}\boldsymbol{\Sigma}_{w^*}\left(\mathbf{A}^{-1}_{w^*}\right)^\top\\ &= \mathbf{A}^{-1}_{w^*+\delta}\underbrace{\left(\boldsymbol{\Sigma}_{w^*+\delta} - \mathbf{A}_{w^*+\delta}\mathbf{A}^{-1}_{w^*}\boldsymbol{\Sigma}_{w^*}(\mathbf{A}^{-1}_{w^*})^\top\mathbf{A}^\top_{w^*+\delta}\right)}_{\mathbf{C}_1}\left(\mathbf{A}^{-1}_{w^*+\delta}\right)^\top.\end{aligned}$$

The matrix $\mathbf{C}_1$ is a semipositive definite matrix, because

$$\begin{aligned}\mathbf{C}_1 &= \boldsymbol{\Sigma}_{w^*+\delta} - \mathbf{A}_{w^*+\delta}\mathbf{A}^{-1}_{w^*}\boldsymbol{\Sigma}_{w^*}(\mathbf{A}^{-1}_{w^*})^\top\mathbf{A}^\top_{w^*+\delta}\\ &= \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\varepsilon_t^2\boldsymbol{\delta}_{t-1}\boldsymbol{\delta}_{t-1}^\top\right] - \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top\right]\mathbf{Q}^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\boldsymbol{\delta}_{t-1}^\top\right]\\ &= \lim_{t\to\infty}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\left(\boldsymbol{\delta}_{t-1}-\boldsymbol{\nu}_{t-1}\right)\left(\boldsymbol{\delta}_{t-1}-\boldsymbol{\nu}_{t-1}\right)^\top\right] \succeq \mathbf{0},\end{aligned}$$

where

$$\boldsymbol{\nu}_{t-1} \equiv \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}[\varepsilon_t^2|s_{t-1}]^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\delta}_{t-1}\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)\}^\top\right]\mathbf{Q}^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*_s}\left[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}\right].$$

Thus we have

$$\mathbf{A}^{-1}_{w^*+\delta}\boldsymbol{\Sigma}_{w^*+\delta}\left(\mathbf{A}^{-1}_{w^*+\delta}\right)^\top - \mathbf{A}^{-1}_{w^*}\boldsymbol{\Sigma}_{w^*}\left(\mathbf{A}^{-1}_{w^*}\right)^\top \succeq \mathbf{0},$$

where $\succeq$ denotes the semipositive definiteness of the subtraction. The equality in the above equation holds only when $\boldsymbol{\delta}_{t-1} \propto \boldsymbol{w}^*_{t-1}$. ■

Appendix H. Proof of Lemma 7

Proof The matrix $\mathbf{A}$ in the asymptotic variance given by Equation (16) can be calculated as

$$\begin{aligned}\mathbf{A} &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}} \boldsymbol{\psi}_t^*(z_t,\boldsymbol{\theta}^*)\right] \\ &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{w}_{t-1}^*(Z_{t-1},\boldsymbol{\theta}^*)\left\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta})\right\}^\top\right] \\ &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\Big[\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}]^{-1} \\ &\qquad \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]^\top\Big].\end{aligned}$$

Also the matrix $\boldsymbol{\Sigma}$ can be calculated as

$$\begin{aligned}\boldsymbol{\Sigma} &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\psi}_t^*(z_t,\boldsymbol{\theta}^*)\boldsymbol{\psi}_t^*(z_t,\boldsymbol{\theta}^*)^\top\right] \\ &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}\right]\boldsymbol{w}_{t-1}^*(Z_{t-1},\boldsymbol{\theta}^*)\{\boldsymbol{w}_{t-1}^*(Z_{t-1},\boldsymbol{\theta}^*)\}^\top\right] \\ &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\Big[\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}\right] \\ &\qquad \left\{\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}\right]^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]\right\} \\ &\qquad \left\{\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}\right]^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]\right\}^\top\Big] \\ &= \lim_{t\to\infty} \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\Big[\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\varepsilon(z_t,\boldsymbol{\theta}^*)^2|s_{t-1}]^{-1} \\ &\qquad \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta}^*)|s_{t-1}]^\top\Big] = \mathbf{A} = \mathbf{Q}.\end{aligned}$$

These observations yield

$$\mathrm{Av}(\hat{\boldsymbol{\theta}}) = \frac{1}{T}\mathbf{A}^{-1}\boldsymbol{\Sigma}(\mathbf{A}^{-1})^\top = \frac{1}{T}\mathbf{Q}^{-1}.$$

■

Appendix I. Proof of Theorem 8

Proof To simplify the following proof, we assume the true parameter is located on the origin without loss of generality: $\boldsymbol{\theta}^* = \mathbf{0}$. Let h_t be $\|\hat{\boldsymbol{\theta}}_t\|^2$. The conditional expectation of variation of h_t can be derived as

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}[h_{t+1}-h_t|s_t] =& -2\eta_{t+1}\hat{\boldsymbol{\theta}}_t^\top\mathbf{R}(\hat{\boldsymbol{\theta}}_t)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)\middle|s_t\right] \\ &+\eta_{t+1}^2\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\|\mathbf{R}(\hat{\boldsymbol{\theta}}_t)\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)\|^2\middle|s_t\right].\end{aligned}$$

From Assumption 5, the second term of this equation is bounded by the second moment, thus we obtain

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[h_{t+1}-(1+\eta_{t+1}^2c_2)h_t|s_t\right] \\ &\le -2\eta_{t+1}\hat{\boldsymbol{\theta}}_t^\top\mathbf{R}(\hat{\boldsymbol{\theta}}_t)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)\middle|s_t\right]+\eta_{t+1}^2c_1.\end{aligned} \tag{42}$$

Now, let $\chi_t = \prod_{k=1}^{t} 1/(1+\eta_k^2 c_2)$ and $h'_t = \chi_t h_t$. From the assumption $\sum_{t=1}^{\infty} \eta_t^2 < \infty$, we easily verify that $0 < \chi_t < 1$. Multiplying the both sides of Equation (42) by χ_{t+1}, we obtain

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[h'_{t+1} - h'_t | Z_t\right] \\ &\leq -2\eta_{t+1}\chi_{t+1}\hat{\boldsymbol{\theta}}_t^\top \mathbf{R}(\hat{\boldsymbol{\theta}}_t)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)\middle| s_t\right] + \eta_{t+1}^2\chi_{t+1}c_1.\end{aligned}$$

The first term of this upper bound is negative because of Assumption 5, the second term is non-negative because η_t, χ_{t+1}, and c_1 are nonnegative, and the sum of the second terms $\sum_{t=1}^{\infty}\eta_t^2\chi_{t+1}c_1$ is finite. Then, the supermartingale convergence theorem (Neveu, 1975; Bertsekas and Tsitsiklis, 1996, Proposition 4.2) guarantees that h'_t converges to a nonnegative random variable almost surely, and $\sum_{t=1}^{\infty}\eta_{t+1}\chi_{t+1}\hat{\boldsymbol{\theta}}_t^\top \mathbf{R}_t(\hat{\boldsymbol{\theta}}_t)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)\middle| s_t\right] < \infty$. Since $\sum_{t=1}^{\infty}\eta_t = \infty$ and $\lim_{t\to\infty}\chi_t = \chi_\infty > 0$, we have $\hat{\boldsymbol{\theta}}_t^\top \mathbf{R}(\hat{\boldsymbol{\theta}}_t)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{\psi}_{t+1}(Z_{t+1},\hat{\boldsymbol{\theta}}_t)\middle| s_t\right] \xrightarrow{a.s.} \mathbf{0}$. This result suggests the conclusion that the on-line learning algorithm converges to the true parameter almost surely: $\hat{\boldsymbol{\theta}}_t \xrightarrow{a.s.} \boldsymbol{\theta}^* = \mathbf{0}$. ■

Appendix J. Proof of Lemma 9

Proof Using Taylor series expansion of the estimating equation $(1/t)\sum_{i=1}^{t}\boldsymbol{\psi}_i(Z_i,\tilde{\boldsymbol{\theta}}_t)$ around $\tilde{\boldsymbol{\theta}}_{t-1}$, we obtain

$$\begin{aligned}\frac{1}{t}\sum_{i=1}^{t}\boldsymbol{\psi}_i(Z_i,\tilde{\boldsymbol{\theta}}_t) =& \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{\psi}_i(Z_i,\tilde{\boldsymbol{\theta}}_{t-1}) \\ &+\frac{1}{t}\sum_{i=1}^{t}\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_i(Z_i,\tilde{\boldsymbol{\theta}}_{t-1})(\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}) + O_p\left(\|\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}\|^2\right).\end{aligned}$$

Since $\sum_{i=1}^{t}\boldsymbol{\psi}_i(Z_i,\tilde{\boldsymbol{\theta}}_t) = \sum_{i=1}^{t-1}\boldsymbol{\psi}_i(Z_i,\tilde{\boldsymbol{\theta}}_{t-1}) = \mathbf{0}$, we obtain the following equation:

$$-\frac{1}{t}\boldsymbol{\psi}_t(Z_t,\tilde{\boldsymbol{\theta}}_{t-1}) = \tilde{\mathbf{R}}_t(\tilde{\boldsymbol{\theta}}_{t-1})(\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}) + O_p\left(\|\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}\|^2\right).$$

We can then rewrite the right hand side as

$$-\frac{1}{t}\boldsymbol{\psi}_t(Z_t,\tilde{\boldsymbol{\theta}}_{t-1}) = \{\tilde{\mathbf{R}}_t(\tilde{\boldsymbol{\theta}}_{t-1}) + O_p(\|\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}\|)\}(\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}),$$

and

$$(\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}) = -\frac{1}{t}\{\tilde{\mathbf{R}}_t^{-1}(\tilde{\boldsymbol{\theta}}_{t-1}) + O_p(\|\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}\|)\}\boldsymbol{\psi}_t(Z_t,\tilde{\boldsymbol{\theta}}_{t-1}).$$

Note that $\tilde{\mathbf{R}}_t^{-1}(\tilde{\boldsymbol{\theta}}_{t-1})$ is uniformly bounded because of the nonsingular condition in Lemma 9. Also $\tilde{\boldsymbol{\theta}}_t$ is uniformly bounded for any t. Furthermore, $\boldsymbol{\psi}_t(Z_t,\tilde{\boldsymbol{\theta}}_{t-1})$ is uniformly bounded for any t since the conditions in Assumptions 2-4 imply that $\boldsymbol{\psi}_t(Z_t,\tilde{\boldsymbol{\theta}}_{t-1})$ is a continuous function of uniformly bounded variables. Hence, the above equation implies that $\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1} = O_p(1/t)$. Therefore, we can obtain the following equation

$$-\frac{1}{t}\boldsymbol{\psi}_t(Z_t,\tilde{\boldsymbol{\theta}}_{t-1}) = \tilde{\mathbf{R}}_t(\tilde{\boldsymbol{\theta}}_{t-1})(\tilde{\boldsymbol{\theta}}_t-\tilde{\boldsymbol{\theta}}_{t-1}) + O_p\left(\frac{1}{t^2}\right).$$

By using the matrix inversion operation, we derive

$$\tilde{\boldsymbol{\theta}}_t = \tilde{\boldsymbol{\theta}}_{t-1} - \frac{1}{t}\tilde{\mathbf{R}}_t^{-1}\psi_t(Z_t, \tilde{\boldsymbol{\theta}}_{t-1}) + O_p\left(\frac{1}{t^2}\right).$$

■

Appendix K. Proof of Theorem 10

Proof Similar to the proof in Appendix I, we assume the true parameter is located at the origin: $\boldsymbol{\theta}^* = \mathbf{0}$. From the assumption in Theorem 10, the online learning converges to the true parameter almost surely; this implies that $\hat{\boldsymbol{\theta}}_t = \boldsymbol{\theta}^* + o_p(1) = o_p(1)$. Note also that $\mathbf{R}_t$ converges to $\mathbf{A}$ almost surely; this implies that $\mathbf{R}_t = \mathbf{A} + o_p(1)$. Furthermore, from condition (d) in Theorem 10, the matrix $\mathbf{R}_t$ is invertible for any t; this implies that $\mathbf{R}_t^{-1} = \mathbf{A}^{-1} + o_p(1)$.

Using Equation (23), $(\hat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^*)(\hat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^*)^\top = \hat{\boldsymbol{\theta}}_t\hat{\boldsymbol{\theta}}_t^\top$ can be expressed as

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_t\hat{\boldsymbol{\theta}}_t^\top = & \left(\hat{\boldsymbol{\theta}}_{t-1} - \frac{1}{t}\hat{\mathbf{R}}_t^{-1}\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1}) + O_p\left(\frac{1}{t^2}\right)\right) \\
& \left(\hat{\boldsymbol{\theta}}_{t-1} - \frac{1}{t}\hat{\mathbf{R}}_t^{-1}\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1}) + O_p\left(\frac{1}{t^2}\right)\right)^\top \\
= & \hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top - \frac{1}{t}\hat{\boldsymbol{\theta}}_{t-1}\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1})^\top(\hat{\mathbf{R}}_t^{-1})^\top - \frac{1}{t}\hat{\mathbf{R}}_t^{-1}\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1})\hat{\boldsymbol{\theta}}_{t-1}^\top \\
& + \frac{1}{t^2}\hat{\mathbf{R}}_t^{-1}\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1})\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1})^\top(\hat{\mathbf{R}}_t^{-1})^\top + o_p\left(\frac{1}{t^2}\right),
\end{aligned}$$

where high order terms are in total represented as $o_p\left(1/t^2\right)$ because of $\hat{\boldsymbol{\theta}}_{t-1}O_p(1/t^2) = o_p(1)O_p(1/t^2) = o_p(1/t^2)$. Taking the conditional expectation of $\hat{\boldsymbol{\theta}}_t\hat{\boldsymbol{\theta}}_t^\top$ given Z_{t-1}, we obtain

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\hat{\boldsymbol{\theta}}_t\hat{\boldsymbol{\theta}}_t^\top | Z_{t-1}\right] = & \hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top - \frac{1}{t}\underbrace{\hat{\boldsymbol{\theta}}_{t-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1})^\top(\hat{\mathbf{R}}_t^{-1})^\top \middle| Z_{t-1}\right]}_{\mathbf{C}_1^\top} \\
& - \frac{1}{t}\underbrace{\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\hat{\mathbf{R}}_t^{-1}\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1}) \middle| Z_{t-1}\right]\hat{\boldsymbol{\theta}}_{t-1}^\top}_{\mathbf{C}_1} \\
& + \frac{1}{t^2}\underbrace{\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\hat{\mathbf{R}}_t^{-1}\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1})\psi_t(Z_t, \hat{\boldsymbol{\theta}}_{t-1})^\top(\hat{\mathbf{R}}_t^{-1})^\top \middle| Z_{t-1}\right]}_{\mathbf{C}_2} + o_p\left(\frac{1}{t^2}\right).
\end{aligned}$$

We now express each of the terms in the above equation.

In order to express $\mathbf{C}_1$ and $\mathbf{C}_2$, we introduce the following lemma.

Lemma 21 *(Bottou and LeCun, 2005, Theorem 4) Let X_t be a uniformly bounded random variable depending on Z_t. Then we have*

$$\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\hat{\mathbf{R}}_t^{-1}X_t \middle| Z_{t-1}\right] = \mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\hat{\mathbf{R}}_t^{-1} \middle| Z_{t-1}\right]\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[X_t | Z_{t-1}\right] + o_p\left(\frac{1}{t}\right).$$

Proof By using assumption (b) in Theorem 10, $\mathbb{E}_{\theta^*,\xi_s^*}[\hat{\mathbf{R}}_t^{-1}X_t|Z_{t-1}]$ can be calculated as

$$\mathbb{E}_{\theta^*,\xi_s^*}\left[\hat{\mathbf{R}}_t^{-1}X_t|Z_{t-1}\right] = \mathbb{E}_{\theta^*,\xi_s^*}[\hat{\mathbf{R}}_t^{-1}|Z_{t-1}]\mathbb{E}_{\theta^*,\xi_s^*}\left[X_t|Z_{t-1}\right] + \mathbb{E}_{\theta^*,\xi_s^*}\left[\varepsilon_t(Z_t)X_t|Z_{t-1}\right],$$

where $\varepsilon_t(Z_t) = o_p(1/t)$. $\mathbb{E}_{\theta^*,\xi_s^*}[\varepsilon_t(Z_t)X_t|Z_{t-1}]$ is summarized as $o_p(1/t)$ because of the Cauchy-Schwartz's inequality:

$$\mathbb{E}_{\theta^*,\xi_s^*}[\varepsilon_t(Z_t)X_t|Z_{t-1}] \le \sqrt{\mathbb{E}_{\theta^*,\xi_s^*}[\,\|\varepsilon_t(Z_t)\|^2|Z_{t-1}]}\sqrt{\mathbb{E}_{\theta^*,\xi_s^*}[\,\|X_t\|^2|Z_{t-1}]}.$$

■

Since condition (a) in Theorem 10 and Assumptions 2-4 lead $\psi_t(Z_t,\hat{\theta}_{t-1})$ and $\hat{\mathbf{R}}_t$ to be continuous functions of uniformly bounded variables, $\psi_t(Z_t,\hat{\theta}_{t-1})$ and $\hat{\mathbf{R}}_t$ are uniformly bounded for any t. Then, using Lemma 21, $\mathbf{C}_2$ can be expressed as

$$\mathbf{C}_2 = \Big\{\mathbb{E}_{\theta^*,\xi_s^*}\left[\hat{\mathbf{R}}_t^{-1}\middle|Z_{t-1}\right] \mathbb{E}_{\theta^*,\xi_s^*}\left[\psi_t(Z_t,\hat{\theta}_{t-1})\psi_t(Z_t,\hat{\theta}_{t-1})^\top\middle|Z_{t-1}\right]\mathbb{E}_{\theta^*,\xi_s^*}\left[(\hat{\mathbf{R}}_t^{-1})^\top\middle|Z_{t-1}\right]\Big\} + o_p\left(\frac{1}{t}\right).$$

We note that $\mathbb{E}_{\theta^*,\xi_s^*}[\psi_t(Z_t,\hat{\theta}_{t-1})\psi_t(Z_t,\hat{\theta}_{t-1})^\top|Z_{t-1}]$ can be calculated as

$$\mathbb{E}_{\theta^*,\xi_s^*}\left[\psi_t(Z_t,\hat{\theta}_{t-1})\psi_t(Z_t,\hat{\theta}_{t-1})^\top\middle|Z_{t-1}\right] = \mathbb{E}_{\theta^*,\xi_s^*}\left[\psi_t(Z_t,\theta^*)\psi_t(Z_t,\theta^*)^\top\middle|Z_{t-1}\right] + o_p(1),$$

because $\hat{\theta}_{t-1}$ converges to the true parameter and $\psi_t(Z_t,\theta)$ is uniformly bounded. Since $\hat{\mathbf{R}}_t^{-1} = \mathbf{A}^{-1} + o_p(1)$ is satisfied, $\mathbf{C}_2$ can be rewritten as

$$\mathbf{C}_2 = \mathbf{A}^{-1}\mathbb{E}_{\theta^*,\xi_s^*}\left[\psi_t(Z_t,\theta^*)\psi_t(Z_t,\theta^*)^\top\middle|Z_{t-1}\right](\mathbf{A}^{-1})^\top + o_p(1). \tag{43}$$

Using similar arguments, $\mathbf{C}_1$ can be expressed as

$$\begin{aligned}\mathbf{C}_1 &= \mathbb{E}_{\theta^*,\xi_s^*}\left[\hat{\mathbf{R}}_t^{-1}\psi_t(Z_t,\hat{\theta}_{t-1})\middle|Z_{t-1}\right]\hat{\theta}_{t-1}\\ &= \mathbb{E}_{\theta^*,\xi_s^*}\left[\hat{\mathbf{R}}_t^{-1}\middle|Z_{t-1}\right]\mathbb{E}_{\theta^*,\xi_s^*}\left[\psi_t(Z_t,\hat{\theta}_{t-1})|Z_{t-1}\right]\hat{\theta}_{t-1}^\top + o_p\left(\frac{1}{t}\right).\end{aligned}$$

We now consider $\mathbb{E}_{\theta^*,\xi_s^*}[\psi_t(Z_t,\hat{\theta}_{t-1})|Z_{t-1}]$. Applying a Taylor series expansion to $\mathbb{E}_{\theta^*,\xi_s^*}[\psi_t(Z_t,\hat{\theta}_{t-1})|Z_{t-1}]$ around the true parameter $\theta^* = \mathbf{0}$, we have

$$\begin{aligned}&\mathbb{E}_{\theta^*,\xi_s^*}\left[\psi_t(Z_t,\hat{\theta}_{t-1})\middle|Z_{t-1}\right]\\ &= \mathbb{E}_{\theta^*,\xi_s^*}\left[\psi_t(Z_t,\theta^*)|Z_{t-1}\right] + \mathbb{E}_{\theta^*,\xi_s^*}\left[\partial_\theta\psi_t(Z_t,\theta^*)|Z_{t-1}\right]\hat{\theta}_{t-1} + o_p\left(|\hat{\theta}_{t-1}|\right)\\ &= \mathbb{E}_{\theta^*,\xi_s^*}\left[\partial_\theta\psi_t(Z_t,\theta^*)|Z_{t-1}\right]\hat{\theta}_{t-1} + o_p\left(|\hat{\theta}_{t-1}|\right),\end{aligned}$$

where we have used the fact that $\mathbb{E}_{\theta^*,\xi_s^*}[\psi_t(Z_t,\theta^*)|Z_{t-1}]$ is zero. Since $\hat{\mathbf{R}}_t = \mathbf{A} + o_p(1)$ is satisfied, $\mathbf{C}_1$ can be rewritten as

$$\begin{aligned}\mathbf{C}_1 &= \left(\mathbf{A}^{-1} + o_p(1)\right)\left(\mathbb{E}_{\theta^*,\xi_s^*}\left[\partial_\theta\psi_t(Z_t,\theta^*)|Z_{t-1}\right]\hat{\theta}_{t-1} + o_p\left(|\hat{\theta}_{t-1}|\right)\right)\hat{\theta}_{t-1}^\top + o_p\left(\frac{1}{t}\right)\\ &= \mathbf{A}^{-1}\mathbb{E}_{\theta^*,\xi_s^*}\left[\partial_\theta\psi_t(Z_t,\theta^*)|Z_{t-1}\right]\hat{\theta}_{t-1}\hat{\theta}_{t-1}^\top + o_p\left(\|\hat{\theta}_{t-1}\|^2\right) + o_p\left(\frac{1}{t}\right).\end{aligned} \tag{44}$$

We now use Equations (43) and (44), leading to

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\hat{\boldsymbol{\theta}}_t\hat{\boldsymbol{\theta}}_t^\top|Z_{t-1}\right] &= \left(1+o_p\left(\frac{1}{t}\right)\right)\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top - \frac{1}{t}\mathbf{A}^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)|Z_{t-1}\right]\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top \\ &\quad - \frac{1}{t}\left(\mathbf{A}^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)|Z_{t-1}\right]\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top\right)^\top \\ &\quad + \frac{1}{t^2}\mathbf{A}^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}_s^*}\left[\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)^\top|Z_{t-1}\right](\mathbf{A}^{-1})^\top + o_p\left(\frac{1}{t^2}\right).\end{aligned}$$

Taking the expectation over the sequence, we obtain

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\hat{\boldsymbol{\theta}}_t\hat{\boldsymbol{\theta}}_t^\top\right] &= \left(1+o\left(\frac{1}{t}\right)\right)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top\right] - \frac{1}{t}\mathbf{A}^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top\right] \\ &\quad - \frac{1}{t}\left(\mathbf{A}^{-1}\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)\hat{\boldsymbol{\theta}}_{t-1}\hat{\boldsymbol{\theta}}_{t-1}^\top\right]\right)^\top + \frac{1}{t^2}\mathbf{A}^{-1}\boldsymbol{\Sigma}(\mathbf{A}^{-1})^\top + o\left(\frac{1}{t^2}\right),\end{aligned}$$

where we have used the fact that $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)^\top\right]$ converges to $\boldsymbol{\Sigma}$: $\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}^*)^\top\right] = \boldsymbol{\Sigma} + o(1)$. Using assumption (c) in Theorem 10 and applying the trace operator, we obtain

$$\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\|\hat{\boldsymbol{\theta}}_t\|^2\right] = \left(1-\frac{2}{t}+o\left(\frac{1}{t}\right)\right)\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\|\hat{\boldsymbol{\theta}}_{t-1}\|^2\right] + \frac{1}{t^2}\mathrm{tr}\left\{\mathbf{A}^{-1}\boldsymbol{\Sigma}(\mathbf{A}^{-1})^\top\right\} + o\left(\frac{1}{t^2}\right).$$

We now introduce the following lemma.

Lemma 22 *(Bottou and LeCun, 2005, Lemma 1) Let $\{u_t\}$ be a positive sequence defined as*

$$u_t = \left(1-\frac{\alpha}{t}+o\left(\frac{1}{t}\right)\right)u_{t-1} + \frac{\beta}{t^2} + o\left(\frac{1}{t^2}\right).$$

If $\alpha > 1$ and $\beta > 0$ hold, then

$$tu_t \to \frac{\beta}{\alpha - 1}.$$

The proof is given in Lemma 1 in Bottou and LeCun (2005). Referring the result of Lemma 22, we have

$$\mathbb{E}_{\boldsymbol{\theta}^*,\boldsymbol{\xi}^*}\left[\|\hat{\boldsymbol{\theta}}_t\|^2\right] = \frac{1}{t}\mathrm{tr}\left\{\mathbf{A}^{-1}\boldsymbol{\Sigma}(\mathbf{A}^{-1})^\top\right\} + o\left(\frac{1}{t}\right).$$

■

Appendix L. Proof of Lemma 11

Proof Since the MRPs defined in Section 2 are ergodic, the MRPs satisfy geometrically uniform mixing. By performing a Taylor series expansion to estimating Equation (15) around the parameter $\boldsymbol{\theta}$, we obtain

$$\mathbf{0} = \sum_{t=1}^{T}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}) + \sum_{t=1}^{T}\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta})(\hat{\boldsymbol{\theta}}_T - \boldsymbol{\theta}) + O\left(\left\|\hat{\boldsymbol{\theta}}_T - \boldsymbol{\theta}\right\|^2\right).$$

Here, high order terms are in total represented as $O(\|\hat{\boldsymbol{\theta}}_T - \boldsymbol{\theta}\|^2)$ because of the twice differentiable condition for the function $g(s,\boldsymbol{\theta})$ as Assumption 3. By applying the law of large numbers (ergodic pointwise theorem) (Billingsley, 1995, Theorem 24.1) to $(1/T)\sum_{t=1}^{T}\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta})$, we have

$$\frac{1}{T}\sum_{t=1}^{T}\partial_{\boldsymbol{\theta}}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{w}_{t-1}(Z_{t-1})\{\partial_{\boldsymbol{\theta}}\varepsilon(z_t,\boldsymbol{\theta})\}^\top \xrightarrow{a.s.} \underbrace{\lim_{t\to\infty}\mathbb{E}\left[\boldsymbol{w}_{t-1}(Z_{t-1})\partial_{\boldsymbol{\theta}}\{\varepsilon(z_t,\boldsymbol{\theta})\}^\top\right]}_{=\mathbf{A}}.$$

Let $\boldsymbol{k}\in\mathbb{R}^m$ be any nonzero vector. By applying the central limit theorem in Lemma 18 to $(1/\sqrt{T})\sum_{t=1}^{T}\boldsymbol{k}^\top\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta})$, we have

$$\frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{k}^\top\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta}) = \frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{k}^\top\boldsymbol{w}_{t-1}(Z_{t-1})\varepsilon_t(z_t,\boldsymbol{\theta})$$

$$\xrightarrow{d}\mathcal{N}\left(0,\boldsymbol{k}\underbrace{\left(\lim_{t\to\infty}\mathbb{E}\left[\varepsilon_t(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]+\lim_{t\to\infty}2\sum_{t'=1}^{\infty}\mathrm{cov}\left[\varepsilon(z_t,\boldsymbol{\theta})\boldsymbol{w}_{t-1},\varepsilon(z_{t+t'},\boldsymbol{\theta})\boldsymbol{w}_{t+t'-1}\right]\right)}_{\boldsymbol{\Sigma}}\boldsymbol{k}\right).$$

Therefore, from the Cramér-Wold theorem (van der Vaart, 2000), $(1/\sqrt{T})\sum_{t=1}^{T}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta})$ converges to a Gaussian distribution as follows;

$$\frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta})\xrightarrow{d}\mathcal{N}\left(\mathbf{0},\boldsymbol{\Sigma}\right).$$

By neglecting higher order terms, we obtain

$$\sqrt{T}(\hat{\boldsymbol{\theta}}_T-\boldsymbol{\theta})\sim\mathcal{N}\left(\mathbf{0},\mathbf{A}^{-1}\boldsymbol{\Sigma}(\mathbf{A}^\top)^{-1}\right).$$

Then, $\hat{\boldsymbol{\theta}}_T$ is Gaussian distributed: $\hat{\boldsymbol{\theta}}_T\sim\mathcal{N}(\boldsymbol{\theta},\widetilde{\mathrm{Av}})$, where the asymptotic variance $\widetilde{\mathrm{Av}}$ is given by Equation (32). ■

Appendix M. Proof of Lemma 12

Proof Let $\boldsymbol{k}\in\mathbb{R}^m$ be any nonzero vector. By applying the central limit theorem to $(1/\sqrt{T})\sum_{t=1}^{T}\boldsymbol{k}^\top\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta})$ in Lemma 18, we have

$$\frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{k}^\top\boldsymbol{\psi}_t(Z_t,\boldsymbol{\theta})\xrightarrow{d}\mathcal{N}\left(0,\boldsymbol{k}^\top\boldsymbol{\Sigma}\boldsymbol{k}\right),$$

where

$$\boldsymbol{k}^\top\boldsymbol{\Sigma}\boldsymbol{k}=\lim_{t\to\infty}\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k}$$
$$+\lim_{t\to\infty}2\sum_{t'=1}^{\infty}\mathrm{cov}\left[\varepsilon(z_t,\boldsymbol{\theta})\boldsymbol{k}^\top\boldsymbol{w}_{t-1},\varepsilon(z_{t+t'},\boldsymbol{\theta})\boldsymbol{k}^\top\boldsymbol{w}_{t+t'-1}\right].$$

Since the target process is a geometrically uniform mixing, there exist some positive constants C and $\rho \in [0,1)$ such that $\varphi(t) \leq C\rho^t$. Then, by using the covariance bound in Lemma 19, we obtain

$$\begin{aligned}\left|\operatorname{cov}\left[\varepsilon(z_t,\boldsymbol{\theta})\boldsymbol{k}^\top \boldsymbol{w}_{t-1}, \varepsilon(z_{t+t'},\boldsymbol{\theta})\boldsymbol{k}^\top \boldsymbol{w}_{t+t'-1}\right]\right| &\leq 2\sqrt{\varphi(t')}\lim_{t\to\infty}\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k} \\ &\leq 2\sqrt{C}\rho^{t'/2}\lim_{t\to\infty}\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k}.\end{aligned}$$

Therefore, $\boldsymbol{k}^\top\boldsymbol{\Sigma}\boldsymbol{k}$ is bounded as

$$\begin{aligned}&\left|\boldsymbol{k}^\top\boldsymbol{\Sigma}\boldsymbol{k}\right| \\ &= \left|\lim_{t\to\infty}\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k} + 2\lim_{t\to\infty}\sum_{t'=1}^{\infty}\operatorname{cov}\left[\varepsilon(z_t,\boldsymbol{\theta})\boldsymbol{k}^\top\boldsymbol{w}_{t-1}, \varepsilon(z_{t+t'},\boldsymbol{\theta})\boldsymbol{k}^\top\boldsymbol{w}_{t+t'-1}\right]\right| \\ &\leq \lim_{t\to\infty}\left|\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k}\right| + 2\lim_{t\to\infty}\sum_{t'=1}^{\infty}\left|\operatorname{cov}\left[\varepsilon(z_t,\boldsymbol{\theta})\boldsymbol{k}^\top\boldsymbol{w}_{t-1}, \varepsilon(z_{t+t'},\boldsymbol{\theta})\boldsymbol{k}^\top\boldsymbol{w}_{t+t'-1}\right]\right| \\ &\leq \lim_{t\to\infty}\left|\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k}\right| + 4\sqrt{C}\sum_{t'=1}^{\infty}\rho^{t'/2}\lim_{t\to\infty}\left|\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k}\right| \\ &= \lim_{t\to\infty}\left|\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k}\right|\left(1+4\sqrt{C}\sum_{t'=1}^{\infty}\rho^{t'/2}\right) \\ &= \lim_{t\to\infty}\left|\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k}\right|\left(1+4\sqrt{C}\frac{\rho^{1/2}}{(1-\rho^{1/2})}\right) \\ &= \Upsilon\lim_{t\to\infty}\left|\boldsymbol{k}^\top\mathbb{E}\left[\varepsilon(z_t,\boldsymbol{\theta})^2\boldsymbol{w}_{t-1}\boldsymbol{w}_{t-1}^\top\right]\boldsymbol{k}\right| = \Upsilon\left|\boldsymbol{k}^\top\boldsymbol{\Sigma}_0\boldsymbol{k}\right|,\end{aligned}$$

where $\Upsilon = 1+4\sqrt{C}\rho^{1/2}/(1-\rho^{1/2})$. Thus, we can obtain the following relation;

$$\boldsymbol{k}^\top\left(\Upsilon\boldsymbol{\Sigma}_0-\boldsymbol{\Sigma}\right)\boldsymbol{k} \geq 0.$$

This implies that $\Upsilon\boldsymbol{\Sigma}_0-\boldsymbol{\Sigma}$ is a semipositive definite matrix, hence we derive

$$\frac{1}{T}\mathbf{A}^{-1}\boldsymbol{\Sigma}\left(\mathbf{A}^{-1}\right)^\top \preceq \frac{\Upsilon}{T}\mathbf{A}^{-1}\boldsymbol{\Sigma}_0\left(\mathbf{A}^{-1}\right)^\top.$$

■

References

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

S. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

S. Amari and J. F. Cardoso. Blind source separation-semiparametric statistical approach. *IEEE Transactions on Signal Processing*, 45(11):2692–2700, 2002.

S. Amari and M. Kawanabe. Information geometry of estimating functions in semi-parametric statistical models. *Bernoulli*, 3(1):29–54, 1997.

S. Amari, H. Park, and K. Fukumizu. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12(6):1399–1409, 2000.

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning*, pages 30–37, 1995.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

P. J. Bickel, C. A. Klaassen, Y. Ritov, and J. A. Wellner. *Efficient and Adaptive Estimation for Semiparametric Models*. Springer, 1998.

P. Billingsley. The Lindeberg-Levy theorem for martingales. *Proceedings of the American Mathematical Society*, 12(5):788–792, 1961.

P. Billingsley. *Probability and Measure*. John Wiley and Sons, 1995.

L. Bottou and Y. LeCun. Large scale online learning. In *Advances in Neural Information Processing Systems 16*, 2004.

L. Bottou and Y. LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(4):137–151, 2005.

J. A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49 (2):233–246, 2002.

R. C. Bradley. Basic properties of strong mixing conditions. A survey and some open questions. *Probability Surveys*, 2:107–144, 2005.

S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1):33–57, 1996.

R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*, pages 1017–1023, 1996.

A. Geramifard, M. Bowling, and R. S. Sutton. Incremental least-squares temporal difference learning. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 356–361. AAAI Press, 2006.

A. Geramifard, M. Bowling, M. Zinkevich, and R. S. Sutton. iLSTD: Eligibility traces and convergence analysis. In *Advances in Neural Information Processing Systems 19*, pages 441–448, 2007.

V. P. Godambe. An optimum property of regular maximum likelihood estimation. *The Annals of Mathematical Statistics*, 31(4):1208–1211, 1960.

V. P. Godambe. The foundations of finite sample estimation in stochastic processes. *Biometrika*, 72 (2):419–428, 1985.

V. P. Godambe, editor. *Estimating Functions*. Oxford University Press, 1991.

S. Grunëwälder and K. Obermayer. Optimality of LSTD and its relation to TD and MC. Technical report, Berlin University of Technology, 2006.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

P. J. Huber and E. M. Ronchetti. *Robust Statistics*. John Wiley and Sons, 2009.

I. A. Ibragimov and I. U. V. Linnik. *Independent and Stationary Sequences of Random Variables*. Wolters-Noordhoff, 1971.

M. Kawanabe and K. Müller. Estimating functions for blind separation when sources have variance dependencies. *Journal of Machine Learning Research*, 6(1):453—482, 2005.

V. R. Konda. *Actor-Critic Algorithm*. PhD thesis, Massachusetts Institute of Technology, 2002.

S. Konishi and G. Kitagawa. Generalised information criteria in model selection. *Biometrika*, 83 (4):875–890, 1996.

N. Le Roux, P. A. Manzagol, and Y. Bengio. Topmoumoute online natural gradient algorithm. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

P. Liang and M. I. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *Proceedings of the 25th International Conference on Machine Learning*, pages 584–591, 2008.

S. Mahadevan and M. Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8: 2169–2231, 2007.

S. Mannor, D. Simester, P. Sun, and J. N. Tsitsiklis. Bias and variance in value function estimation. In *Proceedings of the 21st International Conference on Machine Learning*, pages 308–322, 2004.

S. Mannor, D. Simester, P. Sun, and J. N. Tsitsiklis. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007.

N. Murata and S. Amari. Statistical analysis of learning dynamics. *Signal Processing*, 74(1):3–28, 1999.

A. Nedić and D. P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13(1):79–110, 2003.

J. Neveu. *Discrete-parameter Martingales*. Elsevier, 1975.

S. Singh and D. P. Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems 9*, pages 974–980, 1997.

S. Singh and P. Dayan. Analytical mean squared error curves for temporal difference learning. *Machine Learning*, 32(1):5–40, 1998.

M. Sørensen. On asymptotics of estimating functions. *Brazilian Journal of Probability and Statistics*, 13(2):419–428, 1999.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1): 9–44, 1988.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26thth International Conference on Machine Learning*, pages 993–1000, 2009a.

R. S. Sutton, C. Szepesvári, and R. H. Maei. A convergent $O(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems 21*, 2009b.

G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3): 58 – 68, 1995.

A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 2000.

W. Wefelmeyer. Quasi-likelihood models and optimal inference. *The Annals of Statistics*, 24(1): 405–422, 1996.

H. Yu and D. P. Bertsekas. Convergence results for some temporal difference methods based on least squares. Technical report, LIDS REPORT 2697, 2006.

W. Zhang and T. G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1114–1120, 1995.

The arules R-Package Ecosystem: Analyzing Interesting Patterns from Large Transaction Data Sets

Michael Hahsler MHAHSLER@LYLE.SMU.EDU
Sudheer Chelluboina SCHELLUBOI@LYLE.SMU.EDU
Department of Computer Science and Engineering
Southern Methodist University
Dallas, Texas 75275-0122, USA

Kurt Hornik KURT.HORNIK@WU.AC.AT
Department of Finance, Accounting and Statistics
Wirtschaftsuniversität Wien, Augasse 2-6, A-1090 Wien, Austria

Christian Buchta CHRISTIAN.BUCHTA@WU.AC.AT
Department of Cross-Border Business
Wirtschaftsuniversität Wien, Augasse 2-6, A-1090 Wien, Austria

Editor: Mikio Braun

Abstract

This paper describes the ecosystem of R add-on packages developed around the infrastructure provided by the package **arules**. The packages provide comprehensive functionality for analyzing interesting patterns including frequent itemsets, association rules, frequent sequences and for building applications like associative classification. After discussing the ecosystem's design we illustrate the ease of mining and visualizing rules with a short example.

Keywords: frequent itemsets, association rules, frequent sequences, visualization

1. Overview

Mining frequent itemsets and association rules is a popular and well researched method for discovering interesting relations between variables in large databases. Association rules are used in many applications and have become prominent as an important exploratory method for uncovering cross-selling opportunities in large retail databases.

Agrawal et al. (1993) introduced the problem of mining association rules from transaction data as follows:

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called *items*. Let $\mathcal{D} = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the *database*. Each transaction in $\mathcal{D}$ has a unique transaction ID and contains a subset of the items in I. A *rule* is defined as an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$ are called *itemsets*. On itemsets and rules several quality measures can be defined. The most important measures are support and confidence. The *support* $\text{supp}(X)$ of an itemset X is defined as the proportion of transactions in the data set which contain the itemset. Itemsets with a support which surpasses a user defined threshold σ are called *frequent itemsets*. The *confidence* of a rule is defined as $\text{conf}(X \Rightarrow Y) = \text{supp}(X \cup Y)/\text{supp}(X)$. *Association rules* are rules with $\text{supp}(X \cup Y) \geq \sigma$ and $\text{conf}(X) \geq \delta$ where σ and δ are user defined thresholds.

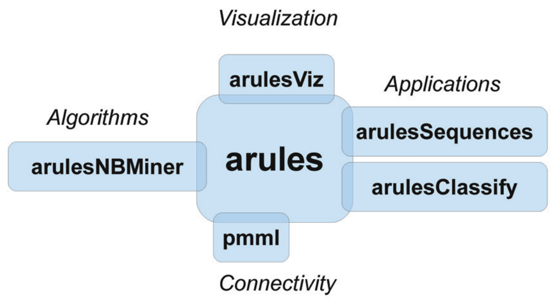


Figure 1: The **arules** ecosystem.

The R package **arules** (Hahsler et al., 2005, 2010) implements the basic infrastructure for creating and manipulating transaction databases and basic algorithms to efficiently find and analyze association rules. Over the last five years several packages were built around the **arules** infrastructure to create the ecosystem shown in Figure 1. Compared to other tools, the **arules** ecosystem is fully integrated, implements the latest approaches and has the vast functionality of R for further analysis of found patterns at its disposal.

2. Design and Implementation

The core package **arules** provides an object-oriented framework to represent transaction databases and patterns. To facilitate extensibility, patterns are implemented as an abstract superclass *associations* and then concrete subclasses implement individual types of patterns. In **arules** the associations *itemsets* and *rules* are provided. Databases and associations both use a sparse matrix representation for efficient storage and basic operations like sorting, subsetting and matching are supported. Different aspects of arules were discussed in previous publications (Hahsler et al., 2005; Hahsler and Hornik, 2007b,a; Hahsler et al., 2008).

In this paper we focus on the ecosystem of several R-packages which are built on top of the arules infrastructure. While arules provides *Apriori* and *Eclat* (implementations by Borgelt, 2003), two of the most important frequent itemset/association rule mining algorithms, additional algorithms can easily be added as new packages. For example, package **arulesNBMiner** (Hahsler, 2010) implements an algorithm to find NB-frequent itemsets (Hahsler, 2006). A collection of further implementations which could be interfaced by arules in the future and a comparison of state-of-the-art algorithms can be found at the Frequent Itemset Mining Implementations Repository.[1]

arulesSequences (Buchta and Hahsler, 2010) implements mining frequent sequences in transaction databases. It implements additional association classes called *sequences* and *sequencerules* and provides the algorithm *cSpade* (Zaki, 2001) to efficiently mine frequent sequences. Another application currently under development is **arulesClassify** which uses the arules infrastructure to implement rule-based classifiers, including *Classification Based on Association rules* (CBA, Liu et al., 1998) and general associative classification techniques (Jalali-Heravi and Zaïane, 2010).

A known drawback of mining for frequent patterns such as association rules is that typically the algorithm returns a very large set of results where only a small fraction of patterns is of interest to the analysts. Many researchers introduced visualization techniques including scatter plots, matrix

1. The Frequent Itemset Mining Implementations Repository can be found at `http://fimi.ua.ac.be/`.

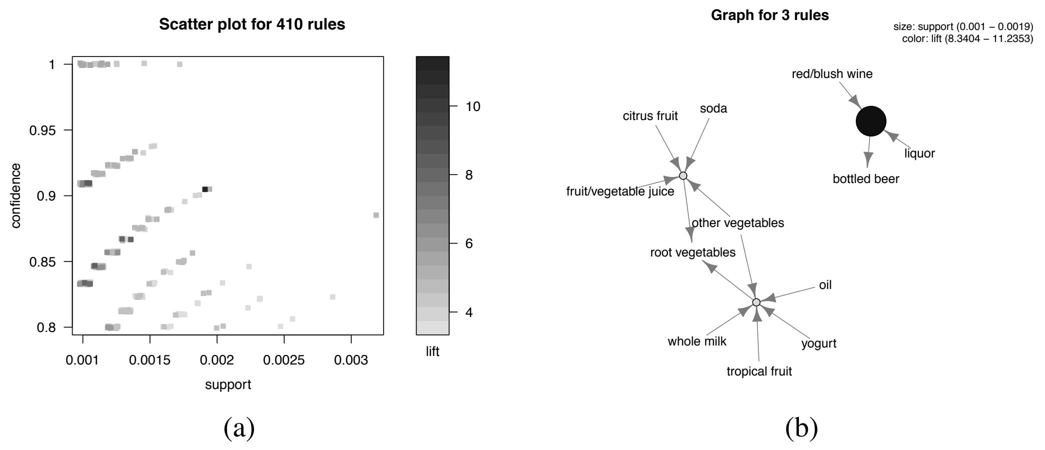


Figure 2: Visualization of all 410 rules as (a) a scatter plot and (b) shows the top 3 rules according to lift as a graph.

visualizations, graphs, mosaic plots and parallel coordinates plots to analyze large sets of association rules (see Bruzzese and Davino, 2008, for a recent overview paper). **arulesViz** (Hahsler and Chelluboina, 2010) implements most of these methods for arules while also providing improvements using color shading, reordering and interactive features.

Finally, arules provides a *Predictive Model Markup Language (PMML)* interface to import and export rules via package **pmml** (Williams et al., 2010). PMML is the leading standard for exchanging statistical and data mining models and is supported by all major solution providers. Although **pmml** provides interfaces for different packages it is still considered part of the arules ecosystem.

The packages in the described ecosystem are available for Linux, OS X and Windows. All packages are distributed via the Comprehensive R Archive Network[2] under GPL-2, along with comprehensive manuals, documentation, regression tests and source code. Development versions of most packages are available from R-Forge.[3]

3. User Interface

We illustrate the user interface and the interaction between the packages in the **arules** ecosystem with a small example using a retail data set called *Groceries* which contains 9835 transactions with items aggregated to 169 categories. We mine association rules and then present the rules found as well as the top 3 rules according to the interest measure *lift* (deviation from independence) in two visualizations.

```
> library("arules")                     ### attach package 'arules'
> library("arulesViz")                  ### attach package 'arulesViz'
> data("Groceries")                     ### load data set
> ### mine association rules
```

2. The Comprehensive R Archive Network can be found at `http://CRAN.R-project.org`.
3. R-Forge can be found at `http://R-Forge.R-project.org`.

```
> rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))
> rules
set of 410 rules

> ### visualize rules as a scatter plot (with jitter to reduce occlusion)
> plot(rules, control=list(jitter=2))
> ### select and inspect rules with highest lift
> rules_high_lift <- head(sort(rules, by="lift"), 3)
> inspect(rules_high_lift)
  lhs                          rhs              support    confidence   lift
1 {liquor, red/blush wine}
                      => {bottled beer}    0.001931876  0.9047619 11.235269
2 {citrus fruit, other vegetables, soda, fruit/vegetable juice}
                      => {root vegetables} 0.001016777  0.9090909  8.340400
3 {tropical fruit, other vegetables, whole milk, yogurt, oil}
                      => {root vegetables} 0.001016777  0.9090909  8.340400

> ### plot selected rules as graph
> plot(rules_high_lift, method="graph", control=list(type="items"))
```

Figure 2 shows the visualizations produced by the example code. Both visualizations clearly show that there exists a rule (`{liquor, red/blush wine} => {bottled beer}`) with high support, confidence and lift. With the additionally available interactive features for the scatter plot and other available plots like the grouped matrix visualization, the rule set can be further explored.

References

Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216. ACM Press, 1993.

Christian Borgelt. Efficient implementations of Apriori and Eclat. In *FIMI'03: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, November 2003.

Dario Bruzzese and Cristina Davino. Visual mining of association rules. In *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*, pages 103–122. Springer-Verlag, 2008.

Christian Buchta and Michael Hahsler. *arulesSequences: Mining Frequent Sequences*, 2010. URL `http://CRAN.R-project.org/package=arulesSequences`. R package version 0.1-11.

Michael Hahsler. A model-based frequency constraint for mining associations from transaction data. *Data Mining and Knowledge Discovery*, 13(2):137–166, September 2006.

Michael Hahsler. *arulesNBMiner: Mining NB-Frequent Itemsets and NB-Precise Rules*, 2010. URL `http://CRAN.R-project.org/package=arulesNBMiner`. R package version 0.1-1.

Michael Hahsler and Sudheer Chelluboina. *arulesViz: Visualizing Association Rules*, 2010. URL `http://CRAN.R-Project.org/package=arulesViz`. R package version 0.1-0.

Michael Hahsler and Kurt Hornik. New probabilistic interest measures for association rules. *Intelligent Data Analysis*, 11(5):437–455, 2007a.

Michael Hahsler and Kurt Hornik. Building on the arules infrastructure for analyzing transaction data with R. In R. Decker and H.-J. Lenz, editors, *Advances in Data Analysis, Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V., Freie Universität Berlin, March 8–10, 2006*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 449–456. Springer-Verlag, 2007b.

Michael Hahsler, Bettina Grün, and Kurt Hornik. arules – A computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, 14(15):1–25, October 2005.

Michael Hahsler, Christian Buchta, and Kurt Hornik. Selective association rule generation. *Computational Statistics*, 23(2):303–315, April 2008.

Michael Hahsler, Christian Buchta, Bettina Grün, and Kurt Hornik. *arules: Mining Association Rules and Frequent Itemsets*, 2010. URL `http://CRAN.R-project.org/package=arules`. R package version 1.0-3.

Mojdeh Jalali-Heravi and Osmar R. Zaïane. A study on interestingness measures for associative classifiers. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1039–1046. ACM, 2010.

Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceedings of the 4rd International Conference Knowledge Discovery and Data Mining (KDD-98)*, pages 80–86. AAAI Press, 1998.

Graham Williams, Michael Hahsler, Hemant Ishwaran, Udaya B. Kogalur, and Rajarshi Guha. *pmml: Generate PMML for various models*, 2010. URL `http://CRAN.R-project.org/package=pmml`. R package version 1.2.22.

Mohammed J. Zaki. SPADE: an efficient algorithm for mining frequent sequences. *Machine Learning*, 42:31–60, January–February 2001.

Journal of Machine Learning Research 12 (2011) 2027-2044 Submitted 1/11; Published 6/11

A Cure for Variance Inflation in High Dimensional Kernel Principal Component Analysis

Trine Julie Abrahamsen TJAB@IMM.DTU.DK
Lars Kai Hansen LKH@IMM.DTU.DK
DTU Informatics
Technical University of Denmark
Richard Petersens Plads, 2800 Lyngby, Denmark

Editor: Manfred Opper

Abstract

Small sample high-dimensional principal component analysis (PCA) suffers from variance inflation and lack of generalizability. It has earlier been pointed out that a simple leave-one-out variance renormalization scheme can cure the problem. In this paper we generalize the cure in two directions: First, we propose a computationally less intensive approximate leave-one-out estimator, secondly, we show that variance inflation is also present in kernel principal component analysis (kPCA) and we provide a non-parametric renormalization scheme which can quite efficiently restore generalizability in kPCA. As for PCA our analysis also suggests a simplified approximate expression.

Keywords: PCA, kernel PCA, generalizability, variance renormalization

1. Introduction

While linear dimensionality reduction by principal component analysis (PCA) is a trusted machine learning workhorse, kernel based methods for *non-linear* dimensionality reduction are only starting to find application. We expect the use of non-linear dimensionality reduction to expand in many applications as recent research has shown that kernel principal component analysis (kPCA) can be expected to work well as a pre-processing device for pattern recognition (Braun et al., 2008). In the following we consider non-linear signal detection by kernel PCA followed by a linear discriminant classifier.

In spite of its conceptual simplicity and ubiquitous use, principal component learning in high dimensions is in fact highly non-trivial (see, e.g., Hoyle and Rattray, 2007; Kjems et al., 2001). In the physics literature much attention has been devoted to learnability phase transitions. In PCA there is a sharp transition as function of sample size from *no learning at all* to a regime where the projections become more and more accurate. In the transition regime where learning is still incomplete there is a mismatch between the test and training projections. In Kjems et al. (2001) it was shown that this can be interpreted as a case of *over-fitting* and leads to pronounced *variance inflation* in the training set projections and results in lack of generalization to test data as illustrated in Figure 1.

Variance inflation is of particular concern if PCA is used to reduce dimensionality prior to, for example, a classifier. When the data analytic pipeline is applied to test data the reduced variance of the PCA text projections can lead to significantly reduced performance. Fortunately, the bias can

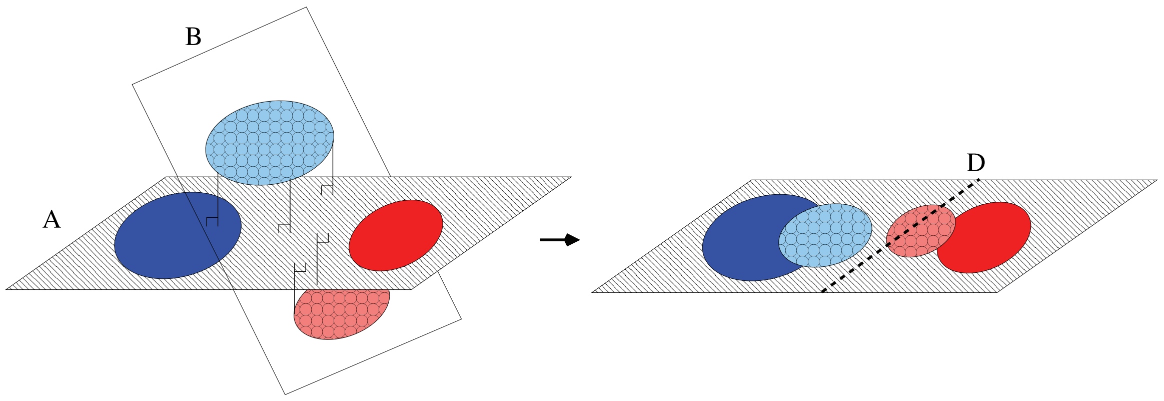


Figure 1: Illustration of the variance inflation problem in PCA. Because PCA maximizes variance, small data sets in high dimensions will be overfitted. When the PCA subspace (A) is applied to a test data set (B) the projected data will have smaller variance. This leads to lack of generalizability if the training data is used to train a classifier, say a linear discriminant (D). In Kjems et al. (2001) this problem was noted and it was shown that the necessary renormalization can be estimated in a leave-one-out procedure

be reduced effectively by a leave-one-out (LOO) scale renormalization of the PCA test projections to restore generalizability (Kjems et al., 2001). In this paper we pursue several extensions of this result. We give a straightforward geometric analysis of the projection problem that suggests a computationally less intensive approximate cure than the one originally proposed by Kjems et al. (2001). Next, we proceed to investigate the issue in the context of *kernel* based unsupervised dimensionality reduction. We show in both simulation and in real world data (USPS handwritten digits and functional MRI data) that variance inflation also happens in kPCA and basically for the same reasons as in PCA. We then provide an extension to the LOO procedure for kPCA which can cope with potential non-Gaussian distributions of the kPCA projections, and finally we propose a simplified approximate renormalization scheme.

2. Generalizability in PCA

The most complete theoretical picture of principal component learning is presented by Hoyle and Rattray (2007), which builds on and extends earlier work by, for example, Biehl and Mietzner (1994), Hoyle and Rattray (2004c), Johnstone (2001), Reimann et al. (1996), and Silverstein and Combettes (1992). Hoyle and Rattray (2007) consider a general PCA model with a multidimensional normal distributed signal that emerges from an isotropic noise background as the sample size increases. The stabilization of a given principal component happens at a given sample size and takes the form of a phase transition. For small sample sizes -below the phase transition point - the training set principal component eigenvectors are in completely random directions in space and there is no learning at all. Then, as the sample size increases, the first principal component stabilizes, and for even larger sample sizes the second, and so forth. Sharp transitions are strictly present only in a limit where both dimensionality and sample size are infinite with a finite ratio $\alpha = N/D$, but the theoretical results are very accurate at realistic dimensions as seen in Figure 2. The location of the first

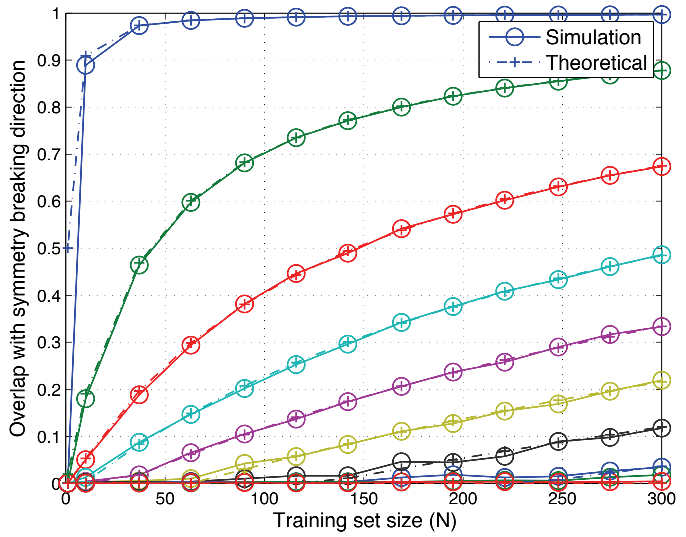


Figure 2: Phase transitions in PCA. Simulated data was created as $\boldsymbol{x} = \eta \boldsymbol{u} + \boldsymbol{\epsilon}$, with a normal distributed signal of unit strength $\eta \sim N(0,1)$, embedded in i.i.d. normal noise $\boldsymbol{\epsilon} \sim N(0, \sigma^2 \mathbf{1})$. In this simulated data set we show the phase transition like behavior of the overlap (the mean square of the projection) of the first PCA eigenvector and the signal direction $\boldsymbol{u}$. The input space has dimension $D = 1000$, and the curves are for 10 values of signal to noise within the interval $\sigma \in [0.01, 0.5]$. For a noise level of, for example, $\sigma = 0.17$ (black curves) there is a sharp transition both in the theoretical curve (dash/cross) and the experimental curve (full/circle) around $N = 120$ examples.

phase transition depends on the signal variance to noise variance ratio (SNR). The theoretical result provides a *mean bias* for a specific model, hence, cannot directly be used to restore generalizability in a given data set.

Now, what happens to the generalization performance of PCA in the noisy region? The PCA projections will be offset by different angles depending on how severe the given component is affected by the noise. Because of the bias the test projections will follow different probability laws than the training data, typically with much lower variance. Hence, if we train a classifier on the training projections the classifier will make additional errors on the test set as visualized in Figure 1.

In the case of PCA the subspace projections are uncorrelated, hence, it is meaningful to renormalize them independently. Assuming approximate normality, a simple affine transformation suffices. The scale factor is simply the ratio of the standard deviations of the training and test projections and can be estimated by a leave-one-out procedure (Kjems et al., 2001). However, since the LOO procedure involves the computation of N SVD's of an $(N-1) \times (N-1)$ matrix, it is of interest to find a simplified estimate.

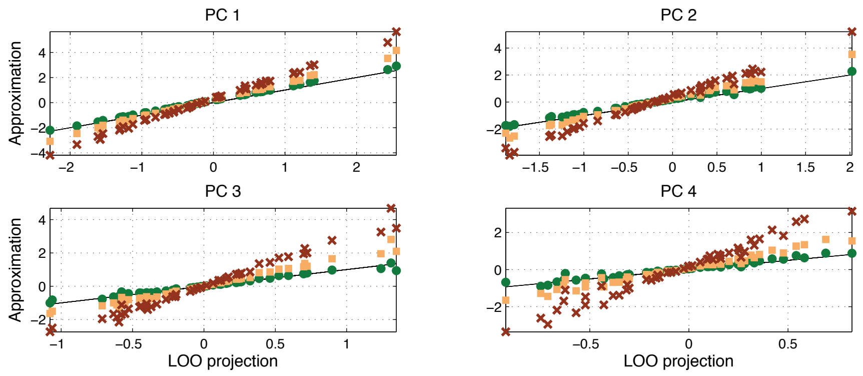


Figure 3: Approximating the leave-one-out (LOO) procedure. Here we simulate data with four normal independent signal components, $\boldsymbol{x} = \Sigma_{k=1}^{4} \eta_k \boldsymbol{u}_k + \boldsymbol{\epsilon}$ of strengths $(1.4, 1.2, 1.0, 0.8,$ embedded in i.i.d. normal noise $\boldsymbol{\epsilon} \sim N(0, \sigma^2 \mathbf{1})$, with $\sigma = 0.2$. The dimension was $D = 2000$ and the sample size was $N = 50$. In the four panels we show the training set projections (red crosses), the projections corrected for the theoretical mean overlap (Hoyle and Rattray, 2007) (yellow squares) and the geometric approximation in Equation (1) (green dots) versus the exact LOO projections (black line).

Let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be N training data points in a D dimensional input space $\mathcal{X}$ (see notation),[1] we consider the case $N \ll D$. The LOO step for the N'th point $\boldsymbol{x}_N$ concerns projecting onto the PCA eigenvectors derived from the subset $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-1}\}$. Define the orthogonal and parallel components of the test point, $\boldsymbol{x}_N = \boldsymbol{x}_N^{\perp} + \boldsymbol{x}_N^{\parallel}$, relative to the subspace spanned by the training data. As the PCA eigenvectors with non-zero variance are all in the span of the training data we obtain

$$\boldsymbol{u}_{N-1,k}^T \cdot \boldsymbol{x}_N = \boldsymbol{u}_{N-1,k}^T \cdot \boldsymbol{x}_N^{\parallel} \,,$$

where $\boldsymbol{u}_{N-1,k}$ is the k'th eigenvector of the LOO training set. Assuming that the changes in the PCA eigenvectors going from sample size N to $N-1$ are small, we can approximate the test projections as

$$\boldsymbol{u}_{N-1,k}^T \cdot \boldsymbol{x}_N = \boldsymbol{u}_{N-1,k}^T \cdot \boldsymbol{x}_N^{\parallel} \approx \boldsymbol{u}_{N,k}^T \cdot \boldsymbol{x}_N^{\parallel} \,, \tag{1}$$

where $\boldsymbol{u}_{N,k}$ is the k'th eigenvector on the full sample. The approximation introduces a small error of order $1/N$ as discussed in detail in the Appendix and further illustrated in a simulation data set in Figure 3. Note that the orthogonal projections $\boldsymbol{x}_N^{\parallel}$ of the N points may be calculated from the inverse matrix of the inner products of all data points, in N steps each of a cost scaling as N^2, thereby achieving a computational burden which scales as N^3 rather than the N^4 scaling for an exact LOO procedure proposed in Kjems et al. (2001).

1. Bold uppercase letters denote matrices, bold lowercase letters represent column vectors, and non-bold letters denote scalars. $\boldsymbol{a}_j$ denotes the *j'th* column of $\boldsymbol{A}$, while a_{ij} denotes the scalar in the *i'th* row and *j'th* column of $\boldsymbol{A}$. Finally $\mathbf{1}_{NN}$ is a $N \times N$ matrix of ones.

3. Renormalization Cure for Variance Inflation in kernel PCA

The statistical properties of kernel PCA have also been studied extensively by Blanchard et al. (2007), Hoyle and Rattray (2004a), Hoyle and Rattray (2004b), Mosci et al. (2007), Shawe-Taylor and Williams (2003) and Zwald and Blanchard (2006), but to our knowledge the geometry of generalization for kPCA has not been discussed in the extremely ill-posed case $N \ll D$.

To better understand the variance inflation problem in relation to kPCA let us recapitulate some basic aspects of this non-linear dimensional reduction technique.

Let $\mathcal{F}$ be the Reproducing Kernel Hilbert Space (RKHS) associated with the kernel function $k(\boldsymbol{x},\boldsymbol{x}') = \varphi(\boldsymbol{x})^T\varphi(\boldsymbol{x}')$, where $\varphi\ : \mathcal{X} \mapsto \mathcal{F}$ is a possibly non-linear map from the D-dimensional input space $\mathcal{X}$ to the high dimensional (possibly infinite) feature space $\mathcal{F}$. In kPCA the PCA step is carried out in the feature space, $\mathcal{F}$, mapped data (Schölkopf et al., 1998). However, as $\mathcal{F}$ can be infinite dimensional we first apply the kernel trick allowing us to work with the Gram matrix of inner products. Let $\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$ be N training data points in $\mathcal{X}$ and $\{\varphi(\boldsymbol{x}_1),\ldots,\varphi(\boldsymbol{x}_N)\}$ be the corresponding images in $\mathcal{F}$. The mean of the φ-mapped data points is denoted φ and the 'centered' images are given by $\tilde{\varphi}(\boldsymbol{x}) = \varphi(\boldsymbol{x}) - \varphi$. The kPCA is performed by solving the eigenvalue problem $\widetilde{\boldsymbol{K}}\boldsymbol{\alpha}_i = \lambda_i\boldsymbol{\alpha}_i$ where the centered kernel matrix, $\widetilde{\boldsymbol{K}}$, is defined as

$$\widetilde{\boldsymbol{K}} = \boldsymbol{K} - \frac{1}{N}\mathbf{1}_{NN}\boldsymbol{K} - \frac{1}{N}\boldsymbol{K}\mathbf{1}_{NN} + \frac{1}{N^2}\mathbf{1}_{NN}\boldsymbol{K}\mathbf{1}_{NN}\ . \tag{2}$$

The projection of a φ-mapped test point onto the *i'th* component is given by

$$\beta_i = \tilde{\varphi}(\boldsymbol{x})^T\boldsymbol{v}_i = \sum_{n=1}^{N}\alpha_{in}\tilde{\varphi}(\boldsymbol{x})^T\tilde{\varphi}(\boldsymbol{x}_n) = \sum_{n=1}^{N}\alpha_{in}\tilde{k}(\boldsymbol{x},\boldsymbol{x}_n)\ , \tag{3}$$

where $\boldsymbol{v}_i$ is the *i'th* eigenvector of the feature space covariance matrix and the $\boldsymbol{\alpha}_i$'s have been normalized. The centered kernel function can be found as $\tilde{k}(\boldsymbol{x},\boldsymbol{x}') = k(\boldsymbol{x},\boldsymbol{x}') - \frac{1}{N}\mathbf{1}_{1N}\boldsymbol{k}_{\boldsymbol{x}} - \frac{1}{N}\mathbf{1}_{1N}\boldsymbol{k}_{\boldsymbol{x}'} + \frac{1}{N^2}\mathbf{1}_{1N}\boldsymbol{K}\mathbf{1}_{N1}$, where $\boldsymbol{k}_{\boldsymbol{x}} = [k(\boldsymbol{x},\boldsymbol{x}_1),\ldots,k(\boldsymbol{x},\boldsymbol{x}_N)]^T$. The projection of $\varphi(\boldsymbol{x})$ onto the first q principal components will in be denoted $P_q(\boldsymbol{x})$.

In the following we focus on a Gaussian kernel of the form $k(\boldsymbol{x},\boldsymbol{x}') = \exp(-\frac{1}{c}||\boldsymbol{x}-\boldsymbol{x}'||^2)$, where c is the scale parameter controlling the non-linearity of the kernel map. By the centering operation, PCA is the obtained in the limit when $c \to \infty$. Thus for large values we expect variance inflation to be present due the reasons discussed above. What happens in the non-linear regime with a finite c? To answer this question we analyze the LOO scenario for kPCA.

Consider the squared distance $||\boldsymbol{x}_n - \boldsymbol{x}_N||^2$ in the exponent in the Gaussian kernel for some training set point $\boldsymbol{x}_n$ and a test point $\boldsymbol{x}_N$. If we split the test point in the orthogonal components as above with respect to the subspace spanned by the training set we obtain,

$$||\boldsymbol{x}_n - \boldsymbol{x}_N||^2 = ||\boldsymbol{x}_n - \boldsymbol{x}_N^{||}||^2 + ||\boldsymbol{x}_N^{\perp}||^2\ .$$

Inserting this expression in the Gaussian kernel in Equation (3) it is seen that the test projection acquire a common factor $\exp\left(-\frac{1}{c}||\boldsymbol{x}_N^{\perp}||^2\right)$:

$$\beta_i(\boldsymbol{x}_N) = \sum_{n=1}^{N-1}\alpha_{in}\tilde{k}(\boldsymbol{x}_N,\boldsymbol{x}_n) = \exp\left(-\frac{1}{c}||\boldsymbol{x}_N^{\perp}||^2\right)\sum_{n=1}^{N-1}\alpha_{in}\tilde{k}(\boldsymbol{x}_N^{||},\boldsymbol{x}_n)\ ,$$

which can be arbitrary small for small values c, that is, in the non-linear regime.

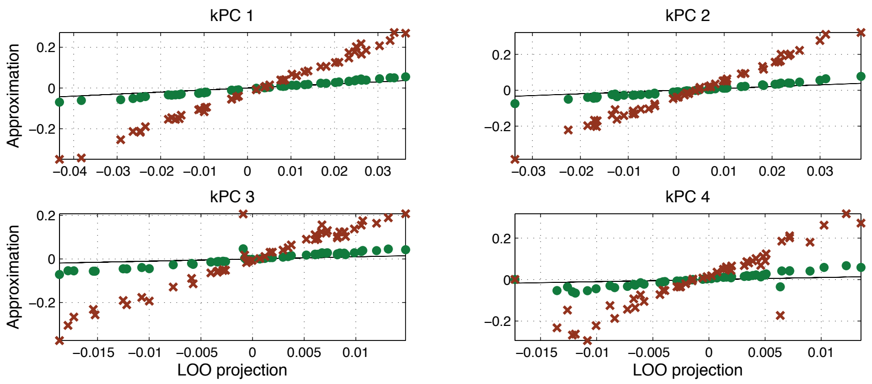


Figure 4: Approximating the leave-one-out (LOO) procedure for kPCA. We simulate a data set with four normal independent signal components, $\boldsymbol{x} = \Sigma_{k=1}^{4} \eta_k \boldsymbol{u}_k + \boldsymbol{\epsilon}$ of strengths $(1.4, 1.2, 1.0, 0.8$, embedded in i.i.d. normal noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{1})$, with $\sigma = 0.2$. The dimension was chosen $D = 2000$ and the sample size was $N = 50$. In the four panels we show the four kPCA component's training set projections (red crosses), and the result of applying the point wise correction factor $\exp(\frac{1}{c}||\boldsymbol{x}_N^{\perp}||^2)$ for the lost orthogonal projection (green dots) versus the exact LOO kPCA test projections (black).

For a coordinate-wise LOO renormalization procedure we thus propose to compute N test projections by repeated kPCA on the $N-1$ sized sub training sets. However, compared to the PCA case we face two additional challenges, namely the potentially strongly non-Gaussian distributions and component dependencies.

To check for dependency we appeal to simple pairwise permutation test of significant mutual information measure (see, e.g., Moddemeijer, 1989). If the null hypothesis is rejected for a given set of components we cannot expect coordinate-wise renormalization to be effective. If, on the other hand, the kernel PCA projections pass the independence test we can proceed to renormalize the components individually. In the following we will assume that a coordinate-wise approach is acceptable. First, as a simple approximation to the full LOO we consider adjusting for the common scaling factor due to the lost orthogonal projection. This may indeed provide for viable approximation as seen in Figure 4.

To address the second challenge, namely the potential non-normality we propose to generalize the affine scaling method of Kjems et al. (2001) by a non-parametric procedure. Assume that there exists a monotonic transformation between the N training and N LOO test set projections. The problem of calibrating for an unknown monotone transformation is a common operation in image processing, and is used, for example, to transform the gray scale of an image in order to standardize the pixel histogram (Gonzalez and Wintz, 1977). Equalizing two equal sized samples, simply involves sorting both and assigning the sorted test projections the sorted values of the training projections, this procedure is easily seen to equalize the histograms without changing the level sets (relative ordering) of the LOO test projections. In Figure 5 a simple 1-dimensional data set is used to illustrate the equalization procedure. The training set clearly contains two classes. However, due to variance inflation (induced by, for example, kernel PCA) the test set does not follow the same

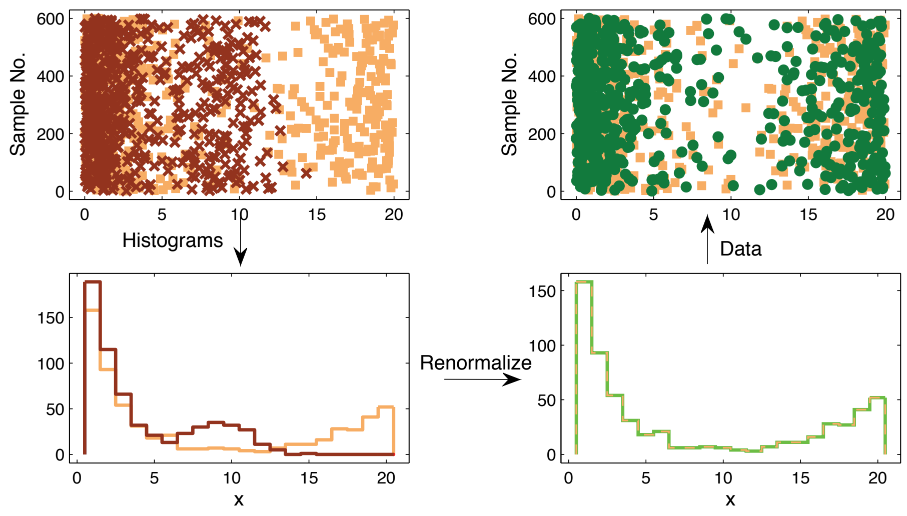


Figure 5: Illustration of renormalization by histogram equalization. The left panel shows the training set (yellow squares) and original test set (red crosses) and their respective histograms. The histograms are then equalized as seen in the right panel, where the green dots are the renormalized test data. The renormalization clearly restores the variation of the test set.

distribution, and may potentially lead to a high misclassification rate. The right panel of the figure shows how histogram equalization restores generalizability.

Technically, the transformation may be described as follows. Let $H(f)$ be the cumulative distribution of values f of a given kPCA projection of the training set. Let the test set projections on the same component for N_{test} samples take values $g(m)$. Let $I(m)$ be the index of sample m in a sorted list of the test set values. Then the renormalized value of the test projection m is

$$\widetilde{g(m)} = H^{-1}(I(m)/N_{\text{test}}) \; .$$

The test set projections can be obtained by the simple relation

$$\widetilde{g(m)} = f_{\text{sort}}(I(m)) \; , \tag{4}$$

where f_{sort} is the sorted list of training set projections. The algorithm for approximate renormalization is summarized in Algorithm 1.[2]

4. Evaluation of the Proposed Cure in Classification Problems

In the following we evaluate the non-parametric exact LOO correction scheme when kPCA is used as a dimensional reduction step in simulated and real classification data sets.

2. We thank the reviewers for pointing out that while non-normality is expected in the case of kPCA, non-normality may also appear in PCA calling for application of the proposed non-parametric renormalization scheme in this case.

Algorithm 1 Approximate renormalization in kernel PCA

```
Require: $\boldsymbol{X}_{tr}$ and $\boldsymbol{X}_{te}$to be $N_{tr} \times D$ and $N_{te} \times D$ respectively
  Compute $\widetilde{\boldsymbol{K}}_{tr}$ using Equation (2) and find the eigenvectors, $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_q$
  for $i = 1$ to $N_{tr}$ do
    $\boldsymbol{f}_{tr}^{i,:} \leftarrow P_q(\boldsymbol{x}_{tr}^{i,:}) = \tilde{\boldsymbol{k}}_{x_i}^T \boldsymbol{\alpha}^{i:q}$ {see Equation (3)}
  end for
  for $j = 1$ to $N_{te}$ do
    $\boldsymbol{f}_{te}^{j,:} \leftarrow P_q(\boldsymbol{x}_{te}^{j,:}) = \tilde{\boldsymbol{k}}_{x_j}^T \boldsymbol{\alpha}^{i:q}$ {see Equation (3)}
  end for
  for $d = 1$ to $q$ do
    $[\boldsymbol{f}_{sort},\ ] \leftarrow \text{sort}(\boldsymbol{f}_{tr}^{:,d})$ {ascending order}
    $[\ ,I] \leftarrow \text{sort}(\boldsymbol{f}_{te}^{:,d})$ {ascending order}
    if $N_{tr} = N_{te}$ then
      $\boldsymbol{h} \leftarrow \boldsymbol{f}_{sort}$
    else {$N_{tr} \neq N_{te}$}
      $\boldsymbol{h} \leftarrow \text{spline}\big([1:N_{tr}], \boldsymbol{f}_{sort}, \text{linspace}(1, N_{tr}, N_{te})\big)$ {interpolate to create $N_{te}$ values of $\boldsymbol{f}_{sort}$ in
      the interval $[1:N_{tr}]$}
    end if
    for $n = 1$ to $N_{te}$ do
      $\tilde{\boldsymbol{g}}_{te}^{I(n),d} \leftarrow \boldsymbol{h}^{n,d}$ {renormalized test data in the principal subspace, see Equation (4)}
    end for
  end for
```

4.1 Simulated Data

To get some insight into the non-linear regime, we design a synthetic data set containing two 2-dimensional semi-circular clusters which cannot be separated linearly (cf., Jenssen et al., 2006). Gaussian noise is added to one of the clusters, and the data is further embedded in 1000 'noise dimensions'. The basis is changed so that the 2D signal space occupies a general position. The noise is as earlier assumed i.i.d. with variance σ^2. The assignment variable is $t = 0, 1$, and in the experiments the data set is assumed unbalanced with $p(t = 0) = 0.6$.

In Figure 6 we show in the left panel a linear discriminant trained on the training set projections in a data set of $N = 500$ in $D = 1000$ dimensions. The role of the non-linearity as controlled by the parameter c in the Gaussian kernel is investigated in Figure 7 for a simulation setup similar to Figure 6. As seen the inflation problem dramatically amplifies as non-linearity increases. Finally, Figure 8 shows how renormalization improves the learning curve for the same problem.

4.2 USPS Handwritten Digit Data

The USPS handwritten digit benchmark data set is often used to illustrate unsupervised and supervised kernel methods. The USPS data set consists of $D = 16 \times 16 = 256$ pixels handwritten digits.[3] For each digit we randomly chose 10 examples for training and another 10 examples for testing. The scale was chosen as the 5th percentile of the mutual distances of the data points leading to $c \approx 120$,

3. The USPS data set is described by Hull (1994) and can be downloaded from `www.kernel-machines.org`.

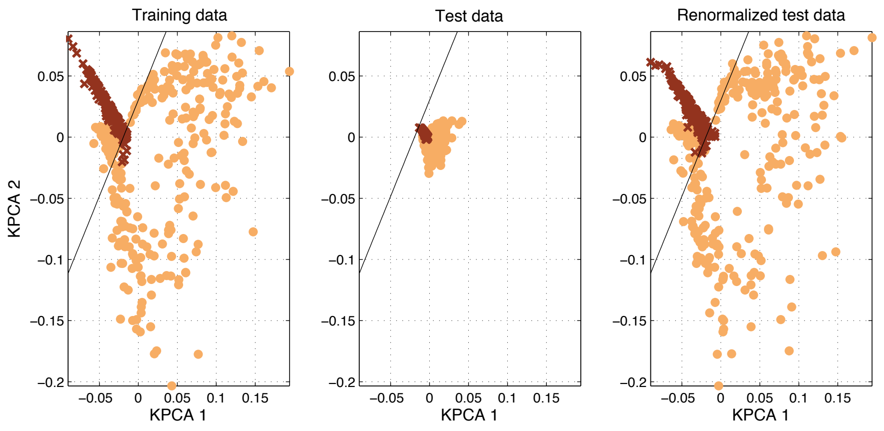


Figure 6: An unbalanced two cluster data set showing a pronounced variance inflation problem in the projections of the test data in the middle panel. In the right panel we have applied the cure based on non-parametric renormalization to equalize training and test projections using histogram equalization. The linear discriminant performs close to the optimal Bayes rate after non-parametric renormalization. The sample size is $N = 500$ in $D = 1000$ dimensions and the SNR is 10. The training error rate is 0.002 while the uncorrected test error rate is 0.4. Renormalization reduces the test error to 0.002.

and the number of principal components was chosen so 85% of the variance was contained in the principal subspace leading to around $q = 57$ PCs to be included.

The first step is to submit the data to the mutual information permutation test. For every pair of principal components a permutation test with 1000 permutations was performed in order to test the null hypothesis of the two given components being independent. Using a $\rho = 0.05$ significance level, we find that the null hypothesis can only be rejected for approximately 2% of the principal component pairs when not using Bonferroni correction. The combinations for which the null hypothesis can be rejected are equally distributed across the principal components. Since the expected number of rejected tests at the given confidence level is 5%, hence, we can safely proceed with the coordinate-wise renormalization process.

In the q dimensional principal subspace the projections of the test set are renormalized to follow the training set histogram. We chose in these experiments for demonstration to classify digit 8 versus the rest. A linear discriminant classifier was trained on the kernel PCA projections of the training set, and the classification error was found using both the conventional kernel PCA projections of the test set and their renormalized counterparts. In order to compare the two methods, the procedure was repeated 300 times using random training and test sets. While classification based on the conventional projections resulted in a mean classification error rate ($\pm$ 1 std) of 0.06 ± 0.01, using the renormalized projections lowered the error rate to 0.05 ± 0.02. A paired t-test showed that this reduction is highly significant ($p = 2.0875 \cdot 10^{-11}$).

Figure 9 shows an example of the projections before and after renormalization. The axis are fixed across the two methods. The top row clearly illustrates the inflation problem for conventional kPCA. Furthermore, due to the imbalanced nature of the data set, the inflation causes a high misclas-

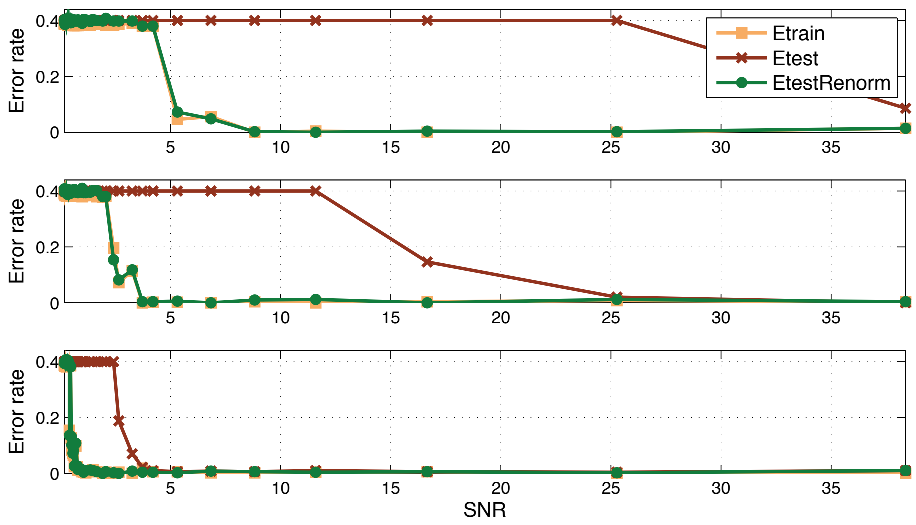


Figure 7: The role of non-linearity on the variance inflation problem. We carry out three experiments at different values of the Gaussian kernel scale parameter (top to bottom: $c = 0.05$, $c = 0.1$, $c = 0.5$). We show classification errors as a function of SNR. The linear discriminant performs close to the optimal Bayes rate after the renormalization operation in all cases, while the un-renormalized systems suffers from poor generalizability. The sample size is $N = 500$ and the number of dimensions is $D = 1000$.

sification rate. The bottom row illustrates how renormalization overcomes the distortions induced by the variance inflation. The discriminant line is seen to separate the two classes appropriately.

To gain a better understanding of how the variance inflation and quality of the renormalization are effected by noise, we added Gaussian noise ($\mathcal{N}(0,\sigma_e^2)$) with $\sigma_e \in [0,5]$. For every noise level, 300 random training and test sets where drawn as explained above and kPCA was performed. Once again our goal was to classify digit 8 versus the rest by a linear classifier in the principal subspace. The results are summarized in Figure 10 where we show the error rate before and after renormalization as well as the result based on renormalizing according to the leave-one-out error. In the last case, the N projections determined from leave-one-out cross validation (LOOCV) are renormalized to follow the entire training set histogram. Renormalization is then only applied to the test set when this renormalized LOOCV error is less than the estimated baseline error. In the right panel of Figure 10 it is seen how renormalizing the projections leads to a much improved classifier as long as the SNR is ‘reasonable’. Even when $\sigma_e = 0$ there is some inherent noise in the data, which explains why renormalization still improves the classification. As σ_e reaches 1 it is no longer possible to identify the digits by visual inspection, and classification becomes increasingly difficult.

The left panel of Figure 10 shows how the conventional error rate converges to the baseline of 0.1 (misclassifying all digits 8), for high noise levels. Basically, increasing the noise result in a more skewed test set subspace in relation to the subspace spanned by the training set (see Figure 1). At a given threshold this causes all the projections to lie on the same side of the discrimination function due to the imbalanced composition, leading to a misclassifications rate of $1/10$. As the idea of renormalization by histogram equalization is to restore the variation in the test set, this be-

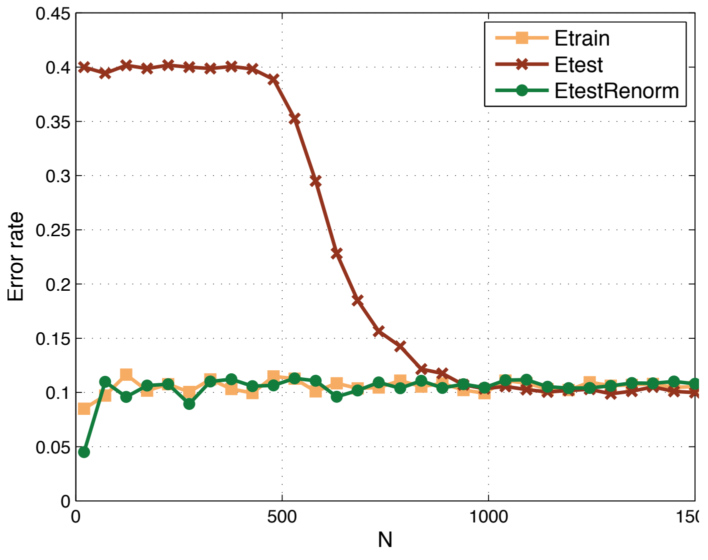


Figure 8: Classification error learning curves for the two semicircular clusters in i.i.d. noise setup. The signal to noise ratio was $SNR = 60$. The linear discriminant performs close to the optimal Bayes rate after the renormalization operation in all cases, while the conventional system suffers from poor generalizability, and requires about ten times as many examples to reach the same error level as the renormalized classifier. The experiment was carried out with $D = 2000$.

havior is naturally not encountered for the renormalized projections. Instead, as the SNR decreases, renormalization increases the error rate, as the test set observations are forced to be distributed on both sides of the discrimination line - which leads to many misclassifications when the signal is suppressed by the noise. However, using LOOCV based renormalization prevents the error rate from blowing up while at the same time improving the classification in the more sensible SNR regime as compared to conventional kPCA.

4.3 Functional MRI Data

As a second high dimensional real data example, functional magnetic resonance imaging (fMRI) data was used to illustrate the effect of renormalization. The fMRI data set was acquired by Dr. Egill Rostrup at Hvidovre Hospital on a 1.5 T Magnetom Vision MR scanner. The scanning sequence was a 2D gradient echo EPI (T2- weighted) with 66 ms echo time and 50° RF flip angle. The images were acquired with a matrix of $D = 128 \times 128 = 16,384$ pixels, with FOV of 230 mm, and 10 mm slice thickness, in a para-axial orientation parallel to the calcarine sulcus. The visual paradigm consisted of a rest period of 20 sec of darkness using a light fixation dot, followed by 10 sec of full-field checkerboard reversing at 8 Hz, and ending with 20 sec of rest (darkness). In total, 150 images were acquired in 50 sec, corresponding to a period of approximately 330 msec per image. The experiment was repeated in 10 separate runs containing 150 images each. In order to reduce saturation effects, the first 29 images were discarded, leaving 121 images for each run. We use a

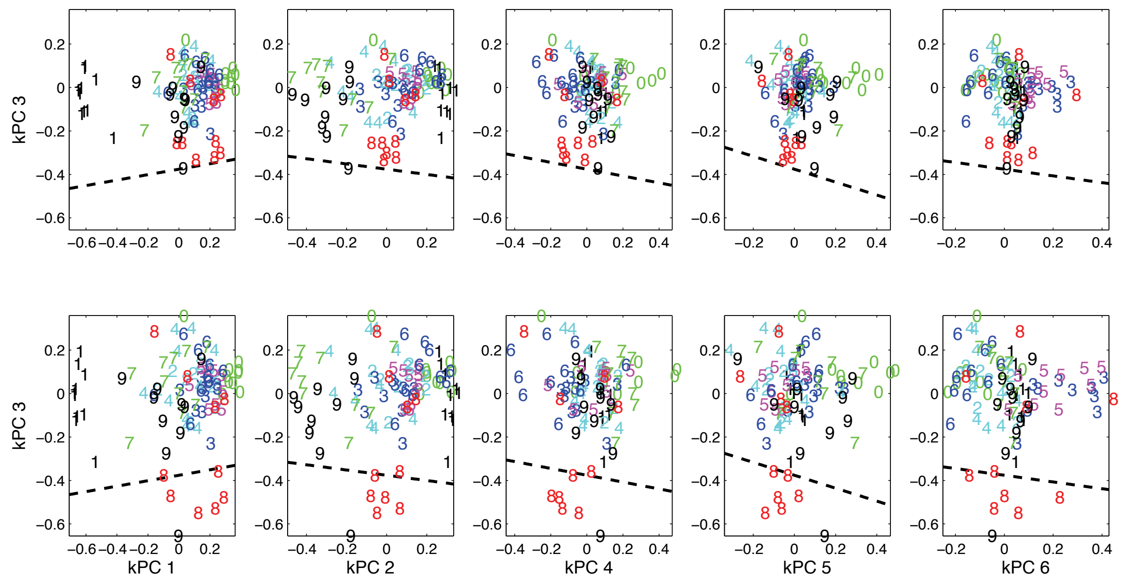


Figure 9: USPS handwritten digits test set projections. The top row shows the conventional projections, while the bottom row shows the projections after renormalization. In this example the third kPC carries a large part of the signal, and hence this component is shown versus the other five first PCs. The variance reduction and the consequent shift is evident from the top row. The dashed line indicates the linear discriminant function for classifying digit 8 vs the rest.

simple on-off activation reference function for supervision of the classifier. The reference function is off-set by 4 seconds to emulate the hemodynamic delay.

The data set is split in two equal sized subsets: Five runs for training and five runs for testing. As the test and training data are independent, the test error estimate is an unbiased estimator of performance. The scale of the Gaussian kernel was chosen as the 5th percentile of the mutual distances leading to $c \approx 15000$, while the dimension of the principal subspace is chosen as $q = 20$.

Again the principal components are tested for independence by a mutual information permutation test. Using 1000 permutations and a $\rho = 0.05$ significance level, we find that the null hypothesis is rejected for approximately 1% of the principal component pairs.

Similar to the handwritten digit data we perform linear classification in the kernel principal subspace. This was repeated 300 times using random splits for different noise levels. The results are summarized in Figure 11. Again renormalization is seen to decrease the error rate significantly, while the LOOCV based scheme furthermore prevents the increase in error rate for high noise levels (low SNR).

Figure 12 shows the projection of the data onto the first kPC's before and after renormalization.

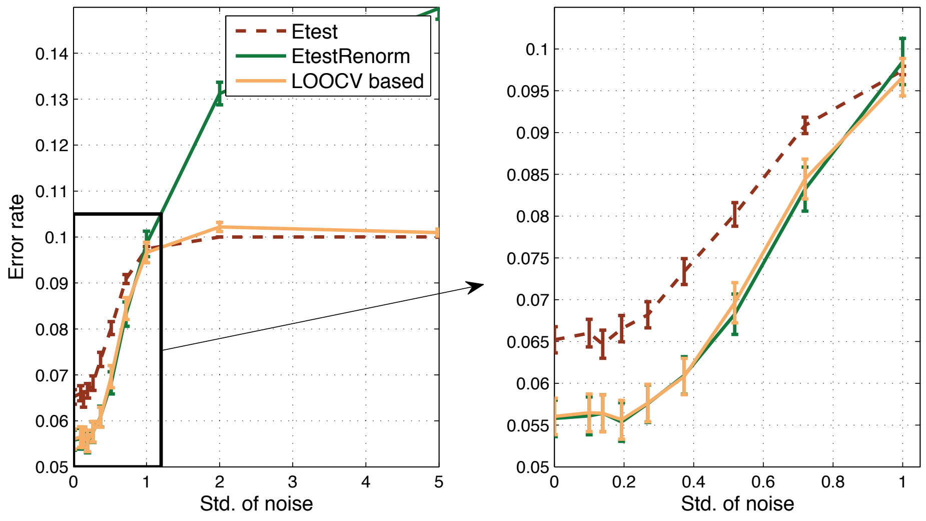


Figure 10: Mean error rates $\pm$ 1 standard deviation as a function of the noise level. The test error based on conventional kernel PCA projections, renormalized projections, and a LOOCV scheme is shown. Renormalization is seen to improve the performance, while LOOCV based renormalization prevents the classification error to blow up in the very low SNR regime.

5. Conclusion

Dimensionality reduction by PCA and kPCA can lack generalization due to training set variance inflation in the extremely ill-posed case when the sample size is much smaller than the input space dimension. In this work we have provided a simple geometric explanation for the main effect, namely that test points 'loose' their orthogonal projections, when their embedding is computed. This insight allowed for a speed-up of a previously proposed LOO scheme for renormalization. For kPCA we showed that the effects can be even more dramatic than in PCA, and we proposed a scheme for exact LOO renormalization of the embedding, and an approximate expression at lower cost. The viability of the new scheme was demonstrated for kPCA when used for dimensionality reduction both in simple synthetic data, in the USPS digit classification problem, and for fMRI brain state decoding.

Acknowledgments

We thank the reviewers of this manuscript and earlier versions for many useful comments. This research was supported by the Danish Lundbeckfonden through the Center for Integrated Molecular Brain Imaging (`www.cimbi.dk`).

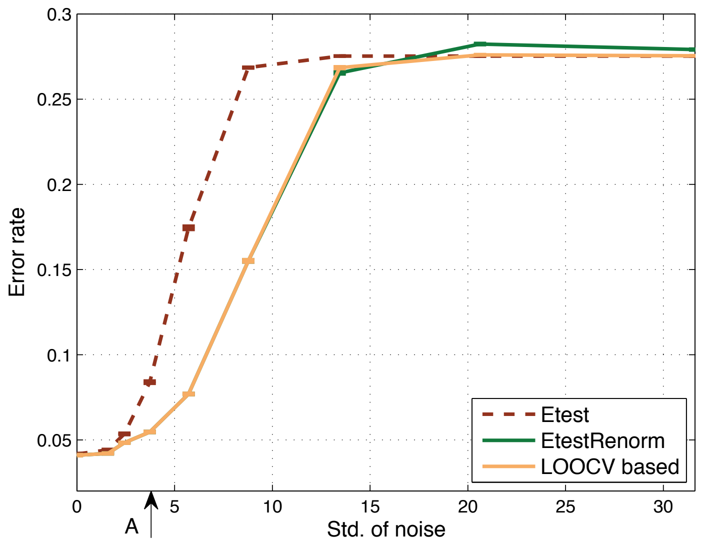


Figure 11: Mean error rates $\pm$ 1 standard deviation as a function of the noise level for fMRI data ($D = 16,384, N = 605$) . The test error based on conventional kernel PCA projections, renormalized projections, and a LOOCV scheme is shown. Renormalization is seen to clearly improve the performance. Arrow 'A' indicates the noise level used in Figure 12

Appendix A.

Let $\boldsymbol{u}_{N,k}$ be the k'th eigenvector of the covariance matrix on the full sample $\boldsymbol{\Sigma}_N$ and $\boldsymbol{u}_{N-1,k}$ be the corresponding eigenvector of LOO training set covariance matrix $\boldsymbol{\Sigma}_{N-1}$. In the following we use first order perturbation theory to show that

$$\boldsymbol{u}_{N-1,k}^T \cdot \boldsymbol{x}_N \approx \boldsymbol{u}_{N,k}^T \cdot \boldsymbol{x}_N^{\parallel} ,$$

where the data vector $\boldsymbol{x}$ has been split in its orthogonal and parallel components, $\boldsymbol{x}_N = \boldsymbol{x}_N^{\perp} + \boldsymbol{x}_N^{\parallel}$, relative to the subspace spanned by the training data. Thus, we are interested in the difference between $\boldsymbol{u}_{N,k}$ and $\boldsymbol{u}_{N-1,k}$. Simple manipulations of the covariance matrices lead to

$$\boldsymbol{\Sigma}_{N-1} = \boldsymbol{\Sigma}_N + \underbrace{\frac{1}{N-1}\boldsymbol{\Sigma}_N - \frac{1}{N}(\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1})(\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1})^T}_{O(\frac{1}{N})} .$$

By introducing the shorthand $\boldsymbol{A} = \boldsymbol{\Sigma}_{N-1}$ and $\boldsymbol{B} = \boldsymbol{\Sigma}_N$ we get

$$\boldsymbol{A} = \boldsymbol{B} + \delta \boldsymbol{C} , \tag{5}$$

where δ is of order $\frac{1}{N}$. Note that all matrices are symmetric. We now look at the k'th eigenvector of $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$\boldsymbol{B}\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k , \tag{6}$$

$$\boldsymbol{A}\boldsymbol{v}_k = \nu_k \boldsymbol{v}_k . \tag{7}$$

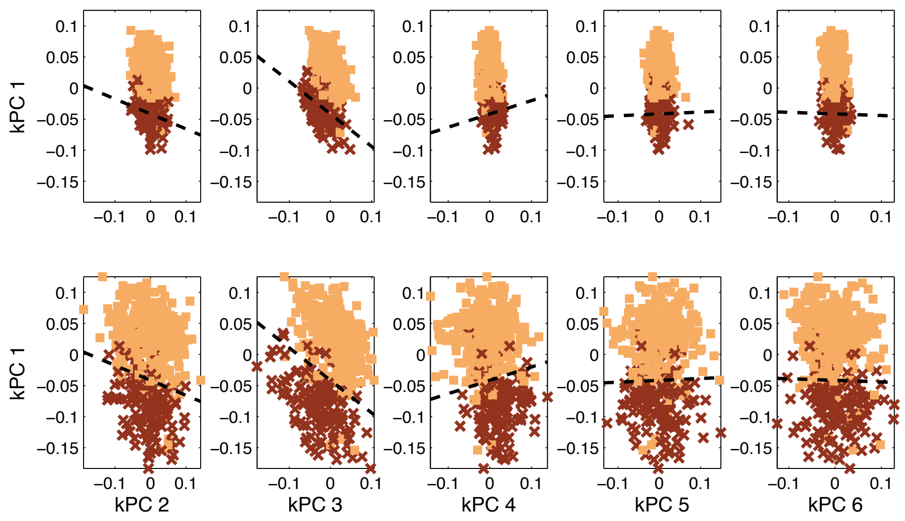


Figure 12: Test set projections of the fMRI data with Gaussian noise added as marked on Figure 11 ($\varepsilon_i = \mathcal{N}(0, 3.8^2)$). The top row shows the conventional projections, while the bottom row shows the projections after renormalization. The 'red class' indicates activation, while the blue observations are acquired during rest. The dashed line marks the linear discriminant. The scale is chosen as the 5th percentile of the mutual distances.

First order perturbation theory posits

$$\nu_k = \lambda_k + \delta\xi_k \,, \tag{8}$$
$$\boldsymbol{v}_k = \boldsymbol{u}_k + \delta\boldsymbol{w}_k \,. \tag{9}$$

That is, when going from N to $N-1$ samples we only have a small ($O(\frac{1}{N})$) change in eigenvalues and rotation of eigenvectors. Since all eigenvectors are orthonormal it follows that $\boldsymbol{u}_k \perp \boldsymbol{w}_k$, c.f.,

$$||\boldsymbol{v}_k||^2 = ||\boldsymbol{u}_k + \delta\boldsymbol{w}_k||^2 = \underbrace{||\boldsymbol{u}_k||^2}_{=1} + \underbrace{\delta^2}_{\approx 0} ||\boldsymbol{w}_k||^2 + 2\delta\boldsymbol{u}_k^T\boldsymbol{w}_k = 1$$
$$\delta\boldsymbol{u}_k^T\boldsymbol{w}_k = 0 \,.$$

We now expand Equation (7) using Equation (5), (8) and (9)

$$\boldsymbol{A}\boldsymbol{v}_k = \nu_k\boldsymbol{v}_k \qquad \Rightarrow$$
$$(\boldsymbol{B} + \delta\boldsymbol{C})(\boldsymbol{u}_k + \delta\boldsymbol{w}_k) = (\lambda_k + \delta\xi_k)(\boldsymbol{u}_k + \delta\boldsymbol{w}_k) \,,$$

ignoring higher order terms of δ gives

$$\boldsymbol{B}\boldsymbol{u}_k + \delta\boldsymbol{C}\boldsymbol{u}_k + \delta\boldsymbol{B}\boldsymbol{w}_k = \lambda_k\boldsymbol{u}_k + \delta\lambda_k\boldsymbol{w}_k + \delta\xi_k\boldsymbol{u}_k \,,$$

Finally, exploiting Equation (6) reduces the above to

$$\boldsymbol{C}\boldsymbol{u}_k + \boldsymbol{B}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k + \xi_k \boldsymbol{u}_k \ . \tag{10}$$

We now look for an estimate of ξ_k by left multiplying with $\boldsymbol{u}_k^T$

$$\boldsymbol{u}_k^T \boldsymbol{C}\boldsymbol{u}_k + \boldsymbol{u}_k^T \boldsymbol{B}\boldsymbol{w}_k = \lambda_k \boldsymbol{u}_k^T \boldsymbol{w}_k + \xi_k \boldsymbol{u}_k^T \boldsymbol{u}_k \ ,$$

using $||\boldsymbol{u}_k||^2 = 1$ and $\boldsymbol{u}_k \perp \boldsymbol{w}_k$ gives

$$\boldsymbol{u}_k^T \boldsymbol{C}\boldsymbol{u}_k + \boldsymbol{u}_k^T \boldsymbol{B}\boldsymbol{w}_k = \xi_k \ ,$$

since $\boldsymbol{B}$ is symmetric, $\boldsymbol{u}_k$ is both a left and right singular vector. Hence, $\boldsymbol{u}_k^T \boldsymbol{B}\boldsymbol{w}_k = \lambda_k \boldsymbol{u}_k^T \boldsymbol{w}_k$=0. Thus finally, it follows that

$$\boldsymbol{u}_k^T \boldsymbol{C}\boldsymbol{u}_k = \xi_k \ . \tag{11}$$

Next, we find an estimate of $\boldsymbol{w}_k$ by left multiplying Equation (10) with $\boldsymbol{u}_j^T \ j \neq k$.

$$\boldsymbol{u}_j^T \boldsymbol{C}\boldsymbol{u}_k + \boldsymbol{u}_j^T \boldsymbol{B}\boldsymbol{w}_k = \lambda_k \boldsymbol{u}_j^T \boldsymbol{w}_k + \xi_k \boldsymbol{u}_j^T \boldsymbol{u}_k \ ,$$

again we exploit the fact that $\boldsymbol{B}$ is symmetric and that $\boldsymbol{u}_j$ is orthogonal to $\boldsymbol{u}_k$, which gives

$$\boldsymbol{u}_j^T \boldsymbol{C}\boldsymbol{u}_k + \lambda_j \boldsymbol{u}_j^T \boldsymbol{w}_k = \lambda_k \boldsymbol{u}_j^T \boldsymbol{w}_k \ . \tag{12}$$

Assuming that span$\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D\}$ = span$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_D\}$, that is, the $\boldsymbol{v}$-basis is a rotation of the $\boldsymbol{u}$-basis, which implies that $\boldsymbol{w}_k$ can be represented as a linear combination of the $\boldsymbol{u}$-vectors (or $\boldsymbol{v}$-vectors), leads to

$$\boldsymbol{w}_k = \sum_{m=1}^{D} h_{km} \boldsymbol{u}_m \ .$$

Due to orthonormality of the eigenvectors, we now realize that $h_{kk} = 0$ and $\boldsymbol{u}_j^T \boldsymbol{w}_k = \boldsymbol{u}_j^T \sum_{m=1}^{D} h_{km} \boldsymbol{u}_m$ will only be non-zero for $m = j$. Hence, Equation (12) reduces to

$$\begin{aligned} \boldsymbol{u}_j^T \boldsymbol{C}\boldsymbol{u}_k + \lambda_j h_{kj} = \lambda_k h_{kj} \qquad &\Rightarrow \\ h_{kj} = \frac{\boldsymbol{u}_j^T \boldsymbol{C}\boldsymbol{u}_k}{\lambda_k - \lambda_j} \ \ k \neq j& \\ h_{kk} = 0 \ .& \end{aligned}$$

In the above we have assumed a nondegenerate system, that is, $\lambda_k \neq \lambda_j \ \forall k \neq j$. Thus, $\boldsymbol{w}_k$ can be expressed as

$$\boldsymbol{w}_k = \sum_{m=1 \neq k}^{N} \frac{\boldsymbol{u}_m^T \boldsymbol{C}\boldsymbol{u}_k}{\lambda_k - \lambda_m} \boldsymbol{u}_m \ , \tag{13}$$

where we used that $\boldsymbol{C}\boldsymbol{u}_k$ is only non-zero for $k \leq N$. We are now ready to return to Equation (8) and (9) inserting the expressions derived for ξ_k and $\boldsymbol{w}_k$ in Equation (11) and (13) respectively:

$$\nu_k = \lambda_k + \delta \boldsymbol{u}_k^T \boldsymbol{C} \boldsymbol{u}_k \tag{14}$$

$$\boldsymbol{v}_k = \boldsymbol{u}_k + \delta \sum_{m=1\neq k}^{N} \frac{(\boldsymbol{u}_{\boldsymbol{m}}^T(\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}))(\boldsymbol{u}_{\boldsymbol{k}}^T(\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}))}{\lambda_k - \lambda_m} \boldsymbol{u}_m \,. \tag{15}$$

Equation (14) shows that the change in eigenvalue is indeed small ($O(\frac{1}{N})$) when going from N to $N-1$ samples. For the eigenvector perturbation, Equation (15), we can bound the squared length of the sum and obtain a similar result,

$$\begin{aligned}
\left|\left| \frac{1}{N} \sum_{m=1\neq k}^{N} \frac{(\boldsymbol{u}_{\boldsymbol{m}}^T(\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}))(\boldsymbol{u}_{\boldsymbol{k}}^T(\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}))}{\lambda_k - \lambda_m} \boldsymbol{u}_m \right|\right|^2 &\leq \\
\frac{1}{N^2} ||\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}||^2 \left|\left| \sum_{m=1\neq k}^{N} \frac{(\boldsymbol{u}_{\boldsymbol{m}}^T(\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}))}{\lambda_k - \lambda_m} \boldsymbol{u}_m \right|\right| &= \\
\frac{1}{N^2} ||\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}||^2 \sum_{m=1\neq k}^{N} \frac{|(\boldsymbol{u}_{\boldsymbol{m}}^T(\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}))|^2}{|\lambda_k - \lambda_m|^2} &\leq \\
\frac{1}{N^2} \frac{2||\boldsymbol{x}_N - \boldsymbol{\mu}_{N-1}||^4}{|\Delta\lambda_k|^2} &,
\end{aligned}$$

where $\Delta\lambda_k$ is the spacing between the k'th eigenvalue and the closest neighbor, and the factor of two compensates for the missing k'th term in the sum, that is, the perturbation is of order $O(1/N)$

References

Michael Biehl and Andreas Mietzner. Statistical mechanics of unsupervised structure recognition. *Journal of Physics A-Mathematical and General*, 27(6):1885–1897, 1994.

Gilles Blanchard, Olivier Bousquet, and Laurent Zwald. Statistical properties of kernel principal component analysis. *Machine Learning*, 66(2-3):259–294, 2007.

Mikio L. Braun, Joachim M. Buhmann, and Klaus-Robert Müller. On relevant dimensions in kernel feature spaces. *Journal of Machine Learning Research*, 9:1875–1908, 2008.

Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing*. 1977. ISBN 0-201-02596-5 (hardcover), 0-201-02597-3 (paperback).

David C. Hoyle and Magnus Rattray. A statistical mechanics analysis of gram matrix eigenvalue spectra. In *Lecture Notes in Computer Science, 17th Annual Conference on Learning Theory*, volume 3120, pages 579–593. Springer Verlag, 2004a.

David C. Hoyle and Magnus Rattray. Limiting form of the sample covariance eigenspectrum in pca and kernel pca. In *Advances in Neural Information Processing Systems 16*, pages 16–23. MIT Press, 2004b.

David C. Hoyle and Magnus Rattray. Principal-component-analysis eigenvalue spectra from data with symmetry-breaking structure. *Physical Review E*, 69(2):026124, 2004c.

David C. Hoyle and Magnus Rattray. Statistical mechanics of learning multiple orthogonal signals: Asymptotic theory and fluctuation effects. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 75(1):016101, 2007.

Jonathan J . Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

Robert Jenssen, Torbjörn Eltoft, Deniz Erdogmus, and Jose C. Principe. Some equivalences between kernel methods and information theoretic methods. *Journal of VLSI Signal Processing*, 45:49–65, 2006.

Iain M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Annals of Statistics*, 29(2):295–327, 2001.

Ulrik Kjems, Lars K. Hansen, and Stephen C. Strother. Generalizable singular value decomposition for ill-posed datasets. In *Advances in Neural Information Processing Systems 13*, pages 549–555. MIT Press, 2001.

Rudy Moddemeijer. On estimation of entropy and mutual information of continuous distributions. *Signal Processing*, 16(3):233–246, 1989.

Sofia Mosci, Lorenzo Rosasco, and Alessandro Verri. Dimensionality reduction and generalization. In *Proceedings of the 24th International Conference on Machine Learning*, pages 657–664, 2007.

Peter Reimann, Chris Van den Broeck, and Geert J. Bex. A Gaussian scenario for unsupervised learning. *Journal of Physics A - Mathematical and General*, 29(13):3521–3535, 1996.

Bernhard Schölkopf, Alex Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

John Shawe-Taylor and Christopher K. I. Williams. The stability of kernel principal components analysis and its relation to the process eigenspectrum. In *Advances in Neural Information Processing Systems 15*, pages 367–374. MIT Press, 2003.

Jack W. Silverstein and Patrick L. Combettes. Signal-detection via spectral theory of large dimensional random matrices. *IEEE Transactions on Signal Processing*, 40(8):2100–2105, 1992.

Laurent Zwald and Gilles Blanchard. On the convergence of eigenspaces in kernel principal component analysis. In *Advances in Neural Information Processing Systems 18*, pages 1649–1656. MIT Press, 2006.

Journal of Machine Learning Research 12 (2011) 2045-2094 Submitted 3/10; Revised 1/11; Published 6/11

Exploiting Best-Match Equations for Efficient Reinforcement Learning

Harm van Seijen HARM.VANSEIJEN@TNO.NL
Distributed Sensor Systems Group
TNO Defence, Security and Safety
P.O. Box 96864
2509 JG, The Hague, The Netherlands

Shimon Whiteson S.A.WHITESON@UVA.NL
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands

Hado van Hasselt H.VAN.HASSELT@CWI.NL
Multi-agent and Adaptive Computation Group
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

Marco Wiering MWIERING@AI.RUG.NL
Department of Artificial Intelligence
University of Groningen
Groningen, The Netherlands

Editor: Peter Dayan

Abstract

This article presents and evaluates *best-match learning*, a new approach to reinforcement learning that trades off the sample efficiency of model-based methods with the space efficiency of model-free methods. Best-match learning works by approximating the solution to a set of *best-match equations*, which combine a sparse model with a model-free Q-value function constructed from samples not used by the model. We prove that, unlike regular sparse model-based methods, best-match learning is guaranteed to converge to the optimal Q-values in the tabular case. Empirical results demonstrate that best-match learning can substantially outperform regular sparse model-based methods, as well as several model-free methods that strive to improve the sample efficiency of temporal-difference methods. In addition, we demonstrate that best-match learning can be successfully combined with function approximation.

Keywords: reinforcement learning, on-line learning, temporal-difference methods, function approximation, data reuse

1. Introduction

In *reinforcement learning* (RL) (Kaelbling et al., 1996; Sutton and Barto, 1998), an agent seeks an optimal control policy for a sequential decision problem in an unknown environment. Unlike in supervised learning, the agent never sees examples of correct or incorrect behavior. Instead, it receives only positive and negative rewards for the actions it tries. Its goal is to maximize the expected *return*, which is the cumulative discounted reward. When the sequential decision problem is modeled as a *Markov decision process* (MDP), the agent's policy can be represented as a mapping from each state it may encounter to a probability distribution over the available actions.

There are several approaches for learning the optimal policy of an MDP. *Model-free*, or direct, methods find an optimal policy by using sample experience to directly update the *state values*, which predict the return when following a specified policy, or the *state-action values*, or *Q-values*, which predict the return when taking an action in a certain state and following a specified policy thereafter. Once the optimal state or state-action values have been found, the optimal policy can easily be constructed. A popular model-free approach is *temporal-difference* (TD) learning (Sutton, 1988), which bootstraps value estimates from other values using updates based on the *Bellman equations* (Bellman, 1957). Temporal-difference methods such as Q-learning (Watkins, 1989) and Sarsa (Rummery and Niranjan, 1994; Sutton, 1996) require only $O(|\mathcal{S}||\mathcal{A}|)$ space and are guaranteed to find optimal policies in the limit. However, they often need prohibitively many samples in practice.

Alternatively, *model-based*, or indirect, methods (Sutton, 1990; Moore and Atkeson, 1993; Brafman and Tennenholtz, 2002; Kearns and Singh, 2002; Strehl and Littman, 2005; Diuk et al., 2009) use sample experience to estimate a model of the MDP and then compute the optimal values using this model via off-line planning techniques such as *dynamic programming* (Bellman, 1957). Because the sample experience gathered by the agent is incorporated into the model, it is reused throughout learning. As a result, some model-based methods can find approximately optimal policies with high probability using only a polynomial number of samples (Brafman and Tennenholtz, 2002; Kearns and Singh, 2002; Strehl and Littman, 2005). However, representing the model requires $O(|\mathcal{S}|^2|\mathcal{A}|)$ space, which can be prohibitive in problems with large state spaces.

To avoid this limitation, methods can learn smaller, approximate models that require only a fraction of the space used by full model-based methods. Kearns and Singh (1999) show that, when using such sparse models, it is still possible to learn probably approximately correct policies. However, the performance of such methods is bounded by the quality of the model approximation. Furthermore, since the models may remain incorrect regardless of how much sample experience is gathered, such methods are not guaranteed to find optimal policies even in the limit.

In this article, we present and evaluate *best-match learning*, a new approach for trading off the strengths of model-based and model-free methods. Best-match learning works by approximating the solution to a set of *best-match equations*, which combine a sparse model with a model-free Q-value function constructed from samples not used by the model. We prove that, unlike regular sparse model-based methods, best-match learning is guaranteed to converge to the optimal policy in the tabular case. This guarantee holds even when using a *last-visit model* (LVM), which stores only the last observed reward and transition state for each state-action pair.

In addition, we present an extensive empirical analysis, comparing the performance of best-match learning to several algorithms with similar space requirements. These results demonstrate that best-match learning can outperform regular sparse model-based methods, as well as several model-free methods that strive to improve the sample efficiency of traditional TD methods. These include *eligibility traces* (Sutton, 1988; Watkins, 1989), which update recently visited states in proportion to a trace parameter; *experience replay* (Lin, 1992), which stores experience sequences and uses them for repeated TD updates; and *delayed Q-learning* (Strehl et al., 2006), which uses optimistic Q-value estimates to follow an approximately correct policy except for $O(|\mathcal{S}||\mathcal{A}|\log(|\mathcal{S}||\mathcal{A}|))$ timesteps.

The rest of this article is organized as follows. Section 2 formally defines the RL problem and summarizes some basic theoretical results. As a conceptual stepping stone, Section 3 presents *just-in-time Q-learning*, which postpones updates until the moment of revisit of the corresponding state. We prove that, although just-in-time Q-learning performs the same number of updates as regular Q-learning, the Q-values used in its update targets generally have received more updates. Thus, it can improve performance without extra computation.

Section 4 extends the idea of using improved update targets to best-match learning with an LVM, in which updates are continually revised such that the update targets constructed from them are more accurate. We show that best-match LVM learning is related to eligibility traces, by proving that under certain conditions they compute the same values. However, we also show that in arbitrary MDPs best-match LVM learning, unlike eligibility traces, performs updates that are unbiased with respect to initial state values. We demonstrate empirically that, as a result, it can substantially outperform TD(λ) despite using similar space and computation.

Section 4 also addresses the control case. We propose an efficient best-match LVM algorithm that uses *prioritized sweeping* (Moore and Atkeson, 1993), a well-known technique for prioritizing model-based updates, to trade off extra computation for improved performance. We prove that, despite the use of a sparse model, this approach converges to the optimal Q-values under the same conditions as Q-learning. In addition, we demonstrate empirically that it can substantially outperform competitors with similar space requirements.

Section 5 proposes a best-match learning algorithm that uses an *n-transition model* (NTM), which maintains an estimate of the transition probability for n transition states per state action pair. By tuning n, the space requirements can be controlled. We prove that the algorithm converges to the optimal Q-values for any value of n. We demonstrate empirically the resulting performance improvement over regular sparse model-based methods with equal space requirements, whose performance is bounded by the quality of the model approximation.

Section 6 proposes *best-match function approximation*, which demonstrates that best-match learning is useful beyond the tabular case. In particular, we combine best-match learning with gradient-descent function approximation and show empirically that it can outperform Sarsa(λ) and experience replay with linear function approximation while using similar computation.

Section 7 discusses the article's theoretical and empirical results, Section 8 outlines future work, and Section 9 concludes.

2. Background

Sequential decision problems are often formalized as *Markov decision processes* (MDPs), which can be described as 4-tuples $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ consisting of $\mathcal{S}$, the set of all states; $\mathcal{A}$, the set of all actions; $\mathcal{P}_{sa}^{s'} = P(s'|s,a)$, the transition probability from state $s \in \mathcal{S}$ to state s' when action $a \in \mathcal{A}$ is taken; and $\mathcal{R}_{sa} = E(r|s,a)$, the reward function giving the expected reward r when action a is taken in state s. Actions are selected at discrete timesteps $t = 0, 1, 2, ...$ and r_{t+1} is defined as the reward received after taking action a_t in state s_t at timestep t. An optimal policy π^* is a mapping from $\mathcal{S}$ to $\mathcal{A}$ that maximizes the expected discounted return

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,,$$

where γ is a discount factor with $0 \leq \gamma \leq 1$.

Most solution methods are based on estimating a value function $V^{\pi}(s)$, which gives the expected return when the agent is in state s and follows policy π, or an action-value function $Q^{\pi}(s,a)$, which gives the expected return when the agent takes action a in state s and follows policy π thereafter.

In the control case, TD methods seek to learn the optimal action-value function $Q^*(s,a)$, which is the solution to the Bellman optimality equations (Bellman, 1957):

$$Q^*(s,a) = \mathcal{R}_{sa} + \gamma \sum_{s'} \mathcal{P}_{sa}^{s'} \max_{a'} Q^*(s',a') \, .$$

By iteratively updating the current estimate $Q_t(s,a)$ each time new experience is obtained, TD methods seek to approximate this function. A common form for these updates is

$$Q_{t+1}(s_t,a_t) \leftarrow (1-\alpha)Q_t(s_t,a_t) + \alpha \upsilon_t \, ,$$

where α is the learning rate and υ_t is the update target. Many update targets are possible, such as the Q-learning (Watkins and Dayan, 1992) update target

$$\upsilon_t = r_{t+1} + \gamma \max_a Q_t(s_{t+1},a) \, .$$

Once the optimal action-value function has been learned, an optimal policy can be derived by taking the greedy action with respect to this function.

Alternatively, the agent can take a model-based approach (Sutton, 1990; Moore and Atkeson, 1993), in which its experience is used to compute maximum-likelihood estimates of $\mathcal{P}$ and $\mathcal{R}$. Using this model, the agent can compute Q (or the value function V) using dynamic programming methods (Bellman, 1957) such as value iteration (Puterman and Shin, 1978). Each time new experience is gathered, the model is updated and Q recomputed.

In the control case, the agent faces the *exploration-exploitation dilemma*. The agent can either exploit its current knowledge by taking the action that predicts the highest expected return given current estimates, or it can explore by taking a different action in order to improve the accuracy of the Q-value of that action.

Related to the control case is the *policy evaluation* case. In this case, the goal is to estimate the value function $V^{\pi}(s)$ belonging to policy π. TD methods iteratively improve the current estimate, $V_t(s)$ each time new experience is obtained using the update rule

$$V_{t+1}(s_t) \leftarrow (1-\alpha)V_t(s_t) + \alpha \upsilon_t \, .$$

An example of an update target for policy evaluation is the TD(0) update target

$$\upsilon_t = r_{t+1} + \gamma V_t(s_{t+1}) \, .$$

3. Just-In-Time Q-Learning

In this section we present just-in-time (JIT) Q-learning, whose underlying principles form a stepping stone towards best-match learning (introduced in Section 4). Like other *lazy learning* methods, for example, Atkeson et al. (1997), JIT Q-learning postpones updates until they are needed. Wiering and Schmidhuber (1998) showed that by postponing updates a computationally efficient version of Q(λ) can be constructed that does not rely on placing a bound on the trace length. We prove that by postponing Q-learning updates until a state is revisited, the update targets involved receive in general more updates, while the total number of updates of the current state stays the same. Empirically, we demonstrate that this leads to a performance gain under a range of settings at similar computational cost.

When a Q-learning update is postponed, the values on which the update target is based are from a more recent timestep. This is advantageous, since Q-learning updates cause the expected error in the values to decrease over time (Watkins and Dayan, 1992) and therefore more recent values will be on average more accurate. However, postponing the update of a value for too long can negatively affect performance, since a value that has not been updated might be used for action selection or for bootstrapping other values. We start by showing that updates can be postponed until their corresponding states are revisited, without negatively affecting performance.

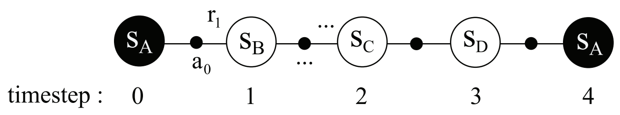


Figure 1: A state transition sequence in which the initial state s_A is revisited at timestep 4. The small black dots in between states represent actions.

Consider the state-action sequence in Figure 1. State s_A is visited at timestep 0 and revisited at timestep 4. With the regular Q-learning update, the Q-value of state-action pair (s_A, a_0) gets updated at timestep 1:

$$Q_1(s_A, a_0) = (1-\alpha)Q_0(s_A, a_0) + \alpha\left[r_1 + \gamma \max_a Q_0(s_B, a)\right],$$

while at timesteps $2-4$ no update of (s_A, a_0) occurs, and therefore $Q_4(s_A, a_0) = Q_1(s_A, a_0)$. The update of the Q-value of (s_A, a_0) at timestep 1 can be considered premature, since the earliest use of its value is in the update target for (s_D, a_3), which uses $Q_3(s_A, a_0)$. Therefore, the update of the Q-value of (s_A, a_0) can be postponed until at least timestep 3 without negatively affecting the update target for (s_D, a_3). When the update of (s_D, a_3) is also postponed, the earliest use of the Q-value of (s_A, a_0) occurs at timestep 4, where it is used for action selection. Thus, if we postpone the update of all state-action pairs, the update of the Q-value of (s_A, a_0) can be postponed until the timestep of its revisit, without causing dependent state values or the action selection procedure to use a value of (s_A, a_0) that has not been updated. We call this type of update a *just-in-time update*, since the update is postponed until just before the updated value is needed.

To denote the Q-values resulting from just-in-time updates we use $\tilde{Q}$ throughout this section. With just-in-time updates, no updates of (s_A, a_0) occur at timesteps 1-3, so $\tilde{Q}_3(s_A, a_0) = \tilde{Q}_0(s_A, a_0)$. Instead, an update occurs when s_A is revisited:

$$\tilde{Q}_4(s_A, a_0) = (1-\alpha)\tilde{Q}_3(s_A, a_0) + \alpha\left[r_1 + \gamma \max_a \tilde{Q}_3(s_B, a)\right].$$

The regular and just-in-time update for (s_A, a_0) can be written in a more similar form by expressing the value at timestep 4 in terms of the value at timestep 0:

$$\begin{aligned} Q_4(s_A, a_0) &= (1-\alpha)Q_0(s_A, a_0) + \alpha[r_1 + \gamma \max_a Q_0(s_B, a)], \\ \tilde{Q}_4(s_A, a_0) &= (1-\alpha)\tilde{Q}_0(s_A, a_0) + \alpha[r_1 + \gamma \max_a \tilde{Q}_3(s_B, a)]. \end{aligned} \quad (1)$$

This formulation highlights the difference between the two update types. At timestep 4, under both update schemes, the Q-value of (s_A, a_0) has received one update based on the same experience sample. However, a just-in-time update uses the most recent value of the Q-values of s_B, while a regular update uses the value at the timestep of the initial visit of s_A. By defining t^* as the timestep of the previous visit of state s_t, we can write the two update types more generally as

$$\begin{aligned} Q_t(s_t, a_{t^*}) &= (1-\alpha)Q_{t^*}(s_t, a_{t^*}) + \alpha[r_{t^*+1} + \gamma \max_a Q_{t^*}(s_{t^*+1}, a)], \quad (2)\\ \tilde{Q}_t(s_t, a_{t^*}) &= (1-\alpha)\tilde{Q}_{t^*}(s_t, a_{t^*}) + \alpha[r_{t^*+1} + \gamma \max_a \tilde{Q}_{t-1}(s_{t^*+1}, a)]. \quad (3) \end{aligned}$$

Note that we express the update target using only values from the past, making an implementation easier to interpret. Note also that while $s_t = s_{t^*}$ per definition (because s_t is revisited), s_{t^*+1} does not have to be equal to s_{t+1}, since the state transition from s_t can be stochastic. Also, a_{t^*} is in general not equal to a_t.

When comparing the two update targets in more detail, two cases can be distinguished. See Figure 2 for an example of each case. In the first case, state s_B is not revisited before the revisit of state s_A. In this case, neither update type makes use of an updated Q-value for s_B in the update target for s_A. The regular update does not since it uses the values of s_B at timestep t^*, and the just-in-time update does not since s_B is not revisited and therefore no update has occurred yet at timestep $t-1$. In the second case, state s_B has been revisited before the revisit of s_A. The regular update still uses the value of s_B from timestep t^* and therefore does not use an updated value. The just-in-time update on the other hand does use an updated value, since this update occurred at the revisit of s_B. Note that for a returning action ($t^* = t-1$), both update types have exactly the same form and this can therefore be treated as an example of case 1. From these two cases, we can deduce the following theorem, which is proven in Appendix A.

Theorem 1 *Given the same experience sequence, each Q-value from the current state has received the same number of updates using JIT updates (Equation 3) as using regular updates (Equation 2). However, each Q-value in the update target of a JIT update has received an equal or greater number of updates as in the update target of the corresponding regular update.*

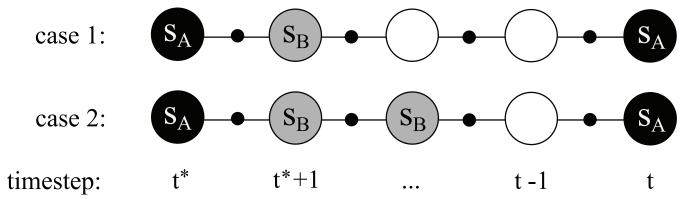


Figure 2: Two cases in which state s_A is revisited. In the first case, neither a regular update nor a just-in-time update make use of an updated value for s_B in the update target of s_A, while in the second case a just-in-time update does.

Algorithm 1 shows pseudocode for the implementation of just-in-time (JIT) Q-learning. The agent stores the reward and transition state received upon the last visit of a state, that is, the *last-visit sample*, in $R'(s)$ and $S'(s)$ respectively, while the action taken at the last visit of a state is stored

Algorithm 1 JIT Q-Learning

```
1: initialize $Q(s,a)$ arbitrarily for all $s$,$a$
2: initialize $S'(s) = \emptyset$ for all $s$
3: loop {over episodes}
4:    initialize s
5:    repeat {for each step in the episode}
6:       if $S'(s) \neq \emptyset$ then
7:          $Q(s,a) \leftarrow (1-\alpha^{sa}) \cdot Q(s,a) + \alpha^{sa} [R'(s) + \gamma \max_{a'} Q(S'(s),a')]$    $//a = A(s)$
8:       end if
9:       select action $a$, based on $Q(s,\cdot)$
10:      take action $a$, observe $r$ and $s'$
11:      $S'(s) \leftarrow s'$; $R'(s) \leftarrow r$; $A(s) \leftarrow a$
12:      $s \leftarrow s'$
13:   until $s$ is terminal
14: end loop
```

in $A(s)$. If $S'(s) = \emptyset$, state s has not been visited yet and no update can be performed. Note that the last-visit sample is not reset at the end of an episode, but maintained across episodes.

Because JIT Q-learning uses more recent values in its update targets than regular Q-learning, we expect a performance improvement over regular Q-learning. We test this hypothesis by comparing the performance of JIT Q-learning with regular Q-learning on the Dyna Maze task (Sutton, 1990). In this navigation task, depicted in Figure 3, the agent has to find its way from start to goal. The agent can choose between four movement actions: up, down, left and right. All actions result in 0 reward, except for when the goal is reached, which results in a reward of +1. The discount factor γ is set to 0.95. We use a deterministic as well as a stochastic environment to test the generality of the hypothesis. In the stochastic version, we employ a probabilistic transition function: with a 20% probability, the agent moves in an arbitrary direction instead of the direction corresponding to the action.

To compare performance, we measure the average return each method accrues from the start state during the first 100 episodes in the deterministic case, averaged over 5000 independent runs per method. For the stochastic version, we measure the return during the first 200 episodes. Each method uses ϵ-greedy action selection with $\epsilon = 0.1$. In the deterministic case, we use a constant learning rate of 1, while in the stochastic case we use an initial learning rate α_0 of 1 that is decayed in the following manner:[1]

$$\alpha^{sa} = \frac{\alpha_0}{d \cdot [n(s,a) - 1] + 1}, \tag{4}$$

where $n(s,a)$ is the total number of times action a has been selected in state s. Note that for $d = 0$, $\alpha^{sa} = \alpha_0$, while for $d = 1$, $\alpha^{sa} = \alpha_0 / n(s,a)$. We optimize the learning rate decay d between 0 and 1 by taking the decay rate with the maximum average return over the measured number of episodes. We use two different initialization schemes for the Q-values to determine whether the performance difference depends on initialization. We use optimistic initialization, by initializing the Q-values to 20, and pessimistic initialization, by setting the Q-values to 0.

1. This decay is similar to the more common form $\frac{c_1}{c_2 + n(s,a)}$, but with the free parameters re-arranged.

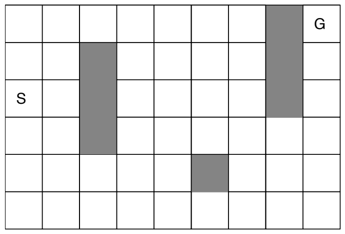


Figure 3: The Dyna Maze task, in which the agent must travel from *S* to *G*. The reward is +1 when the goal state is reached and 0 otherwise.

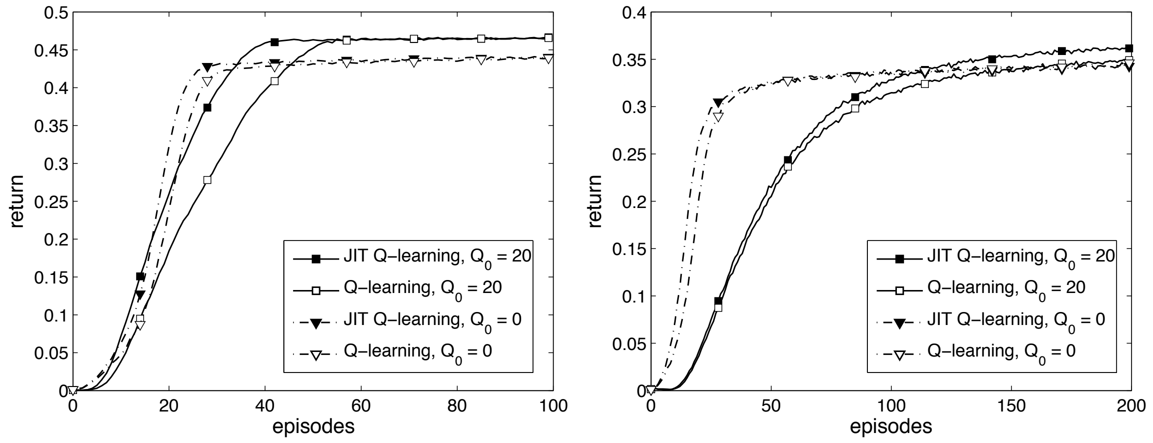


Figure 4: Comparison of the performance of JIT Q-learning and regular Q-learning on the deterministic (left) and stochastic (right) Dyna Maze task for two different initialization schemes.

	deterministic - 100 eps.			stochastic - 200 eps.		
	d	average return	standard error	d	average return	standard error
Q-learning, $Q_0 = 0$	0	0.3506	0.0004	1.0	0.3039	0.0003
JIT Q-learning, $Q_0 = 0$	0	0.3628	0.0004	1.0	0.3083	0.0003
Q-learning, $Q_0 = 20$	0	0.3438	0.0002	0.005	0.2562	0.0002
JIT Q-learning, $Q_0 = 20$	0	0.3714	0.0002	0.010	0.2674	0.0002

Table 1: The performance of JIT Q-learning and regular Q-learning on the Dyna Maze task and the optimal learning rate decay *d*.

Figure 4 plots the return as a function of the number of episodes, while Table 1 shows the average return and optimal learning rate. The computation time for both methods was similar. JIT Q-learning outperforms regular Q-learning in the deterministic as well as the stochastic environment and for both types of initialization, although not always by a large margin. This confirms our intuition that, since JIT Q-learning uses values from a later time which are in general more accurate, a performance benefit is gained over regular Q-learning in a broad range of settings. The performance benefit in the deterministic case can be explained by exploration, which causes the order in which states are visited to change despite the deterministic state transitions.

4. Best-Match Last-Visit Model

In this section, we demonstrate that updates can be postponed much further than is done by JIT Q-learning, without negatively affecting other updates, when *best-match updates* are performed. Best-match updates are updates that can correct previous updates when more recent information becomes available. This insight leads to the derivation of the *best-match last-visit model equations*, which combine a *last-visit model* (LVM), consisting of the last experienced reward and transition state for each state-action pair, with *model-free Q-values*, constructed from model-free updates of all observed samples, except the ones stored in the LVM. We present an evaluation as well as a control algorithm based on solving these equations and empirically demonstrate that these methods can outperform competitors with similar space requirements.

4.1 Best-Match LVM Equations

In the example presented in Section 3, the update of $Q(s_A, a_0)$ is postponed until state s_A is revisited. In this section, we demonstrate that the update can be postponed even further in the case that a different action is selected upon revisit. Since we will consider multiple updates per timestep in this section, we denote the Q-value function using two iteration indices: t and i. Each time an update occurs, i is increased, while each time an action is taken, t is increased and i is reset to 0. Therefore, if I denotes the total number of updates that occurs at time t, by definition $Q_{t,I} = Q_{t+1,0}$. Action selection at time t is based on $Q_{t,I}$. Using this convention, the regular Q-learning update can be written as

$$Q_{t+1,1}(s_t, a_t) = (1-\alpha)Q_{t+1,0}(s_t, a_t) + \alpha[r_{t+1} + \max_{a'} Q_{t+1,0}(s_{t+1}, a')] \,.$$

Now consider the example shown in Figure 5, which extends Figure 1 to include a second revisit of s_0 at timestep $t = 7$. Suppose that a different action is selected on the first revisit, that is, $a_4 \neq a_0$. Using just-in-time updates, the Q-value of state-action pair (s_A, a_0) gets updated at time $t = 4$. Using the two indices convention we can rewrite Equation 1 as[2]

$$Q_{4,1}(s_A, a_0) = (1-\alpha)Q_{1,0}(s_A, a_0) + \alpha[r_1 + \gamma \max_a Q_{4,0}(s_B, a)] \,. \tag{5}$$

To perform this update, the experience set (r_1, s_B) resulting from taking action a_0 in s_A is temporarily stored. With JIT Q-learning, this experience is stored per state. If the state is revisited and a new action is taken, the previous experience is overwritten and lost. However, if the experience is stored per state-action pair, then the previous experience is not overwritten until the same action is selected again. If the same action is not selected upon revisit, the experience can be used again

2. We use Q now instead of $\tilde{Q}$, since the only purpose of the tilde was to distinguish it from the Q-values of regular Q-learning.

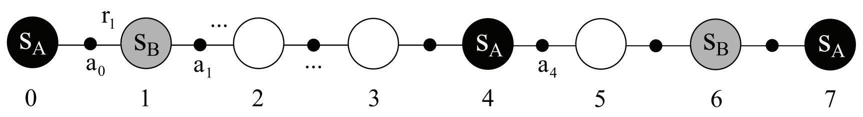


Figure 5: A state transition sequence in which best-match updates can enable further postponing. Timesteps are shown below each state.

to redo the update at a later time, using more recent values for the next state. In the example from Figure 5, the update of (s_A, a_0) can be redone at timestep 7:

$$Q_{7,1}(s_A, a_0) = (1-\alpha)Q_{1,0}(s_A, a_0) + \alpha[r_1 + \gamma \max_a Q_{7,0}(s_B, a)]\,. \tag{6}$$

Since state s_B is revisited at timestep 6, (s_B, a_1) has received an extra update and therefore $Q_{7,0}(s_B, a_1)$ is likely to be more accurate than $Q_{4,0}(s_B, a_1)$.

Equation 6 is not equivalent to a (postponed) Q-learning update, in contrast to Equation 5, since $Q_{1,0}(s_A, a_0)$ is not equal to $Q_{7,0}(s_A, a_0)$ due to the update at timestep 4. Equation 6 corrects the update from timestep 4, by redoing it using the most recent Q-values for the update target. We call this update a *best-match update* (this name will be explained later in the section), while we call $Q_{1,0}(s_A, a_0)$ the *model-free Q-value* of (s_A, a_0).

Before formally defining a best-match update, we define the last-visit experience and the model-free Q-values.

Definition 2 *The last-visit experience of state-action pair (s,a) denotes the last-visit reward, $R'_t(s,a)$, that is, the reward received upon the last visit of (s,a), and the last-visit transition state, $S'_t(s,a)$, that is, the state transitioned to upon the last visit of (s,a). For a state-action pair that has not yet been visited, we define $R'_t(s,a) = \emptyset$ and $S'_t(s,a) = \emptyset$.*

The LVM consists of the last-visit experience from all state-action pairs.

Definition 3 *The model-free Q-value of a state-action pair (s,a), $Q^{mf}_t(s,a)$, is a Q-value that has received updates from all observed samples except those stored in the LVM, that is, $R'_t(s,a)$ and $S'_t(s,a)$. For a state-action pair that has not yet been visited, we define $Q^{mf}_t(s,a) = Q_{0,0}(s,a)$.*

While Q can be updated multiple times per timestep, Q^{mf} is updated only once per timestep. Therefore, it is uses a single time index t. We define a best-match update as:

Definition 4 *A best-match update combines the model-free Q-value of a state-action pair with its last-visit experience from the same timestep according to*

$$Q_{t,i+1}(s,a) = (1-\alpha)Q^{mf}_t(s,a) + \alpha[R'_t(s,a) + \gamma \max_{a'} Q_{t,i}(S'_t(s,a), a')]\,.$$

Using best-match updates to extend the postponing period of a sample update requires additional computation, as the agent typically performs multiple best-match updates per timestep. In the example, at timestep 7 the agent redoes the update of $Q(s_A, a_0)$, but also performs an update of $Q(s_A, a_4)$.

The model-free Q-value function is updated only once per timestep. Specifically, at timestep $t+1$ Q^{mf} is updated according to

$$Q^{mf}_{t+1}(s_t,a_t) = Q_{t+1,0}(s_t,a_t)\,. \tag{7}$$

Assuming (s_t,a_t) has received a best-match update at timestep t, Equation 7 is equivalent to the update

$$Q^{mf}_{t+1}(s_t,a_t) = (1-\alpha)Q^{mf}_t(s_t,a_t) + \alpha[R'_t(s_t,a_t) + \gamma \max_{a'} Q_{t,i}(S'_t(s_t,a_t),a')]\,,$$

where the value of i depends on the order of best-match updates at timestep t. After Q^{mf} has been updated, the last-visit experience for (s_t,a_t) is overwritten with the new experience

$$\begin{aligned} R'_{t+1}(s_t,a_t) &= r_{t+1}\,, \\ S'_{t+1}(s_t,a_t) &= s_{t+1}\,. \end{aligned}$$

In the approach described above, best-match updates are used to postpone the update from a sample without negatively affecting other updates or the action selection process. However, best-match updates can be exploited far beyond simply avoiding these negative effects. As an example, consider the state-action sequence in Figure 6. s_B is not revisited before the revisit of s_A. With the update strategy described above, best-match updates occur only when a state is revisited. Consequently, the experience from (s_B,a_1) is not used in the update target of (s_A,a_0). However, it is not necessary to wait for a revisit of s_B to perform a best-match update. Instead, it can be performed at the moment it is needed: when s_A is revisited. Thus, if at timestep 3 the agent performs a best-match update of $Q(s_B,a_1)$, before updating $Q(s_A,s_0)$, the latter update will exploit more recent Q-values for s_B, just as if s_B had been revisited.

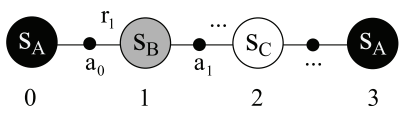


Figure 6: A state transition sequence in which s_B is not revisited. Timesteps are shown below each state.

Taking this idea further, the agent can first update the Q-values of s_C before updating the Q-values of s_B. In other words, the agent uses the Q-values of s_A to perform a best-match update of s_C, then performs a best-match update of s_B and finally updates s_A. However, once the Q-values of s_A have changed, it is possible to further improve the Q-values of s_C by performing a new best-match update. The new Q-values of s_C can then be used to redo the update of s_B, which in turn can be used to re-update s_A. This process can repeat until the Q-values reach a fixed point, which is the solution to a system of $|\mathcal{S}||\mathcal{A}|$ *best-match LVM equations*. We call this solution the *best-match Q-value function*, Q^B, which forms the best match between the LVM and the model-free Q-values.

Definition 5 *The best-match LVM equations at timestep t are defined as*

$$Q^B_t(s,a) = \begin{cases} (1-\alpha^{sa}_t)Q^{mf}_t(s,a) + \alpha^{sa}_t\,[R'_t(s,a) + \gamma \max_c Q^B_t(S'_t(s,a),c)] & \textit{if}\, S'_t(s,a) \neq \emptyset \\ Q^{mf}_t(s,a) & \textit{if}\, S'_t(s,a) = \emptyset\,. \end{cases}$$

There are different ways to look at these equations. One way is to see them as the limit case of redoing updates using (in general) increasingly more accurate update targets. Another way is to see them as Bellman optimality equations based on an induced model. For state-action pair (s,a) this induced model can be described as a transition with probability α to state $S'(s,a)$ with a reward of $R'(s,a)$ and a transition with probability $1-\alpha$ to a terminal state s_T (with a value of 0) and a reward of $Q^{mf}(s,a)$ (see Figure 7).[3]

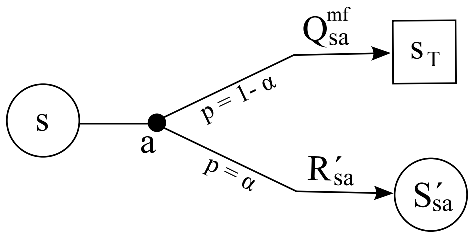


Figure 7: Illustration of the induced model for state-action pair (s,a) corresponding with the best-match LVM equations. The small black dot represents the stochastic action a leading with probability α to state $S'(s,a)$ and with probability 1-α to state s_T.

The advantage of solving the Bellman optimality equations for this induced model, compared to solving it using only the LVM, is that the bias towards the samples in the LVM can be controlled using the learning rates. With annealing learning rates, the transition probability to $S'_t(s,a)$ is decreased over time in favor of transition to the terminal state. On the other hand, when using only the LVM, the solution of the Bellman equations depends only on the samples of the LVM and does not take into account any previous samples. Clearly, in a stochastic environment, this will lead to a sub-optimal policy. Also when the solution is not computed exactly, but approximated by only performing a finite number of updates at each timestep (which is the case for any practical algorithm), using the induced model leads to a better performance, because of the strong bias towards the most recent samples that occurs when using only the LVM.

Section 4.3 discusses how to solve the best-match equations. However, we first discuss the policy evaluation case, for which analogous equations can be defined.

Definition 6 *The best-match LVM equations for state values at time t are*

$$V_t^B(s) = \begin{cases} (1-\alpha_t^s)V_t^{mf}(s) + \alpha_t^s\,[R'_t(s) + \gamma V_t^B(S'_t(s))] & \text{if } S'_t(s) \neq \emptyset \\ V_t^{mf}(s) & \text{if } S'_t(s) = \emptyset\,. \end{cases}$$

The model-free state values are updated according to $V_{t+1}^{mf}(s_t) = V_{t+1,0}(s_t)$.

While in general the value function V can be seen as a special case of the action-value function Q (with all states only having a single action), V has a linear set of best-match equations, in contrast to Q, a property we exploit in best-match LVM evaluation.

3. We assume $S'_t(s,a) \neq \emptyset$ for (s,a) in this case.

4.2 Best-Match LVM Evaluation

In the evaluation case, the best-match LVM equations form a linear set that can be solved exactly. This section proposes an algorithm that does so in a computationally efficient way, using updates that are unbiased with respect to the initial state values.

The algorithm is based on two observations. First, not all $|\mathcal{S}|$ best-match equations necessarily depend on each other. The subset of equations needed to compute the best-match value for s_t can be found by iterating through the sequence of last-visit transition states, starting with $S'(s_t)$. The corresponding N best-match equations form the linear set of equations to solve. For readability, we write s_t as $s_{[0]}$ and use the notation $s_{[n]} = S'(s_{[n-1]})$ and $r_{[n]} = R'(s_{[n-1]})$ for the subsequent transition state and reward. In addition, we use $\alpha^{[n]}$ for $\alpha^{s_{[n]}}$. The equations can now be written as

$$V^B(s_{[n]}) = (1-\alpha^{[n]})V^{mf}(s_{[n]}) + \alpha^{[n]}\left[r_{[n+1]} + \gamma V^B(s_{[n+1]})\right], \qquad \text{for all } n \in [0, N-1].$$

Second, the last state of this sequence, $s_{[N]}$, is always either a terminal state or the current state. Furthermore, none of the intermediate states can appear twice, making the N equations independent. This can be proven by contradiction. First, assume that the sequence has a dead-end, that is, ends with a state for which $S' = \emptyset$. This is impossible because it would cause the agent to get stuck in this state, preventing it from reaching the current state. Since last-visit information is maintained across episodes, $s_{[N]}$ is a terminal state if the path followed after the previous visit of s_t led to a terminal state. Next, assume the sequence contains the same intermediate state twice. After the second visit of this intermediate state, the subsequent sequence would be the same as after the first visit, since there is only a single last-visit next state defined per state. This would create an infinite sequence of next states, also preventing the agent from reaching the current state.

The set of equations can be solved by backwards substituting the equations, that is, substituting the equation for $V^B(s_{[n+1]})$ in the one for $V^B(s_{[n]})$ and so on until a single equation for $V^B(s_{[0]})$ remains of the form

$$V^B(s_{[0]}) = c_A + c_B V^B(s_{[N]}),$$

with c_A and c_B defined as

$$c_A = \sum_{i=0}^{N-1}\left((1-\alpha^{[i]})V^{mf}(s_{[i]}) + \alpha^{[i]} r_{[i+1]}\right)\prod_{k=0}^{i-1}\gamma\alpha^{[k]}, \tag{8}$$

$$c_B = \prod_{i=0}^{N-1}\gamma\alpha^{[i]}. \tag{9}$$

If $s_{[N]}$ is a terminal state, its value is 0 and $V^B(s_t) = c_A$. On the other hand, if $s_{[N]} = s_t$ then $V^B(s_t) = c_A/(1-c_B)$.

Algorithm 2 shows pseudocode of the on-line policy evaluation algorithm, which computes the best-match value of the current state at each timestep. Lines 7-12 compute the values of c_A and c_B in a forward, incremental way by going from one next state to the other. Note that it is not necessary to store V^{mf} and R' separately, since they are always used in the same combination, $(1-\alpha)V^{mf}(s) + \alpha R'(s)$, which is stored in a single variable, V_r^{mf}, saving space and computation. Line 20 combines the assignments $V^{mf}(s_t) = V(s_t)$, $R'(s_t) = r_{t+1}$ and the computation of V_r^{mf} in a single update. Note that the algorithm makes use of the just-in-time learning principle, that is, updating states at the moment of their revisit. In JIT Q-learning, it is used to improve the

Algorithm 2 Best-Match LVM Evaluation

```
1:  initialize $V(s)$ arbitrarily for all $s$
2:  initialize $S'(s) = \emptyset$ for all $s$
3:  loop {over episodes}
4:     initialize s
5:     repeat {for each step in the episode}
6:        if $S'(s) \neq \emptyset$ then
7:           $c_A \leftarrow V_r^{mf}(s)$; $c_B \leftarrow \gamma \alpha^s$; $s' \leftarrow S'(s)$; $n \leftarrow 0$
8:           while $s' \neq s \wedge s'$ is not terminal do
9:              $c_A \leftarrow c_A + c_B \cdot V_r^{mf}(s')$
10:             $c_B \leftarrow c_B \cdot \gamma \alpha^{s'}$
11:             $s' \leftarrow S'(s')$
12:          end while
13:          if $s' = s$ then
14:             $V(s) \leftarrow c_A/(1 - c_B)$
15:          else
16:             $V(s) \leftarrow c_A$
17:          end if
18:       end if
19:       take action $\pi(s)$, observe $r$ and $s'$
20:       $V_r^{mf}(s) \leftarrow (1 - \alpha^s)V(s) + \alpha^s \cdot r$
21:       $S'(s) \leftarrow s'$; $s \leftarrow s'$
22:    until $s$ is terminal
23: end loop
```

performance without increasing the computation cost, while in the best-match evaluation algorithm it is used to efficiently compute the best-match values.

Algorithm 2 is an on-line algorithm that computes at each timestep the best-match value of the current state. We define the off-line version as one that computes at the end of each episode the best-match values of the states that were visited during that episode. This off-line algorithm is related to off-line TD(λ), as demonstrated by the following theorem. We prove this theorem in Appendix B.

Theorem 7 *For an episodic, acyclic, evaluation task, off-line best-match LVM evaluation computes the same values as off-line TD(λ) with $\lambda_t = \alpha_t(s_t)$.*

For acyclic tasks, that is, episodic tasks with no revisits of states within an episode, $TD(\lambda)$ with $\lambda_t = \alpha_t(s_t)$ can perform TD updates that are unbiased with respect to the initial values (Sutton and Singh, 1994). Because of Theorem 7, this also holds for best-match LVM evaluation. However, in contrast to $TD(\lambda)$, best-match LVM evaluation can perform unbiased updates for any MDP, as we demonstrate with the following theorem, also proven in Appendix B.

Theorem 8 *The state values computed by the on-line best-match LVM evaluation algorithm (Algorithm 2) are unbiased with respect to the initial state values, when the initial learning rates $\alpha_0(s)$ are set to 1 for all s.*

Because best-match LVM evaluation can perform unbiased updates for any MDP, it can often substantially outperform TD(λ) while requiring similar space and computation. We demonstrate this empirically using the two tasks shown in Figure 8. Besides comparing against TD(λ), we also compare against experience replay (Lin, 1992), which stores the n last experience samples and uses them for repeated TD updates.

Task A features a small circular network consisting of four identical states, each having a deterministic transition to a neighbor. The reward received after each transition is +1. Task B is a stochastic variation on the first task, with stochastic transitions and a reward drawn from a normal distribution with mean 1 and standard deviation 0.5. The discount factor is 0.95, resulting in a state value of 20 on both tasks for all states. We compare the RMS error of the current state value $V_t(s_t)$ for all three methods. For experience replay, we performed a TD update for each of the last 4 samples at every timestep, resulting in a computation time similar to best-match LVM and TD(λ). In addition, we implemented a version where all observed samples are stored and updated at each timestep. The learning rate is initialized to 1 and decayed according to

$$\alpha^s = \frac{\alpha_0}{d \cdot [n(s) - 1] + 1}\,.$$

where $n(s)$ is the total number of times state s has been visited. We optimize d as well as λ between 0 and 1. Results are averaged over 5000 runs.

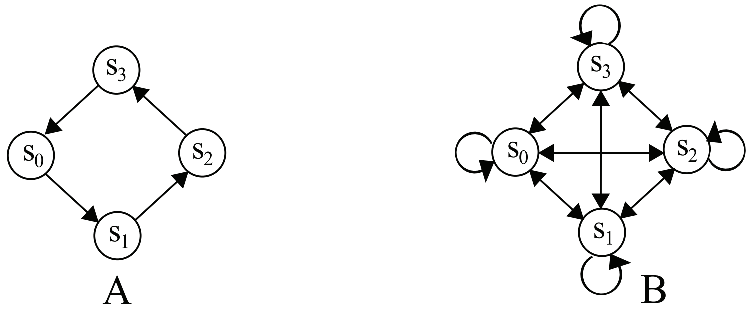


Figure 8: Two tasks for policy evaluation. Task A has deterministic state transitions and a deterministic reward of +1, while task B has stochastic transitions and a reward drawn from a normal distribution with mean +1 and standard deviation 0.5.

Figure 9 shows the experimental results in these tasks. In task A, at timestep 4 the start state is revisited and the RMS error for best-match LVM drops to 0. The reason is that in the deterministic case the last-visit model is equal to the full model once every state has been visited. Furthermore, with learning rates of 1, the best-match LVM equations reduce to the Bellman optimality equations. Therefore best-match LVM effectively performs model-based learning. TD(λ), on the other hand, has to incrementally improve upon the initial values of 0. The spiky behavior of TD(λ) is caused by the combination of a λ of 1, with zero learning rate decay (which were the optimal settings in this case). Experience replay has a performance in between best-match LVM and TD(λ). In task B, the RMS error drops more smoothly. Best-match LVM again substantially outperforms TD(λ) and experience replay, even when all samples are stored and updated. The total computation time for the 5000 runs was marginally higher for experience replay with N=4, which has to maintain a queue of recent samples, than for best-match LVM and TD(λ): on task A, around 90 ms compared

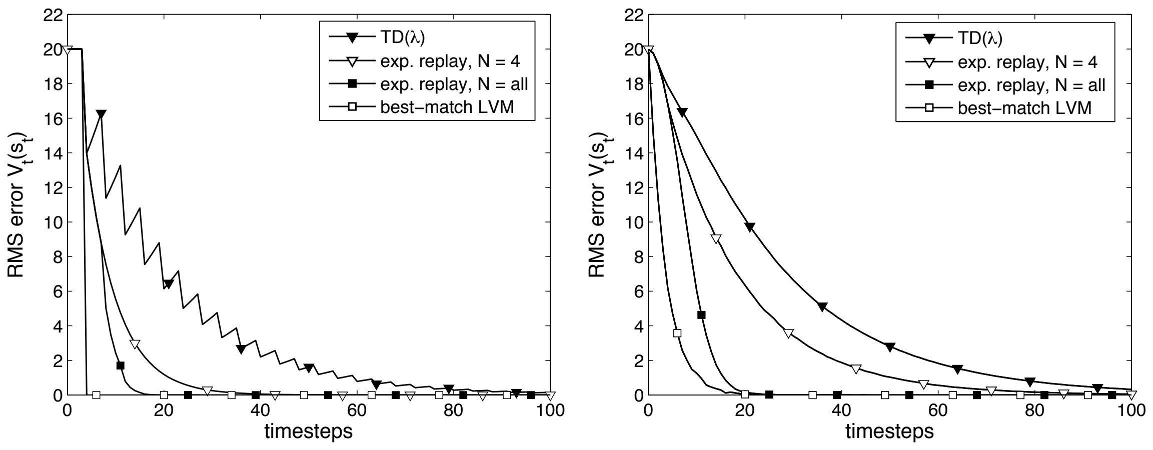


Figure 9: Comparison of the performance of best-match LVM, TD(λ) and experience replay on tasks A (left) and task B (right) of Figure 8.

to 80 ms for both best-match LVM and TD(λ). Experience replay with all samples updated had a computation time of 280 ms. On task B, all methods were about 10 ms slower.

4.3 Best-Match LVM Control

The best-match LVM equations for the control case form a nonlinear set. Therefore, it is in general not possible to compute the exact best-match Q-values at each timestep. However, they can be approximated to arbitrary accuracy via update sweeps through the state-action space, in a manner similar to value iteration, as we prove in the following lemma.

Lemma 9 *For the best-match Q-values the following equation holds for all (s,a):*

$$Q_t^B(s,a) = \lim_{i\to\infty} Q_{t,i}(s,a)\,,$$

where $Q_{t,i}$ is initialized arbitrarily for $i = 0$ and is defined for $i > 0$ as

$$Q_{t,i}(s,a) = \begin{cases} (1-\alpha)Q_t^{mf}(s,a) + \alpha\,[R'_t(s,a) + \gamma\,\max_{a'} Q_{t,i-i}(S'_t(s,a),a')] & \textit{if}\; S'_t(s,a) \neq \emptyset \\ Q_t^{mf}(s,a) & \textit{if}\; S'_t(s,a) = \emptyset\,. \end{cases}$$

Proof For state-action pairs (s,a) with $S'_t(s,a) = \emptyset$ the proof follows directly from the definition of Q_t^B and $Q_{t,i}$. For (s,a) with $S'_t(s,a) \neq \emptyset$, the absolute difference between $Q_{t,i}(s,a)$ and $Q_t^B(s,a)$ can be written as

$$\begin{aligned} |Q_{t,i}(s,a) - Q_t^B(s,a)| &= \alpha\gamma\,|\max_c Q_{t,i-i}(S'_t(s,a),c) - \max_c Q_t^B(S'_t(s,a),c)| \\ &\leq \alpha\gamma\,\max_c |Q_{t,i-i}(S'_t(s,a),c) - Q_t^B(S'_t(s,a),c)| \\ &\leq \alpha\gamma\,||Q_{t,i-i} - Q_t^B||\,. \end{aligned}$$

From this it follows that

$$||Q_{t,i} - Q_t^B|| \leq \alpha\gamma\, ||Q_{t,i-i} - Q_t^B||\,.$$

For $\alpha\gamma < 1$, it follows that for $i \to \infty$, $Q_{t,i} \to Q_t^B$. ■

Lemma 9 shows that Q_t^B can be approximated to arbitrary accuracy with a finite number of best-match updates.

Algorithm 3 shows the pseudocode for a general class of algorithms that approximate the best-match Q-values by performing best-match updates.[4] Lines 9 to 12 perform a series of best-match updates. Note that while only a single Q^{mf} value is updated per timestep, many Q-values can be updated at the same timestep. By varying the way state-action pairs are selected for updating (line 10) and changing the stopping criterion (line 12), a whole range of algorithms can be constructed that trade off computation cost per timestep for better approximations of the best-match Q-values. Note that JIT Q-learning and even regular Q-learning are members of this general class of algorithms. If the state-action pair selection criterion is the state-action pair visited at the previous timestep and the stopping criterion allows only a single update, the algorithm reduces to the regular Q-learning algorithm. Thus, Q-learning is a form of best-match control with a simplistic approximation of the best-match Q-values. However, we reserve the term 'best-match learning' for algorithms that use the same sample multiple times to redo updates.

Algorithm 3 General Best-Match LVM Control

```
1:  initialize $Q(s,a)$ arbitrarily for all $s$,$a$
2:  initialize $S'(s,a) = \emptyset$ for all $s$,$a$
3:  loop {over episodes}
4:      initialize $s$
5:      repeat {for each step in the episode}
6:          select action $a$, based on $Q(s,\cdot)$
7:          take action $a$, observe $r$ and $s'$
8:          $Q^{mf}(s,a) \leftarrow Q(s,a); S'(s,a) \leftarrow s'; R'(s,a) \leftarrow r$
9:          repeat
10:             select some $(s,a)$ pair with $S'(s,a) \neq \emptyset$   {each pair is selected at least once before its
                revisit}
11:             $Q(s,a) \leftarrow (1-\alpha^{sa})Q^{mf}(s,a) + \alpha^{sa}\,[R'(s,a) + \gamma\,\max_c Q(S'(s,a),c)]$
12:         until some stopping criterion has been met
13:         $s \leftarrow s'$
14:     until $s$ is terminal
15: end loop
```

The following theorem states that, for any member of the best-match LVM control class, the Q-values converge to the optimal Q-values.

Theorem 10 *The Q-values of a member of the best-match LVM control class, shown in Algorithm 3, converge to Q^* if the following conditions are satisfied:*

1. *S and A are finite.*

4. Similar to the variable V_r^{mf} of Algorithm 2, a variable Q_r^{mf} can be defined that combines the variables Q^{mf} and R', saving space and computation. For readability we do not show this for Algorithm 3.

2. $\alpha_t(s,a) \in [0,1]$, $\sum_t \alpha_t(s,a) = \infty$, $\sum_t(\alpha_t(s,a))^2 < \infty$ *w.p.1* *and* $\alpha_t(s,a) = 0$ *unless* $(s,a) = (s_t,a_t)$.

3. $Var\{R(s,a,s')\} < \infty$.

4. $\gamma < 1$.

We prove this theorem in Appendix D.

4.4 Best-Match LVM Prioritized Sweeping

A wide range of methods can be constructed within the general class of best-match LVM control algorithms that trade off increased computation time for better approximation of the best-match Q-values in different ways. This section proposes one method that performs this trade-off with a strategy based on *prioritized sweeping* (PS) (Moore and Atkeson, 1993).

PS makes the planning step of model-based RL more efficient by focusing on the updates expected to have the largest effect on the Q-value function. The algorithm maintains a priority queue of state-action pairs in consideration for updating. When a state-action pair (s,a) is updated, all predecessors (i.e., those state-action pairs whose estimated transition probabilities to s are greater than 0) are added to the queue according to a heuristic estimating the impact of the update. At each timestep, the top N state-action pairs from this queue are updated, with N depending on the available computation time. Because PS maintains a full model, it requires $O(|\mathcal{S}|^2|\mathcal{A}|)$ space.

This same idea can be applied to the best-match equations for efficient approximation of the best-match values. A priority queue of state-action pairs is maintained whose corresponding best-match updates have the largest expected effect on the best-match Q-value estimates. When a state-action pair has received an update, all state-action pairs whose last-visit transition state equals the state from the updated state-action pair are placed into the priority queue with a priority equal to the absolute change an update would cause in its Q-value. Since this approach uses only an LVM, it requires only $O(|\mathcal{S}||\mathcal{A}|)$ space.

Algorithm 4 shows the pseudocode of this algorithm, which we call *best-match LVM prioritized sweeping* (BM-LVM). By always putting the state-action pair from the previous timestep on top of the priority queue (line 10), the requirement that each visited state-action pair receives at least one best-match update is fulfilled, guaranteeing convergence in the limit.

On the surface, this algorithm resembles *deterministic prioritized sweeping* (DPS) (Sutton and Barto, 1998), a simpler variation that learns only a deterministic model, uses a slightly different priority heuristic, and performs Q-learning updates to its Q-values. While clearly designed for deterministic tasks, it can also be applied to stochastic tasks, in which case updates are based on an LVM.

However, there is a crucial difference between DPS and BM-LVM. By performing updates with respect to Q^{mf} instead of Q, BM-LVM corrects previous updates instead of performing multiple updates based on the same sample. This ensures proper averaging of experience and enables convergence to the optimal Q-values using only an LVM, even in stochastic environments. This is not guaranteed for DPS since if some samples are used more often than others a bias towards these samples is created, which can prevent convergence to the optimal Q-values.

We compare the performance of PS, DPS, and BM-LVM on the deterministic and stochastic variation of the Dyna Maze task shown in Figure 3. In addition, we also compare to Q(λ) as

Algorithm 4 Best-Match LVM Prioritized Sweeping (BM-LVM)

```
1: initialize $Q(s,a)$ arbitrarily for all $s$,$a$
2: initialize $S'(s,a) = \emptyset$ for all $s$,$a$
3: initialize PQueue as an empty queue
4: loop {over episodes}
5:   initialize s
6:   repeat {for each step in the episode}
7:     select action $a$, based on $Q(s,\cdot)$
8:     Take action $a$, observe $r$ and $s'$
9:     $S'(s,a) \leftarrow s'; R'(s,a) \leftarrow r; Q^{mf}(s,a) \leftarrow Q(s,a)$
10:    promote $(s,a)$ to top of priority queue
11:    $n \leftarrow 0$
12:    while $(n < N) \wedge$ ($PQueue$ is not empty) do
13:      $s_1, a_1 \leftarrow first(PQueue)$
14:      $Q(s_1,a_1) \leftarrow (1-\alpha^{s_1 a_1})\, Q^{mf}(s_1,a_1) + \alpha^{s_1 a_1}\,[R'(s_1,a_1) + \gamma \max_c Q(S'(s_1,a_1),c)]$
15:      $V_{s_1} \leftarrow max_{a'}\, Q(s_1,a')$
16:      for all $(s,a)$ with $S'(s,a) = s_1$ do
17:        $p \leftarrow |(1-\alpha^{sa})\, Q^{mf}(s,a) + \alpha^{sa}\,[R'(s,a) + \gamma V_{s_1}] - Q(s,a)|$
18:        if $p > \theta$ then
19:          insert $(s,a)$ into $PQueue$ with priority $p$
20:        end if
21:      end for
22:      $n \leftarrow n+1$
23:    end while
24:    $s \leftarrow s'$
25:  until $s$ is terminal
26: end loop
```

described by Watkins (1989). This is an off-policy control version of eligibility traces. We also tried Sarsa(λ), the on-policy version, since it can sometimes outperform Q(λ) considerably, but saw no significant difference for these experiments and present only the Q(λ) results. Note that when a greedy behavior policy is used, as in the deterministic experiment, Q(λ) computes exactly the same values as Sarsa(λ). As in Section 4.2, we also compare to experience replay.

Finally, we compare to delayed Q-learning (Strehl et al., 2006), a model-free method that, like some model-based methods (Brafman and Tennenholtz, 2002; Kearns and Singh, 2002; Strehl and Littman, 2005), is proven to be *probably approximately correct* (PAC), that is, its sample complexity is polynomial with high probability. Delayed Q-learning initializes its Q-values optimistically and ensures that value estimates are not reduced until the corresponding state-action pairs have been sufficiently explored. Because it does not maintain a model, it has the same $O(|\mathcal{S}||\mathcal{A}|)$ space requirements as best-match prioritized sweeping. However, to our knowledge, its empirical performance has never been evaluated before.

For each method, the free parameters are optimized within a certain range. In the deterministic case, for Q(λ) we optimized the λ value in the range from 0 to 1, and the learning rate decay d (using Equation 4) in the range from 0 to 1, while α_0 was set to 1. We also optimized the (unbounded) trace type (replacing versus accumulating). For delayed Q-learning we optimized m in the range from 1 to 5 with steps of 1 and e_1 in the range 0 to 0.020 with steps of 0.001. For DPS and BM-LVM, we did not optimize any parameters in the deterministic case, but simply used a constant α of 1. In the stochastic case, we also optimized the learning rate decay d for DPS and BM-LVM.

For all methods, we used optimistic initialization with $Q_0 = 20$ in order to get a fair comparison with delayed Q-learning, for which initialization to $R_{max}/(1-\gamma)$ is part of the algorithm.[5]

In the deterministic case we used a greedy behavior policy, while we used an ε-greedy policy with $\varepsilon = 0.1$ in the stochastic variant. For all prioritized-sweeping algorithms we performed a maximum of 20 updates per timestep (i.e., N = 20). For experience replay we used the last 20 samples, which also results in 20 updates per timestep. Results are averaged over 1000 independent runs.

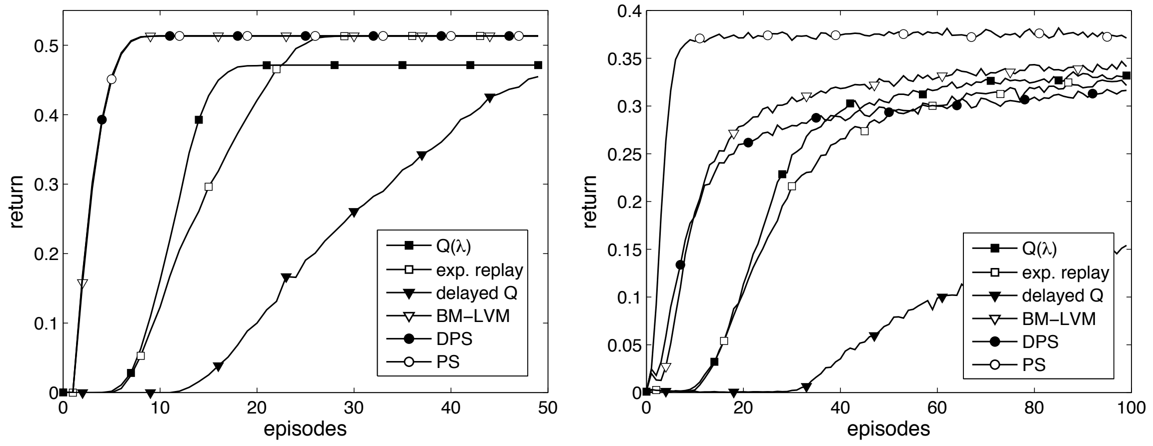


Figure 10: Comparison of the performance of BM-LVM and several competitors on the deterministic (left) and stochastic (right) Dyna Maze task.

Figure 10 shows the return as a function of the number of episodes, while Tables 2 and 3 show the average return over the measured episodes and the optimal parameter values. In the deterministic experiment, we see that the performance of PS, DPS, and BM-LVM is exactly equal, as expected when $\alpha = 1$, since the last-visit experience is equal to the model of the environment. Q(λ) performs considerably worse than the prioritized sweeping methods and does not converge to the optimal policy. In contrast, the combination of a greedy behavior policy with optimistic initialization enables the prioritized sweeping methods to converge to the optimal policy in a deterministic environment. Experience replay performs similarly to Q(λ), though it does converge to the optimal policy. Delayed Q-learning also converges to the optimal policy, as predicted by the theory, but does so much more slowly.

In the stochastic experiment, PS has a clear performance advantage. However, the goal of BM-LVM is not to match or even come close to the performance of PS. It cannot match this performance in general, since PS takes advantage of its higher space complexity. Instead, the goal of BM-LVM

5. For this task $r = R_{max}$ only when the exit is reached and 0 otherwise. Thus, the Q-values can never be higher than 1 and $Q_0 = 20$ is overly optimistic. However, since realizing that an initialization of 1 is possible would require extra prior knowledge, we initialize to 20.

	deterministic - 50 eps.			
	optimal parameters	average return	standard error	time per step $(\cdot 10^{-6} s)$
Q(λ)	λ: 0.8, d: 0	0.3606	0.0007	0.68
exp. replay	d: 0	0.3602	0.0004	0.37
delayed Q	m: 1, $e_1 = 0$	0.1878	0.0004	0.11
BM-LVM	d: 0	0.4769	0.0002	0.88
DPS	d: 0	0.4774	0.0002	0.85
PS	-	0.4772	0.0002	0.95

Table 2: Average return and optimal parameters (d = α decay rate) of best-match prioritized sweeping and several competitors on the deterministic Dyna Maze task.

	stochastic - 100 eps.			
	optimal parameters	average return	standard error	time per step $(\cdot 10^{-6} s)$
Q(λ)	λ: 0.9, d: 0.03	0.2417	0.0007	0.59
exp. replay	d: 0.18	0.2272	0.0006	0.43
delayed Q	m: 2, e_1:0.015	0.0668	0.0004	0.12
BM-LVM	d: 0.02	0.2911	0.0006	3.2
DPS	d: 0.30	0.2683	0.0008	3.7
PS	-	0.3603	0.0004	4.7

Table 3: Average return and optimal parameters (d = α decay rate) of best-match prioritized sweeping and several competitors on the stochastic Dyna Maze task.

is to optimally perform at a space complexity of $O(|\mathcal{S}||\mathcal{A}|)$. The results confirm that BM-LVM is considerably better than the other methods with this space complexity, like Q(λ) and DPS. DPS initially performs well, but cannot keep up with BM-LVM after about 10 episodes, even though BM-LVM has similar space and computation costs per timestep. Experience replay performs slightly worse than Q(λ). We tested whether doubling the size of the stored experience sequence improves the performance of experience replay, but this led to no significant performance increase. Delayed Q-learning also performs poorly in the stochastic case, despite its PAC bounds.

The computation time of BM-LVM, DPS and PS is in the deterministic experiment considerably lower than in the stochastic case. The reason for this is that while in both cases the maximum number of updates per timestep is 20, in the deterministic case the priority queue often has fewer than 20 samples, so fewer updates occur. The computation time of Q(λ) is slightly better than that of BM-LVM, while experience replay is about twice as fast as BM-LVM.

In the stochastic experiment, the computation time of Q(λ) is much better than that of any of the prioritized sweeping algorithms, which could suggest that Q(λ) is a better choice than BM-LVM when computation power is scarce. To test this hypothesis, we performed additional experiments with smaller values of N. The computation time for BM-LVM for $N = 4$ ($0.61 \cdot 10^{-6}$ s) was similar to that of Q(λ). The average return of BM-LVM dropped to 0.2598 in this case, which is still considerably better than the average return of Q(λ). This demonstrates that BM-LVM is a better choice than Q(λ) even under severe computational constraints.

Together, these results clearly demonstrate the strength of best-match learning, since BM-LVM outperforms several competitors with similar space complexity. However, the results also show that the performance gap with full model-based learning can be considerable. Therefore, if more space is available, a better approximate model would be preferred. We address this need in the next section by applying best-match learning to an *n*-transition model, which estimates the transition function for *n* next states per state-action pair, allowing increased space requirements to be traded for improved performance.

5. Best-Match *n*-Transition Model

The best-match LVM equations described above combine model-free Q-values with the last-visit model. When state-action pairs have only a small number of possible next states, the last-visit model can effectively approximate the full model. In other cases, however, the last-visit model captures only a fraction of the full model and the effect of the best-match updates will be small. In this section, we combine best-match learning with the *n*-transition model, which estimates the transition probability for *n* possible next states of each state-action pair. By tuning *n*, increased space requirements can be traded for improved performance.

5.1 Generalized Best-Match Equations

Best-match LVM learning takes the idea of using more accurate update targets to the extreme by continuously revising update targets with best-match updates. For a specific sample, the update target is revised until the moment of revisit of the corresponding state-action pair, since at that moment the sample is overwritten with the newly collected sample. However, if space allows, the new sample can be stored along with the old sample instead of overwriting it, allowing the update target from the new as well as the old sample to be further improved. We explain with an example how this changes the best-match equations.

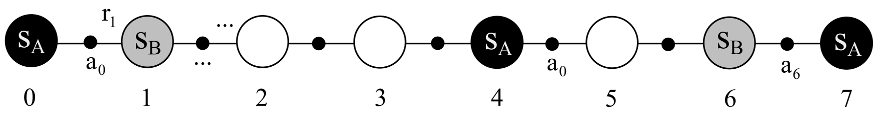


Figure 11: A state transition sequence in which best-match updates can enable further postponing. Timesteps are shown below each state.

Consider the state-action sequence from Figure 11 and assume the best-match Q-values are computed at each timestep. At the revisit of s_A, action a_0 is retaken. Therefore, when using the LVM, at timestep 5 the old experience sample is overwritten with the new experience. Before this occurs, the old experience is used in a final update of Q^{mf}. Let υ_y^x indicate the update target from the sample collected at timestep x based on the best-match Q-value of timestep y: $\upsilon_y^x = r_x + \gamma \max_a Q_y^B(s_x, a)$. Using this convention the update of Q^{mf} at timestep 5 becomes

$$Q_5^{mf}(s_A, a_0) = (1-\alpha) Q_0^{mf}(s_A, a_0) + \alpha \upsilon_4^1 .$$

At timestep 7, the best-match LVM equation for (s_A, a_0) can be written as

$$\begin{aligned} Q_7^B(s_A,a_0) &= (1-\alpha)Q_7^{mf}(s_A,a_0)+\alpha\upsilon_7^5 \\ &= (1-\alpha)Q_5^{mf}(s_A,a_0)+\alpha\upsilon_7^5 \\ &= (1-\alpha)^2 Q_0^{mf}(s_A,a_0)+\alpha(1-\alpha)\upsilon_4^1+\alpha\upsilon_7^5 . \end{aligned}$$

Thus, the best-match Q-value of (s_A, a_0) at timestep 7 is equal to a weighted average of Q_0^{mf}, υ_4^1 and υ_7^5. On the other hand, if both the old and the new sample are stored, Q-values from timestep 7 could also be used for the update target of the old sample, yielding

$$Q_7^B(s_A,a_0) = (1-\alpha)^2 Q_0^{mf}(s_A,a_0)+\alpha(1-\alpha)\upsilon_7^1+\alpha\upsilon_7^5 . \tag{10}$$

For the state-sequence from Figure 11 this means that the experience resulting from (s_B, a_6) is also taken into account in the update target for (s_A, a_0).

The above example shows how the best-match LVM equations can be naturally extended to two samples per state-action pair. Following the same pattern, we can define best-match equations given an arbitrary set of samples. Consider the set of samples X of size N_X, where a sample $x \in X$ has the form $\{s, a, r, s'\}$. These samples can be grouped according to their state-action pairs. We define X_{sa} as the subset of X containing all samples belonging to state-action pair (s, a) and N_{sa}^x as the size of X_{sa}. Without loss of generality, we index the samples from X_{sa} as x_k^{sa} for $1 \le k \le N_{sa}^x$. In addition, we define W_{sa} as a set consisting of $N_{sa}^x + 1$ weights $w_k^{sa} \in \mathbb{R}$ such that $0 \le w_k^{sa} \le 1$ for $0 \le k \le N_{sa}^x$ and $\sum_{k=0}^{N_{sa}^x} w_k^{sa} = 1$. We define W as the union of the weight sets from all state-action pairs.

Definition 11 *The generalized best-match equations with respect to Q_t^{mf}, X and W are*

$$Q_t^B(s,a) = w_0^{sa} Q_t^{mf}(s,a) + w_1^{sa}\upsilon_1^{sa} + w_2^{sa}\upsilon_2^{sa} + ... + w_{N_{sa}^x}^{sa}\upsilon_{N_{sa}^x}, \qquad \textit{for all } s,a, \tag{11}$$

where $\upsilon_k^{sa} = r + \gamma \max_c Q_t^B(s', c)\ |r, s' \in x_k^{sa}$.

Note that Equation 11 reduces to $Q_t^B(s,a) = Q_t^{mf}(s,a)$ for state-action pairs with no samples in X.

Within this context, Q^{mf} is defined as a model-free Q-value constructed from all observed samples except those in X. Consequently, when a sample is removed from X, it is used for a model-free update of Q^{mf}.

Using Definition 11, a range of algorithms can be constructed based on different sets of samples X and weights W. When the samples are combined by incremental Q-learning updates, like in Equation 10, the weights have the values

$$w_0^{sa} = \prod_{i=1}^{N_{sa}^x}(1-\alpha_i^{sa}), \tag{12}$$

$$w_k^{sa} = \alpha_k^{sa}\prod_{i=k+1}^{N_{sa}^x}(1-\alpha_i^{sa}), \qquad \text{for } 1 \le k \le N_{sa}^x . \tag{13}$$

With this weight distribution, the update targets from older samples have lower weights than more recent samples. In Q-learning, more recent samples in general have more accurate update targets so giving them higher weight makes sense. However, in best-match learning the update targets from all stored samples have the same time index so there is no reason to use different weights for them. A better weight distribution gives all samples the same weights:

$$w_k^{sa} = (1 - w_0^{sa})/N_{sa}^x, \qquad \text{for } 1 \leq k \leq N_{sa}^x,$$

for some value of w_0^{sa}.

The last-visit model, storing one sample for each state-action pair, is one possible sample set. A straightforward extension is to store n samples per state-action pair. In the following section, however, we propose a different sample set, called the n-transition model, which can be stored more compactly.

5.2 Best-Match Learning based on the n-transition Model

While BM-LVM outperforms model-free methods with the same space complexity, it does not perform as well as PS, which stores a full model. This is symptomatic of an important limitation of BM-LVM: it offers only a single trade-off between space and performance. When there is not enough space available to store the full model, but more than enough to store the LVM, a more sophisticated method is needed to make maximal use of the available space. Using the generalized best-match equations, we can construct such a method.

An obvious approach is to store n samples per state-action pair. However, obtaining an accurate model often requires a large n, even when the number of next states per state-action pair is small. A more space-efficient solution is to group together samples that have the same next state. If we store the size of such a group in $N_{sas'}^x$ and give each sample a weight of $1/N_{sa}$, where N_{sa} is the total number of times state-action pair (s,a) is visited, then we can rewrite the contribution from all samples of X_{sa} to the best-match equations as

$$\sum_{k=1}^{N_{sa}^x} w_k \upsilon_k = \frac{1}{N_{sa}} \left[\sum_X r_{sa} + \gamma \sum_{s'} N_{sas'}^x \max_{a'} Q^B(s',a') \right],$$

where $\sum_X r_{sa}$ is the sum of the rewards from all samples in the sample set belonging to (s,a). Using $w_0^{sa} = 1 - N_{sa}^x/N_{sa}$, $\hat{\mathcal{P}}_{sa}^{s'} = N_{sas'}^x/N_{sa}^x$ and $\hat{\mathcal{R}}_{sa} = \sum_X r_{sa}/N_{sa}^x$, the generalized best-match equations can now be rewritten as

$$Q^B(s,a) = w_0^{sa} Q^{mf}(s,a) + (1 - w_0^{sa}) \left[\hat{\mathcal{R}}_{sa} + \gamma \sum_{s'} \hat{\mathcal{P}}_{sa}^{s'} \max_{a'} Q^B(s',a') \right], \qquad \text{for all } s,a.$$

In these equations, $\hat{\mathcal{P}}$ and $\hat{\mathcal{R}}$ constitute a sparse, approximate model, whose size can be controlled by limiting the number of next states per state-action pair for which $\hat{\mathcal{P}}$ is estimated. w_0^{sa} is the fraction of all samples belonging to (s,a) not used by the sparse model. We define an *n-transition model* (NTM) to be one that estimates the transition probability $\hat{\mathcal{P}}$ for n next states per state action pair. Once a sample enters the model, that is, is used to update $\hat{\mathcal{P}}$, it stays in the model. Each sample not used to update the model is used for a model-free update of Q^{mf}. Different strategies can be used to determine which samples enter the model. A simple approach is to use the first n unique next states that are encountered for a specific state-action pair.

Algorithm 5 shows general pseudocode for best-match NTM learning. The algorithm presents two trade-offs. First, the space complexity can be traded off with performance by selecting n.

Algorithm 5 General Best-Match NTM Control

```
1:  initialize $Q(s,a) = Q^{mf}(s,a)$ arbitrarily for all $s$,$a$
2:  initialize $N_{sa}, N^x_{sa}, R^{sum}_{sa}$ to 0 for all $s$,$a$
3:  initialize $N^x_{sas'}$ to 0 for all $s$,$a$ and $s' \in NTM(s,a)$
4:  initialize $w_0^{sa}$ to 1 for all $s$,$a$
5:  loop {over episodes}
6:    initialize $s$
7:    repeat {for each step in the episode}
8:      select action $a$, based on $Q(s,\cdot)$
9:      take action $a$, observe $r$ and $s'$
10:     if $s' \in NTM(s,a)$ then
11:       $N^x_{sa} = N^x_{sa} + 1$;   $N^x_{sas'} = N^x_{sas'} + 1$;   $R^{sum}_{sa} = R^{sum}_{sa} + r$
12:       $\hat{\mathcal{P}}^{s'}_{sa} = N^x_{sas'} / N^x_{sa}$;   $\hat{\mathcal{R}}_{sa} = R^{sum}_{sa} / N^x_{sa}$
13:     else
14:       $Q^{mf}(s,a) \leftarrow (1-\alpha^{sa}) Q^{mf}(s,a) + \alpha^{sa} [r + \gamma \max_c Q(s',c)]$
15:     end if
16:     $N_{sa} = N_{sa} + 1$
17:     $w_0^{sa} = 1 - N^x_{sa} / N_{sa}$
18:     repeat
19:       select some $(s,a)$ pair with $N_{sa} > 0$   {each pair is selected at least once before its
          revisit}
20:       $Q(s,a) \leftarrow w_0^{sa} Q^{mf}(s,a) + (1-w_0^{sa}) \left[\hat{\mathcal{R}}_{sa} + \gamma \sum_{s'} \hat{\mathcal{P}}^{s'}_{sa} \max_c Q(s',c)\right]$
21:     until some stopping criterion has been met
22:     $s \leftarrow s'$
23:   until $s$ is terminal
24: end loop
```

Second, the computation time per simulation step can be traded off with performance by controlling the number of best-match updates performed per timestep.

Based on this general control algorithm, various specific algorithms can be constructed using different stopping criteria and strategies for selecting state-action pairs to receive best-match updates. The following theorem states that, for any member of this class, the Q-values converge to the optimal Q-values. We prove this theorem in Appendix E.

Theorem 12 *The Q-values of a member of the best-match NTM control class, shown in Algorithm 5, converge to Q^* if the following conditions are satisfied:*

1. *S and A are finite.*
2. $\alpha_t(s,a) \in [0,1]$, $\sum_t \alpha_t(s,a) = \infty$, $\sum_t (\alpha_t(s,a))^2 < \infty$ *w.p.1* *and* $\alpha_t(s,a) = 0$ *unless* $(s,a) = (s_t,a_t)$ *and* $s_{t+1} \notin NTM(s_t,a_t)$.
3. $Var\{R(s,a,s')\} < \infty$.
4. $\gamma < 1$.

5.3 Experimental Results

As in BM-LVM, prioritized sweeping can be used to trade off computation time and performance in Algorithm 5, yielding a method we call BM-NTM. We compare its performance to BM-LVM, Q-learning, and a sparse model-based method that combines prioritized sweeping with an NTM without best-match updates, which we call PS-NTM. While BM-NTM uses the samples that are not part of the NTM to update Q^{mf}, PS-NTM ignores these samples. The priority of a state-action pair (s,a) for BM-NTM is defined as

$$p = (1 - w_0^{sa})\hat{\mathcal{P}}_{sa}^{s_1} \cdot |\Delta V(s_1)|\,,$$

where $\Delta V(s_1)$ is the difference in the state value of s_1 before and after the best-match update of one of the Q-values of s_1. For PS-NTM, the priority is defined similarly:

$$p = \hat{\mathcal{P}}_{sa}^{s_1} \cdot |\Delta V(s_1)|\,.$$

The NTM we use for BM-NTM and PS-NTM is defined by the first n unique next states that are encountered for a specific state-action pair. Although more sophisticated models could be used (e.g., by estimating the n most likely transition states), this model is sufficient for our experimental setting since most transition states have similar transition probabilities.

We consider the large maze task shown at the left in Figure 12. For this maze, the reward received by the agent is -0.1 at each timestep, while reaching the goal state results in a reward of +100. The discount factor is 0.99. The agent can take four actions, ‘north’,‘south’,‘east’ and ‘west’. The action outcomes are made very stochastic, in order to compare different model sizes. The right side of Figure 12 shows the relative action outcome for a ‘north’ action. In free space, there are 15 possible next states, each with equal transition probability. On the other hand, walls prevent not only the transition to the square the wall is located on, but also any squares behind the wall. Therefore, close to a wall the number of possible next states is less than 15. When transition to a square is blocked by a wall, the transition probability of that square is added to the transition probability of the square in front of the wall. In order to make reaching the goal feasible despite the stochastic actions, we use a goal area consisting of four goal states.

To compare performance, we measure the average return for each method over the first 500 episodes. For all methods, we use an ε-greedy policy with $\varepsilon = 0.05$ and initialize Q-values to 0. BM-NTM, PS-NTM and BM-LVM perform a maximum of 5 updates per timestep. For all learning rate based methods, we use an initial learning rate of 1 and decay the learning rate according to Equation 4, while optimizing the decay rate d. Results are averaged over 200 independent runs. An episode is stopped prematurely if the goal is not reached within 500 steps.

Table 4 presents the results, including the average return, optimal parameters, and computation time per simulation step. The model sizes used are N = 1, 3, 5, and 15. For $N = 15$, all samples enter the model. Therefore, BM-NTM has no decay rate in this case. The model weight indicates the fraction of samples that entered the model. BM-NTM has in general a slightly higher weight than PS-NTM, indicating the agent spends less time in open spaces and more time close to a wall.

For model sizes $N = 1$ and $N = 3$, the average return of BM-NTM is much better than that of PS-NTM, despite the fact that for $N = 3$ more than a third of the samples are stored in the model. For $N = 1$, the average return of PS-NTM is even worse than that of Q-learning. Figure 13 shows the return as a function of the number of episodes for BM-NTM and PS-NTM with $N = 1$ and $N = 3$. Unlike BM-NTM, the asymptotic performance for PS-NTM is clearly bounded by the size of the

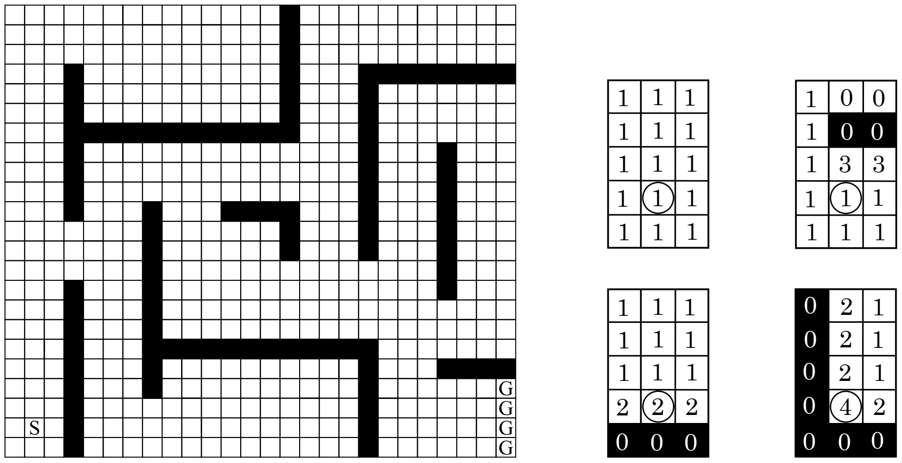


Figure 12: Left, the large maze task, in which the agent must travel from *S* to one of the *G*'s. Right, transition probabilities $(\cdot\frac{1}{15})$ of a 'north' action for different positions of the agent (indicated by the circle) with respect to the walls (black squares). When the transition to a square is blocked by a wall, its transition probability is added to that of the square in front of the wall.

model. Thus, PS-NTM can match the performance of BM-NTM only when the space reduction over the full model is quite small (i.e., less than a factor of 2).

Interestingly, when $N = 1$, BM-LVM outperforms BM-NTM despite having the same space complexity. Thus, when space is scarce, BM-LVM is a good option. In contrast, BM-NTM can exploit larger models to further improve performance. The computation time per simulation step for BM-NTM is comparable to that of PS-NTM, with the exception of $N = 1$, for which it is four times larger. The reason is that the priority queue of PS-NTM is often close to empty in this case and thus the 5 updates per timestep are often not reached.

Overall, these results clearly demonstrate the strength of best-match NTM learning. When a significant space reduction over storing the full model is required, BM-NTM performs dramatically better than PS-NTM at similar computational cost.

6. Best-Match Function Approximation

The BM-NTM method described in the previous section has a space complexity of $O(n|\mathcal{S}||\mathcal{A}|)$ compared to $O(|\mathcal{S}|^2|\mathcal{A}|)$ for full model-based methods. However, in problems with large state spaces, this space complexity may be prohibitive even when $n = 1$. In addition, BM-NTM cannot be applied in problems with continuous state spaces. To address these limitations, this section demonstrates that the principles behind best-match learning can also be applied to function approximation. We show that the resulting algorithm, which combines the N most recent samples with the model-free Q-value function, outperforms both linear Sarsa(λ) and linear experience replay on the mountain car task. We start by describing best-match learning based on the N most recent samples for the tabular case, and then we show how this can be extended to the function approximation case.

	model size	model weight	optimal parameters	average return	standard error	time per step ($\cdot 10^{-6}$ s)
PS-NTM	1	0.12	-	-16.9	0.4	0.21
	3	0.36	-	9.8	0.3	1.5
	5	0.57	-	22.6	0.2	2.1
	15	1.00	-	28.9	0.2	3.1
BM-NTM	1	0.14	d = 0.04	15.4	0.3	0.85
	3	0.40	d = 0.09	19.6	0.2	1.7
	5	0.60	d = 0.06	22.3	0.2	2.2
	15	1.00	-	29.3	0.2	3.1
BM-LVM	-	-	d = 0.09	17.4	0.3	1.5
Q-learning	-	-	d = 0.03	2.4	0.2	0.09

Table 4: Average return over the first 500 episodes, optimal parameters (d: α decay rate) and computation time per simulation step on the Large Maze task.

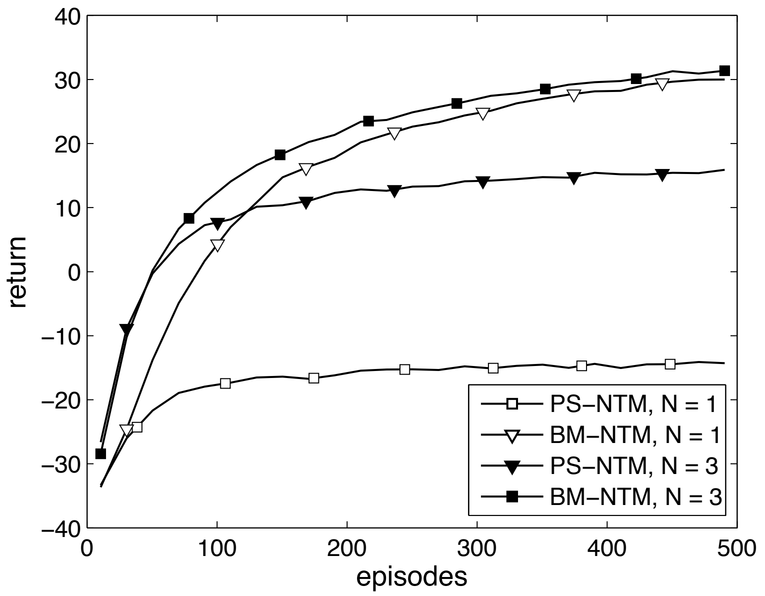


Figure 13: Performance of BM-NTM and PS-NTM on the large maze task.

6.1 Tabular Sequence Based Best-Match Learning

The generalized best-match equations are defined for an arbitrary set of samples (see Definition 11), which can be stored in a model or as an explicit set. To combine best-match principles with function approximation, we employ an explicit set consisting of the last N observed samples, an approach we call *sequence based best-match learning*. In this section we describe sequence based best-match learning for the tabular case and its advantage over experience replay, which also exploits a set of recent samples. In the next section, we extend the tabular version of sequence based best-match learning to function approximation.

Assume that a queue of the last N samples is maintained. When the queue is full and a new sample is added to the back of the queue, the sample at the front of the queue is removed and used to perform a model-free update of $Q^{mf}(s,a)$. The queue may contain multiple samples that belong to the same state-action pair. If there are N^x_{sa} samples belonging to state-action pair (s,a), then the best-match update based on these samples is

$$Q_{t,i+1}(s,a) = w^{sa}_0\, Q^{mf}_t(s,a) + w^{sa}_1 \upsilon^{sa}_1 + w^{sa}_2 \upsilon^{sa}_2 + ... + w^{sa}_{N^x_{sa}} \upsilon_{N^x_{sa}}\,, \qquad (14)$$

where $\upsilon^{sa}_k = r + \gamma \max_c Q_{t,i}(s',c)\, |r,s' \in x^{sa}_k$. When the weights are defined according to Equations 12 and 13, this update can be implemented incrementally by performing N^x_{sa} Q-learning updates:

$$Q_{<k>}(s,a) = (1-\alpha)Q_{<k-1>}(s,a) + \alpha\,[r_k + \gamma \max_{a'} Q_{t,i}(s'_k,a')]\,, \qquad \text{for } 1 \le k \le N^x_{sa}\,,$$

with $Q_{<0>}(s,a) = Q^{mf}_t(s,a)$ and $Q_{t,i+1}(s,a) = Q_{<N^x_{sa}>}(s,a)$.

By stepping through the queue from front to back and using each sample to perform an incremental Q-learning update, all state-action pairs with samples in the queue receive one full best-match update, according to Equation 14. By storing the intermediate $Q_{<k>}$ values at the same location as the final Q-value, $Q_{<N^x_{sa}>}$ automatically becomes $Q_{t,i+1}$ after all incremental updates have been performed. This implementation requires that the Q-values from the state-action pairs with samples in the queue are set equal to $Q_{<0>}$, that is, to Q^{mf}_t, before the update sweep begins. Before resetting these Q-values, the update targets of the samples must be recomputed.

Despite a superficial resemblance, sequence based best-match learning is fundamentally different from experience replay. Best-match learning uses the stored samples to correct previous updates based on those samples, whereas experience replay performs additional updates with the same sample. To illustrate the effect of this difference, suppose that sample (s,a,r,s') is observed at timestep $t = 1$ and used for an update n timesteps in a row. For simplicity, assume there are no other samples belonging to (s,a) in the sample queue and that the learning rate α is constant. We indicate the update target of the sample with υ_i, where i corresponds to the timestep at which the update is performed. Therefore, υ_{i+1} is likely to be more accurate than υ_i since it uses more recent Q-values for s'. Since experience replay performs additional updates we can express $Q_n(s,a)$, the Q-value of (s,a) at timestep n, in terms of $Q_0(s,a)$ and the update targets from the different timesteps as follows:

$$Q_n(s,a) = w_0\, Q_0(s,a) + w_1\upsilon_1 + w_2\upsilon_2 + ... + w_n\upsilon_n\,,$$

with $w_0 = \prod_{i=1}^n (1-\alpha)$ and $w_k = \alpha \prod_{i=k+1}^n (1-\alpha)$ for $k > 0$. If $\alpha \ll 1$, the weights can be accurately described with first-order approximations in α, yielding $w_0 \approx 1 - n\alpha$ and $w_k \approx \alpha$ for $k > 0$. Using these approximations, we can write for $Q_n(s,a)$:

$$Q_n(s,a) \approx (1-\beta)Q_0(s,a) + \beta\, \frac{\sum_{i=1}^n \upsilon_i}{n}\,, \qquad (15)$$

with $\beta = n\alpha$. On the other hand, best-match learning uses the sample for best-match updates, that is, $Q_n(s,a) = (1-\alpha)Q_n^{mf}(s,a) + \alpha\upsilon_n$. However, since $Q_i^{mf}(s,a)$ gets updated only when a sample is removed from the queue, $Q_n^{mf}(s,a) = Q_0(s,a)$ in this case. Therefore, the following holds for best-match learning:

$$Q_n(s,a) = (1-\alpha)Q_0(s,a) + \alpha\upsilon_n\,. \tag{16}$$

The difference between Equation 15 and Equation 16 illustrates the fundamental advantage of sequence based best-match learning, for which Q_n can be seen as an update with sample (s,a,r,s') using the most recent update target. In contrast, experience replay effectively performs an update using an update target that is an average of the update targets from the different timesteps. Therefore, the older, less accurate update targets still have an effect on Q_n.

6.2 Best-Match Gradient Descent Learning

Since tabular sequence based best-match learning can be implemented by incremental Q-learning updates, it can be easily extended to function approximation by combining it with the general gradient descent update for Q-values (Sutton and Barto, 1998)

$$\theta_{t+1} = \theta_t + \alpha\left[\upsilon_t - Q_t(s_t,a_t)\right]\nabla_{\theta_t} Q_t(s_t,a_t)\,, \tag{17}$$

where θ_t is a weight vector corresponding to the basis functions of the approximation.

Algorithm 6 shows pseudocode for general gradient descent best-match function approximation. Note that a learning rate and the most recent update target are stored per sample. The updates of θ and θ^{mf} are based on Equation 17.

We evaluate a linear version of the best-match gradient descent algorithm by comparing its performance with linear Sarsa(λ) as well as a linear version of experience replay on the mountain car task (Boyan and Moore, 1995; Sutton, 1996; Sutton and Barto, 1998) using the settings as described in Sutton and Barto (1998). This involves tile coding with ten 9x9 tilings, a discount factor of 1, an exploration parameter $\varepsilon = 0$, and Q-values optimistically initialized to 0. Additionally, to bound the run-time of an experiment, an episode is stopped prematurely if the goal is not reached within 1000 steps. Linear Sarsa(λ) is known for its good performance on this task (Sutton and Barto, 1998) and is therefore a good benchmark test. For Sarsa(λ), we use the settings that showed the best performance over the first 20 episodes: $\alpha = 0.14$ and $\lambda = 0.9$ with replacing traces. We tested whether decaying the learning rate improves the performance for a number of different α values around 0.14 but did not find a significant improvement. To make Sarsa(λ) more computationally efficient, traces are cut-off for state-action pairs that were visited longer than 20 timesteps ago. For best-match and experience replay, a queue of the 20 most recent samples is used and a single update sweep through this sample set is performed at every timestep. We optimize the initial learning rate α_0 and the learning rate decay d (see Equation 4). Results are averaged over 5000 independent runs.

Table 5 shows the average return over the first 20 episodes, the optimal parameters, and the computation time per simulation step for the 5000 runs. Figure 14 shows the return as a function of the number of episodes. For trace length/N = 20, the performance of linear best-match is about 27% better than that of linear Sarsa(λ).[6] On the other hand, Sarsa(λ) is about twice as fast.

Surprisingly, while experience replay performed comparably to Sarsa(λ) in the tabular case, in the mountain car task it performs 16% better than linear Sarsa(λ). However, as expected, it performs

6. The linear Sarsa(λ) performance is in accordance with the performance found by several other researchers (`http://webdocs.cs.ualberta.ca/~sutton/book/errata.html`).

Algorithm 6 General Gradient-Descent Best-Match

```
1: set $N, \gamma$
2: initialize $\theta, \alpha$ and set $\theta^{mf} = \theta$
3: initialize $SampleQueue$ to empty
4: loop {over episodes}
5:   initialize s
6:   while $s \neq$ terminal state do
7:     select action $a$, based on $\theta$
8:     take action $a$, observe $s'$,$r$
9:     if size $SampleQueue = N$ then
10:      pop sample $x$ from front of the $SampleQueue$
11:      update $\theta^{mf}$ using $x$
12:    end if
13:    decay $\alpha$;  $\upsilon = \emptyset$
14:    push new sample $\{s, a, r, s', \alpha, \upsilon\}$ to back of $SampleQueue$
15:    for all samples $x$ update $\upsilon_x \leftarrow r_x + \gamma \cdot V_{s'_x}$ using $\theta$
16:    for all samples $x$ do
17:      for all features from $x$: $\theta \leftarrow \theta^{mf}$
18:    end for
19:    for all samples $x$ (from front to back of $SampleQueue$) do
20:      update $\theta$ using $\upsilon_x$
21:    end for
22:    $s \leftarrow s'$
23:  end while
24: end loop
```

	optimal parameters	average return	standard error	time per step ($\cdot 10^{-6} s$)
best-match, N=20	$\alpha_0 = 0.10$, d = 0.09	-170.1	0.4	3.0
exp. replay, N=20	$\alpha_0 = 0.10$, d = 0.16	-195.1	0.4	2.5
Sarsa(λ), trace=20	$\lambda = 0.9$, $\alpha_0 = 0.14$, d = 0.0	-231.9	0.4	1.5
best-match, N=15	$\alpha_0 = 0.10$, d = 0.03	-176.3	0.4	2.5
best-match, N= 5	$\alpha_0 = 0.10$, d = 0.03	-215.1	0.4	1.5
Sarsa(λ), trace=∞	$\lambda = 0.9$, $\alpha_0 = 0.14$, d = 0.0	-228.2	0.4	6.7

Table 5: Average performance over the first 20 episodes and the computation time per simulation step on the Mountain Car task ('trace' indicates trace length)

worse than linear best-match. Thus, a substantial portion of the performance improvement linear best-match offers over Sarsa(λ) is due to the use of best-match principles, not simply the reuse of data.

Besides a comparison with equal number of samples/updates, it is interesting to make a comparison with equal computation time. To achieve this, we can either increase the sample set size used by experience replay and Sarsa(λ), or decrease the sample set size used by linear best-match, in such

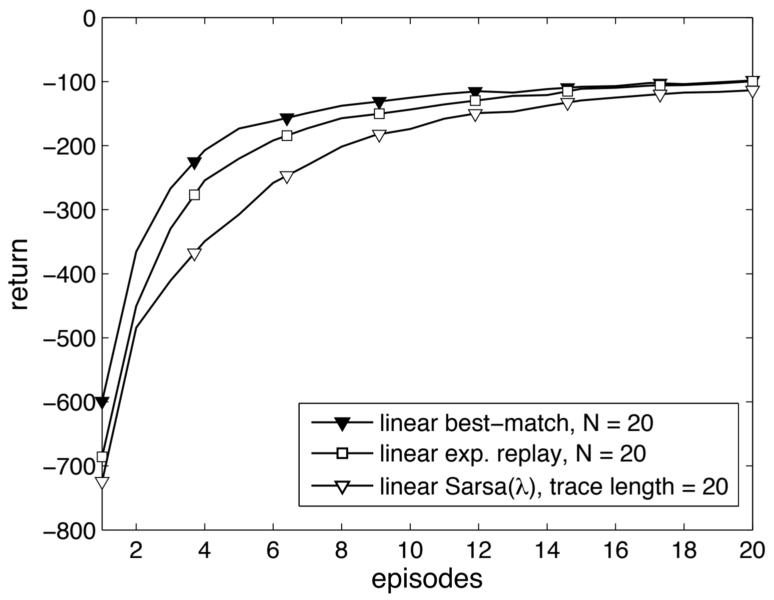


Figure 14: Performance of linear best-match, experience replay and linear Sarsa(λ) on the Mountain Car task using the 20 most recent samples.

a way that the computation times approximately match. We chose to decrease the sample set size of linear best-match. Using $N = 15$ and $N = 5$ resulted in a computation time matching that of experience replay and Sarsa(λ), respectively. Table 5 shows that the performance of linear best-match is also better with equal amount of computation time. In addition, we performed an experiment with Sarsa(λ) without bound on the trace length. This resulted in an average return of -228.2, demonstrating that the performance of Sarsa(λ) cannot be improved significantly by increasing the trace length.

Overall, these results show that best-match learning can be successfully applied to function approximation. Furthermore, they demonstrate that using samples to correct previous updates can lead to better performance that using them to perform additional updates.

7. Discussion

The methods presented in this article approximate solutions to different instantiations of the generalized best-match equations (Definition 11). These best-match equations provide a theoretical foundation for combining model-free learning (through updates of Q^{mf}) with model-based learning (through updates of Q). The resulting methods offer two trade-offs. First, the selection of a sparse, approximate model provides a trade-off between space and performance. Second, the number of best-match updates performed per timestep provides a trade-off between computation cost per timestep and performance. The performance gain offered by best-match learning can be explained from the perspective of the update targets. By performing best-match updates, the update targets from the samples stored in the model are continually recomputed and the Q-values are updated to incorporate any resulting changes.

In the case of best-match LVM, this produces an evaluation method that leads to the same values as TD(λ) with $\lambda_t = \alpha_t(s_t)$ for acyclic tasks, as proven in Theorem 7. This equivalence arises from the fact that both best-match LVM learning and eligibility traces outperform 1-step methods by correcting previous updates with newly obtained samples. However, our theoretical and empirical results suggest that the best-match LVM equations provide a much stronger basis for exploiting this principle.

Theorem 8 proves that best-match LVM evaluation can perform updates that are unbiased with respect to the initial values for an arbitrary MDP, while for TD(λ) this can only be achieved for acyclic tasks. In the control case, Theorem 10 proves convergence in the limit to the optimal Q-values for a general class of best-match LVM control algorithms. Similar converge guarantees do not exist for eligibility traces. In addition, best-match LVM learning avoids the need to choose between different trace types (accumulating or replacing) and does not require an extra λ parameter. Furthermore, in deterministic problems, best-match LVM learning, reduces to model-based learning, as one would expect for an algorithm that makes optimal use of the $O(|\mathcal{S}||\mathcal{A}|)$ space complexity.

Our empirical results show that best-match LVM evaluation substantially outperforms TD(λ) and experience replay (Figure 9), despite having similar computational costs. For the control case, we show that BM-LVM, which uses prioritized sweeping to trade-off computation cost with performance, substantially outperforms not only Q(λ), but also other methods with a space complexity of $O(|\mathcal{S}||\mathcal{A}|)$ (Figure 10). These results illustrate how best-match LVM learning efficiently exploits its stored samples.

Alternatively, best-match learning can be combined with an n-transition model, yielding space complexity between $O(|\mathcal{S}||\mathcal{A}|)$ and $O(|\mathcal{S}|^2|\mathcal{A}|)$. Without using best-match learning, the performance of an NTM is bounded by the quality of the model approximation. In contrast, Theorem 12 proves that BM-NTM converges in the limit to the optimal Q-values. Empirically, we demonstrate that, for any significant space reduction compared to the full model, BM-NTM performs much better than using only the NTM (Figure 13).

Finally, our results demonstrate that the ideas behind best-match learning can be successfully extended to function approximation by combining sequence based best-match learning with gradient descent updates (Algorithm 6). In particular, a linear implementation outperforms Sarsa(λ) and experience replay on a benchmark task (Figure 14).

8. Future Work

Several avenues of future research are suggested by the work presented in this article. For example, in Section 4.2 we proved that the best-match LVM evaluation algorithm can eliminate bias with respect to the initial values. It may be possible to extend this result to the control case. One approach would be to define a state value as the maximum of the Q-values over previously taken actions instead of the maximum over all available actions. However, a potential problem is that the control algorithms compute an approximation of the best-match Q-values, instead of the exact values. It is an open question whether efficient approximations exist that are also unbiased. A second potential problem is that many exploration schemes, such as optimistic initialization, depend on the Q-values and might not work as well when updates are unbiased.

The convergence results for the tabular best-match methods are similar to those of Q-learning: convergence in the limit to the optimal policy. It may be possible, however, to construct best-match methods that are probably approximately correct (PAC). Since Strehl et al. (2006) showed that a full model is not required for a method to be PAC, we are optimistic that such methods exist.

Finally, it may be possible to develop novel combinations of best-match function approximation with other sample-based approaches such as fitted Q-iteration (Ernst et al., 2005) or LSPI (Lagoudakis and Parr, 2003). By combining the strengths of each approach, such methods could yield even better on-line performance. Fitted Q-iteration, for example, is an off-line algorithm that computes a policy based on a large set of samples, by performing iterative update sweeps through the sample set. For a good approximation, the number of samples should be much larger than the number of parameters of the approximation. By using a combination between a model-free Q-value function and a sample set, a smaller sample set might be possible, reducing the requirements with respect to space and computation, and potentially producing an efficient on-line version of fitted Q-iteration.

9. Conclusion

This article introduced best-match learning, a reinforcement learning approach that combines model-free and model-based learning by using some samples to update a sparse model and the rest to update a model-free Q-value. The final Q-values are computed from best-match updates that combine the model-free Q-values with the sparse model. By controlling which samples enter the model, the size of the model, and hence the space requirements, can be controlled. In the tabular case, the combination with the model-free Q-values ensures convergence to the optimal Q-values for a variety of model approximations.

Our empirical results demonstrate that in the tabular case, when there is not enough space available to store the full model, methods that exploit the best-match equations perform substantially better than methods based on only model-free learning or sparse model-based methods. This suggests that best-match learning should be the strategy of choice when limited space is available.

In addition, we demonstrated that best-match learning can be successfully extended to the function approximation domain, where the sparse model is replaced by an explicit set of samples. An interesting result in this domain is that best-match learning, which uses the sample set to correct previous updates, outperforms experience replay, which uses the same sample set but performs additional updates.

Overall, we believe that best-match learning provides an important missing link between model-free and model-based learning and that the methods introduced in this article constitute a new benchmark for reinforcement learning algorithms that are efficient with respect to both space and computation.

Acknowledgments

The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

Appendix A. Proof of Theorem 1

Theorem 1 *Given the same experience sequence, each Q-value from the current state has received the same number of updates using JIT updates (Equation 3) as using regular updates (Equation*

2). However, each Q-value in the update target of a JIT update has received an equal or greater number of updates as in the update target of the corresponding regular update.

Proof To prove the theorem, we need to prove

$$U[\tilde{Q}_t(s_t,a)] = U[Q_t(s_t,a)], \qquad \text{for all } a, \tag{18}$$
$$U[\tilde{Q}_{t-1}(s_{t^*+1},a)] \geq U[Q_{t^*}(s_{t^*+1},a)], \qquad \text{for all } a, \tag{19}$$

where $U[Q_k]$ is the total number of updates a Q-value has received at time k. From Equation 2 and 3 it follows that for both update types (s_t, a_{t^*}) is updated once between timestep t^* and timestep t, while the Q-values of the other actions of s_t are not updated during this period. Since this applies to all visits and $U[\tilde{Q}_0(s,a)] = U[Q_0(s,a)] = 0$ for all s and a, the total number of updates for a state-action pair is always equal for just-in-time updates and regular updates, when the state is the current state, proving (18).

To prove (19), first assume that a_{t^*} is a returning action, that is, $t-1 = t^*$. In this case clearly (19) is true. Now, assume a_{t^*} is not a returning action, that is, $t-1 > t^*$. From (18) it follows that $U[\tilde{Q}_{t^*+1}(s_{t^*+1},a)] = U[Q_{t^*+1}(s_{t^*+1},a)]$. Since $t-1 \geq t^*+1$ and $U[\tilde{Q}]$ increases monotonically over time, it follows that (19) is true. When state s_{t^*+1} is revisited before t, an extra update is performed and there is at least one action a, for which $U[\tilde{Q}_{t-1}(s_{t^*+1},a)] > U[Q_{t^*}(s_{t^*+1},a)]$. ■

Appendix B. Relationship between Best-Match LVM and TD(λ)

Sutton and Singh (1994) showed that it is possible to perform TD updates that are unbiased with respect to the initial values, by using TD(λ) where λ is made time-dependent and set equal to $\alpha_t(s_t)$. However, TD(λ) can be made unbiased only for acyclic tasks, that is, episodic tasks with no revisits of states within an episode. In this appendix, we prove that best-match LVM evaluation and TD(λ) can lead to the same values for acyclic tasks and that best-match LVM evaluation can perform unbiased updates for all MDPs.

B.1 Background on TD(λ)

The forward view of TD(λ) relates it to the λ-*return* (Watkins, 1989; Jaakkola et al., 1994), defined by

$$R_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)},$$

where $R_t^{(n)}$ indicates the *n-step return*, defined by

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}).$$

The λ-*return algorithm* updates state s_t with R_t^{λ}. It can only be implemented off-line, since it makes use of values from timesteps larger than t for the update of state s_t. While the off-line version of TD(λ) computes the same state values as the λ-return algorithm (Sutton and Barto, 1998), TD(λ) can also be implemented on-line, since it does not rely on values from the future. On-line TD(λ) can lead to more accurate updates than off-line TD(λ), although the interpretation as an incremental implementation of the λ-return holds only by approximation for the on-line case (Sutton and Barto, 1998).

The backward view of TD(λ) interprets λ as the trace decay parameter of an eligibility trace. Each sample is used to update, not just the current state, but all states, proportional to their *trace* parameter. At each timestep the trace of the current state is increased, while the other traces are decreased by $\gamma\lambda$. *Accumulating traces* increase the trace parameter of a visited state by 1, while *replacing traces* set it equal to 1.

Sutton and Singh (1994) proposed several ways for setting α and λ that eliminate bias towards initial state values, normally inherent to temporal-difference methods. One of these is to use TD(λ) where $\lambda_t = \alpha_t(s_t)$ and $\alpha_0(s) = 1$ for all s. This produces the same values as processing a state backwards with TD(0). All the proposed methods eliminate the bias only for acyclic tasks.

The equation for the λ-return with time-dependent λ is (Sutton and Barto, 1998)

$$\begin{aligned} R_t^{\lambda_t} &= \sum_{n=1}^{\infty} R_t^{(n)}(1-\lambda_{t+n})\prod_{i=1}^{n-1}\lambda_{t+i} \\ &= \sum_{n=1}^{T-t-1} R_t^{(n)}(1-\lambda_{t+n})\prod_{i=1}^{n-1}\lambda_{t+i} + R_t\prod_{i=1}^{T-t-1}\lambda_{t+i}\,, \end{aligned} \tag{20}$$

where T is the last timestep of the episode and R_t is the complete return. Note that $R_t = R_t^{(T-t)}$.

B.2 Forward View Best-Match LVM Values

The λ-return is based on the experience sequence encountered by the agent when interacting with the environment. We can define for each visited state a *last-visit experience sequence* based on the LVM by going through the transition states defined in the LVM. Using this sequence we define a last-visit version of the *n*-step return and of a special version of the λ-return.

Definition 13 *The last-visit experience sequence for state s is*

$$s_{[0]}, r_{[1]}, s_{[1]}, r_{[2]}, s_{[2]}, \ldots, r_{[N]}, s_{[N]}\,,$$

where $s_{[0]} = s$, $s_{[n]} = S'(s_{[n-1]})$ *for* $n > 0$ *and* $r_{[n]} = R'(s_{[n-1]})$. *The sequence ends when a state is encountered that is terminal, equal to* $s_{[0]}$ *or that has no transition state. We call* $s_{[N]}$ *the last-visit sequence end state.*

Using this definition, we define a last-visit version of the n-step return.

Definition 14 *The last-visit n-step return of s is the n-step return applied to the last-visit experience sequence of s:*

$$\breve{R}_s^{(n)} = r_{[1]} + \gamma r_{[2]} + \gamma^2 r_{[3]} + \ldots + \gamma^{n-1} r_{[n]} + \gamma^n V^{mf}(s_{[n]})\,. \tag{21}$$

We can now define a special version of the λ-return, which we call the *last-visit* α*-return*: a last-visit version of the time dependent λ-return (Equation 20) with $\lambda_t = \alpha_t(s_t)$.

Definition 15 *The last-visit* α*-return of s is*

$$\breve{R}_s^{\alpha} = \sum_{n=1}^{N-1} \breve{R}_s^{(n)}(1-\alpha^{[n]})\prod_{i=1}^{n-1}\alpha^{[i]} + \breve{R}_s^{(N)}\prod_{i=1}^{N-1}\alpha^{[i]}\,, \tag{22}$$

where $\alpha^{[k]}$ *is shorthand for* $\alpha(s_{[k]})$, $s_{[k]}$ *is the* k^{th} *state from the last-visit experience sequence of s and N is the index of the last-visit sequence end state.*

The following lemma relates the last-visit α-return of s to the best-match value of s. The lemma is proven in Appendix C.

Lemma 16 *If the last-visit sequence end state of s is a terminal state, the following equation holds for the best-match value of s:*

$$V^B(s) = (1-\alpha^s)V^{mf}(s) + \alpha^s \breve{R}_s^{\alpha}.$$

This lemma forms the basis for the proof of the following theorem.

Theorem 7 *For an episodic, acyclic, evaluation task, off-line best-match LVM evaluation computes the same values as off-line TD(λ) with* $\lambda_t = \alpha_t(s_t)$.

Proof Let V_k be the state value function after the off-line updates at the end of episode k. For all states that are visited during an episode, V is updated according to Lemma 16, since the last-visit sequence end state is a terminal state for all these visited states. For the off-line algorithm, before $V_k(s)$ is computed, the update $V_k^{mf}(s) = V_{k-1}(s)$ is performed for all visited states. Therefore, the value updates of the visited states can be written as

$$V_k(s) = (1-\alpha^s)V_{k-1}(s) + \alpha^s \breve{R}_s^{\alpha}.$$

If the task is acyclic, the last-visit experience sequence of a visited state s is equal to the experience sequence followed by the agent from this state to the terminal state. Therefore, $\breve{R}_s^{\alpha} = R_t^{\lambda=\alpha_t(s_t)}$. Finally, since the values computed by off-line TD(λ) are equal to the values computed by the λ-return algorithm, off-line TD(λ) with $\lambda_t = \alpha_t(s_t)$ performs the same updates as off-line best-match LVM evaluation. ■

It follows from Theorem 7 that best-match evaluation can also eliminate the bias for acyclic tasks. The next theorem extends this property to a general MDP.

Theorem 8 *The state values computed by the on-line best-match LVM evaluation algorithm (Algorithm 2) are unbiased with respect to the initial state values, when the initial learning rates* $\alpha_0(s)$ *are set to 1 for all s.*

Proof Algorithm 2 computes at each timestep the best-match value of the current state. We will prove that if the best-match values of visited states computed at timesteps smaller than t are unbiased with respect to the initial state values, then so is the best-match value computed at timestep t. Since for $t = 0$ there are no visited states, it follows by induction that the values computed for all timesteps t are unbiased.

The best-match values are computed using $V^B(s_{[0]}) = c_A + c_B V^B(s_{[N]})$ with c_A and c_B defined as in (8) and (9) respectively. In Section 4.2 we showed that for the current state, $s_{[N]}$ is either a terminal state or equal to $s_{[0]}$. If $s_{[N]}$ is a terminal state, $V^B(s_{[0]}) = c_A$, while if $s_{[0]} = s_{[N]}$, then $V^B(s_{[0]}) = c_A/(1-c_B)$. In either case, the computed best-match value depends only on the variables in c_A and c_B, which consists of the learning rates, $V^{mf}(s_{[i]})$, $s_{[i]}$ and $r_{[i]}$ for $0 \leq i \leq N$. Clearly, only $V^{mf}(s_{[i]})$ can be affected by the initial state values. $s_{[i]}$ has been visited at least once, else it would not appear in the last-visit experience sequence. If $s_{[i]}$ has been visited once, $V^{mf}(s_{[i]})$ is equal to the initial value $V_0(s_{[i]})$. However, since we assumed initial learning rates of 1, this value of $V^{mf}(s_{[i]})$ is removed from c_A. If $s_{[i]}$ has been visited more than once, it is equal to the best-match value of $s_{[i]}$ computed at a timestep smaller than t. From this it follows that if the best-match values computed at timesteps smaller than t are unbiased with respect to the initial values, then so is the best-match value computed at timestep t. ∎

Appendix C. Proof of Lemma 16

For the sake of brevity, we present only the proof of Lemma 16 for constant α. The proof for state dependent α follows the same pattern.

Lemma 16 *If the last-visit sequence end state of s is a terminal state, the following equation holds for the best-match value of s:*

$$V^B(s) = (1-\alpha^s)V^{mf}(s) + \alpha^s \breve{R}^{\alpha}_s .$$

Proof The best-match values in case of an LVM are defined as the solution of the set of best-match LVM equations (Definition 6). In Section 4.2 we showed that by backward substitution of best-match equations we can express the best-match value of $s_{[0]}$ in terms of the best-match value of $s_{[N]}$. If $s_{[N]}$ is a terminal state, $V^B(s_{[N]}) = 0$ and $V^B(s_{[0]})$ is equal to c_A defined as in (8). This yields

$$\begin{aligned} V^B(s_{[0]}) &= \sum_{i=0}^{N-1} \big((1-\alpha)V^{mf}(s_{[i]}) + \alpha r_{[i+1]}\big) \prod_{k=0}^{i-1} \gamma\alpha , \\ &= \alpha \sum_{k=1}^{N} (\alpha\gamma)^{k-1} r_{[k]} + (1-\alpha) \sum_{k=0}^{N-1} (\alpha\gamma)^k V^{mf}(s_{[k]}) . \end{aligned} \tag{23}$$

On the other hand, by substituting the definitions of the last-visit α-return (22) and the last-visit n-step return (21) into the lemma, the following equation for $V^B(s_{[0]})$ appears:

$$\begin{aligned} V^B(s_{[0]}) &= (1-\alpha)V^{mf}(s_{[0]}) + \alpha \Bigg[(1-\alpha) \sum_{k=1}^{N-1} \alpha^{k-1} \left(\sum_{p=1}^{k} \gamma^{p-1} r_{[p]} + \gamma^k V^{mf}(s_{[k]}) \right) \\ &\quad + \alpha^{N-1} \sum_{p=1}^{N} \gamma^{p-1} r_{[p]} \Bigg] . \end{aligned} \tag{24}$$

The rest of this proof shows that (23) is equal to (24).

We start by separating (24) into its state value components (V^c) and its reward components (R^c). We then simplify these components separately:

$$\begin{aligned} V^c &= (1-\alpha)V^{mf}(s_{[0]}) + \alpha(1-\alpha) \sum_{k=1}^{N-1} \alpha^{k-1}\gamma^k V^{mf}(s_{[k]}) \\ &= (1-\alpha)\left(V^{mf}(s_{[0]}) + \sum_{k=1}^{N-1} (\alpha\gamma)^k V^{mf}(s_{[k]}) \right) \\ &= (1-\alpha) \sum_{k=0}^{N-1} (\alpha\gamma)^k V^{mf}(s_{[k]}) , \end{aligned}$$

$$\begin{aligned}
R^c &= (1-\alpha)\sum_{k=1}^{N-1}\sum_{p=1}^{k}\alpha^k\gamma^{p-1}r_{[p]} + \alpha^N\sum_{p=1}^{N}\gamma^{p-1}r_{[p]} \\
&= (1-\alpha)\sum_{p=1}^{N-1}\sum_{k=p}^{N-1}\alpha^k\gamma^{p-1}r_{[p]} + \alpha^N\sum_{p=1}^{N-1}\gamma^{p-1}r_{[p]} + \alpha^N\gamma^{N-1}r_{[N]} \\
&= \sum_{p=1}^{N-1}\left[(1-\alpha)\sum_{k=p}^{N-1}\alpha^k\gamma^{p-1}r_{[p]} + \alpha^N\gamma^{p-1}r_{[p]}\right] + \alpha^N\gamma^{N-1}r_{[N]} \\
&= \sum_{p=1}^{N-1}\left[\left(\sum_{k=p}^{N-1}\alpha^k - \sum_{k=p}^{N-1}\alpha^{k+1} + \alpha^N\right)\gamma^{p-1}r_{[p]}\right] + \alpha^N\gamma^{N-1}r_{[N]} \\
&= \sum_{p=1}^{N-1}\left[\left(\sum_{k=p}^{N}\alpha^k - \sum_{k=p}^{N-1}\alpha^{k+1}\right)\gamma^{p-1}r_{[p]}\right] + \alpha^N\gamma^{N-1}r_{[N]} \\
&= \sum_{p=1}^{N-1}\left[\left(\sum_{j=p-1}^{N-1}\alpha^{j+1} - \sum_{k=p}^{N-1}\alpha^{k+1}\right)\gamma^{p-1}r_{[p]}\right] + \alpha^N\gamma^{N-1}r_{[N]} \\
&= \sum_{p=1}^{N-1}\left[\alpha^p\gamma^{p-1}r_{[p]}\right] + \alpha^N\gamma^{N-1}r_{[N]} \\
&= \alpha\sum_{p=1}^{N}(\alpha\gamma)^{p-1}r_{[p]}\,.
\end{aligned}$$

Adding these simplified components back together yields Equation 23. ■

Appendix D. Proof of Theorem 10

Theorem 10 *The Q-values of a member of the best-match LVM control class, shown in Algorithm 3, converge to* Q^* *if the following conditions are satisfied:*

1. *S and A are finite.*
2. $\alpha_t(s,a) \in [0,1]$, $\sum_t \alpha_t(s,a) = \infty$, $\sum_t(\alpha_t(s,a))^2 < \infty$ *with probability 1 (w.p.1) and* $\alpha_t(s,a) = 0$ *unless* $(s,a) = (s_t,a_t)$.
3. $Var\{R(s,a,s')\} < \infty$.
4. $\gamma < 1$.

Proof We prove that the Q-values of an arbitrary instantiation of Algorithm 3 converge in the limit w.p.1 to those of the regular Q-learning algorithm. Because the algorithm requires that each visited state action pair is updated at least once before its revisit, the following equation holds

$$Q^{mf}_{t+1}(s_t,a_t) = (1-\alpha_t(s_t,a_t))\,Q^{mf}_t(s_t,a_t) + \alpha_t(s_t,a_t)\left(r_{t^*+1} + \max_{a'} Q_{\tau,i}(s_{t^*+1},a')\right),$$

where t^* is the timestep of the previous visit of (s_t, a_t) and $Q_{\tau,i}$ is the Q-value of s_{t^*+1} that is used in the update target of the last best-match update of (s_t, a_t), at timestep τ. Note that $t^* + 1 \leq \tau \leq t$. Assume that Q-learning is applied to the same state-action sequence produced by the given instantiation of Algorithm 3. We denote the Q-values from Q-learning by $\tilde{Q}$. Subtracting the update equation for Q-learning at time t^*+1 using learning rate $\alpha_t(s_t, a_t)$ and defining $\Delta_t(s,a) = Q_t^{mf}(s,a) - \tilde{Q}_{t^*}(s,a)$ yields

$$\Delta_{t+1}(s_t, a_t) = (1 - \alpha_t(s_t, a_t))\Delta_t(s_t, a_t) + \alpha_t(s_t, a_t) F_t(s_t, a_t), \tag{25}$$

where $F_t(s_t, a_t) = \gamma \left(\max_c Q_{\tau,i}(s_{t^*+1}, c) - \max_c \tilde{Q}_{t^*}(s_{t^*+1}, c) \right)$.

We now prove that Q_t^{mf} and Q_{t^*} converge in the limit to each other using the same lemma used to prove the convergence of Sarsa (Singh et al., 2000):

Lemma 17 *Consider a stochastic process $(\alpha_t, \Delta_t, F_t)$, $t \geq 0$, where $\alpha_t, \Delta_t, F_t : X \to \mathbb{R}$ satisfy the equations:*

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\Delta_t(x) + \alpha_t(x) F_t(x) \quad ,$$

where $x \in X$ and $t = 0, 1, 2, \ldots$. Let P_t be a sequence of increasing σ-fields such that α_0 and Δ_0 are P_0-measurable and ζ_t, Δ_t and F_{t-1} are P_t-measurable, $t = 1, 2, \ldots$. Assume that the following conditions hold:

1. *The set X is finite.*
2. $\alpha_t(x) \in [0,1]$, $\sum_t \alpha_t(x) = \infty$, $\sum_t (\alpha_t(x))^2 < \infty$ *w.p.1 .*
3. $\|E\{F_t | P_t\}\| \leq \kappa \|\Delta_t\| + c_t$*, where* $\kappa \in [0,1)$ *and* c_t *converges to zero w.p.1, and*
4. *Var*$\{F_t(x_t)|P_t\} \leq K(1 + \kappa\|\Delta_t\|)^2$*, where K is some constant,*

where $\|\cdot\|$ denotes a maximum norm. Then Δ_t converges to zero with probability one.

The correspondence of (25) to Lemma 17 follows from associating X with the set of state-action pairs (s,a) and $\alpha_t(x)$ with $\alpha_t(s,a)$. We now prove that the 4 conditions hold.

The first two conditions follow from the first two conditions of Theorem 10. We define P_t as the set $\{Q_0, \alpha_0, a_0, s_0, ..., r_{t-1}, \alpha_t, a_t, s_t\}$. With this definition, $\text{Var}\{F_t(s_t, a_t)|P_t\} = 0$, satisfying condition 4, and $E\{F_t(s_t, a_t)|P_t\} = F_t(s_t, a_t)$. For $|F_t(s_t, a_t)|$ the following holds:

$$\begin{aligned}
|F_t(s_t, a_t)| &= \gamma \,|\max_b Q_{\tau,i}(s_{t^*+1}, b) - \max_b \tilde{Q}_{t^*}(s_{t^*+1}, b)| \\
&\leq \gamma ||Q_{\tau,i}(u,b) - \tilde{Q}_{t^*}(u,b)|| \\
&= \gamma ||\Delta_t(u,b) + Q_{\tau,i}(u,b) - Q_t^{mf}(u,b)|| \\
&\leq \gamma ||\Delta_t|| + ||Q_{\tau,i}(u,b) - Q_t^{mf}(u,b)|| .
\end{aligned}$$

We further define $F_t(s,a) = 0$ if $(s,a) \neq (s_t, a_t)$. Therefore, $||F_t(s,a)|| = |F_t(s_t, a_t)| \leq \gamma ||\Delta_t|| + C_t$, where $C_t = ||Q_{\tau,i}(u,b) - Q_t^{mf}(u,b)||$. We now show that C_t converges to zero w.p.1. For C_t, the following holds:

$$C_t \leq ||Q_{\tau,i}(u,b) - Q_{\tau^*}^{mf}(u,b)|| + ||Q_{\tau^*}^{mf}(u,b) - Q_t^{mf}(u,b)|| ,$$

where τ^* is the timestep of the last visit of (u,b) before timestep τ. $Q_{\tau,i}(u,b)$ is the result of a best-match update of $Q^{mf}_{\tau^*}(u,b)$ or is equal to it if no best-match update has been performed yet. In the latter case, the first term is zero; in the former case it is

$$Q_{\tau,i}(u,b) = (1-\alpha_\tau(u,b))Q^{mf}_{\tau^*}(u,b) + \alpha_\tau(u,b)\upsilon^{ub}_\tau .$$

Because of condition 2 of Theorem 10, $\alpha_\tau(u,b)$ converges to 0 w.p.1 and $Q_{\tau,i}(u,b)$ converges to $Q^{mf}_{\tau^*}(u,b)$ w.p.1. Therefore, the first term converges to 0 w.p.1. For the same reason, the second term converges to zero.

Thus, the third condition of the lemma also holds and $Q^{mf}(s,a)$ converges to $\tilde{Q}(s,a)$, the Q-values from Q-learning. Because of the convergence guarantee of Q-learning, $Q^{mf}(s,a)$ also converges to $Q^*(s,a)$. Finally, since the Q-values of the given instantiation are a best-match update of $Q^{mf}(s,a)$ and because $\alpha_t(s,a)$ converges to zero w.p.1, this also proves that the Q-values of the instantiation converge to Q^*. ■

Appendix E. Proof of Theorem 12

Theorem 12 *The Q-values of a member of the best-match NTM control class, shown in Algorithm 5, converge to Q^* if the following conditions are satisfied:*

1. *S and A are finite.*
2. *$\alpha^{sa}_t \in [0,1]$, $\sum_t \alpha^{sa}_t = \infty$, $\sum_t (\alpha^{sa}_t)^2 < \infty$ with probability 1 (w.p.1), and $\alpha^{sa}_t = 0$ unless $(s,a) = (s_t,a_t)$ and $s_{t+1} \notin NTM(s_t,a_t)$.*
3. *$Var\{R(s,a,s')\} < \infty$.*
4. *$\gamma < 1$.*

E.1 Preliminaries

In this proof, we indicate the NTM by $\mathcal{M}$. Also, we indicate the model-free Q-value, Q^{mf}, by $\breve{Q}$. In addition, we use a single iteration index j for Q as well as $\breve{Q}$. This global index is increased each time an update (of either $\breve{Q}$ or Q) occurs. Thus, j is equal to the total number of model-free updates plus best-match updates that have occurred since the start of an episode. Clearly, $t \to \infty$ implies $j \to \infty$.

By denoting the state-action pair that gets updated by the j-th update as (s_j,a_j), we can write the model-free (mf) update as

$$\breve{Q}_{j+1}(s_j,a_j) = (1-\alpha^{s_j a_j})\breve{Q}_j(s_j,a_j) + \alpha^{s_j a_j}\left[r_{j+1} + \gamma \max_{a'} Q_j(s'_{j+1},a')\right], \tag{26}$$

where r_{j+1} and s'_{j+1} are the reward and transition state from the sample $(s_t,a_t,r_{t+1},s_{t+1})$ corresponding to (s_j,a_j). We use s'_{j+1} instead of s_{j+1}, since s'_{j+1}, the transition state for s_j, is in general not equal to s_{j+1}, the state that receives an update at iteration step $j+1$. The best-match (bm) update is

$$Q_{j+1}(s_j,a_j) = w_0^{s_j,a_j}\breve{Q}_j(s_j,a_j) + (1-w_0^{s_j a_j})\left[\hat{\mathcal{R}}_{s_j a_j} + \gamma \sum_{s'} \hat{\mathcal{P}}^{s'}_{s_j a_j} \max_{a'} Q_j(s',a')\right].$$

Note that there is no specific sample corresponding to a best-match update, since the update is based on the model estimate and can occur multiple times per timestep.

Let $\mathcal{P}_{sa}^{\mathcal{M}} = \sum_{s' \in \mathcal{M}} \mathcal{P}_{sa}^{s'}$. If $\mathcal{P}_{sa}^{\mathcal{M}} = 0$, w_0^{sa} will always be 1 and the best-match update reduces to $Q_{j+1}(s_j, a_j) = \breve{Q}_j(s_j, a_j)$. We make this explicit by the following equation:

$$Q_{j+1}(s_j,a_j) = \begin{cases} \breve{Q}_j(s_j,a_j) & \text{if } \mathcal{P}_{sa}^{\mathcal{M}} = 0 \\ Y_j(s_j,a_j) & \text{if } \mathcal{P}_{sa}^{\mathcal{M}} > 0\,, \end{cases} \tag{27}$$

with

$$Y_j(s_j,a_j) = w_0^{s_j,a_j}\breve{Q}_j(s_j,a_j) + (1 - w_0^{s_j a_j})\left[\hat{\mathcal{R}}_{s_j a_j} + \gamma \sum_{s'} \hat{\mathcal{P}}_{s_j a_j}^{s'} \max_{a'} Q_j(s',a')\right].$$

Each time a sample is observed by the algorithm, w_0 gets updated. In addition, when the sample is part of $\mathcal{M}$, $\hat{\mathcal{R}}$ and $\hat{\mathcal{P}}$ get updated. Therefore, the values of these variables can change between iteration steps. However, for readability, we omit the j subscript for these variables. From the definition of w_0, $\hat{\mathcal{R}}$ and $\hat{\mathcal{P}}$, and the law of large numbers, it follows that in the limit the following holds:[7]

$$\begin{aligned} \lim_{j\to\infty} w_0^{sa} &= 1 - \mathcal{P}_{sa}^{\mathcal{M}}\,, && (28)\\ \lim_{j\to\infty} \hat{\mathcal{R}}_{sa} &= \sum_{s' \in \mathcal{M}} \mathcal{P}_{sa}^{s'} \mathcal{R}_{sa}^{s'} / \mathcal{P}_{sa}^{\mathcal{M}}\,, && (29)\\ \lim_{j\to\infty} \hat{\mathcal{P}}_{sa}^{s'} &= \mathcal{P}_{sa}^{s'} / \mathcal{P}_{sa}^{\mathcal{M}}\,. && (30) \end{aligned}$$

In general, the model-free Q-values, $\breve{Q}$, will not converge to Q^*, since they do not receive updates from samples corresponding to the next states stored by the NTM. However, as part of the proof, we show that the model-free Q-values converge to an alternative value, which we indicate by $\breve{Q}^*$. This value is defined as[8]

$$\breve{Q}^*(s,a) = \sum_{s' \notin \mathcal{M}} \mathcal{P}_{sa}^{s'} [\mathcal{R}_{sa}^{s'} + \gamma \max_{a'} Q^*(s',a')] / (1 - \mathcal{P}_{sa}^{\mathcal{M}})\,. \tag{31}$$

Using this equation, we can express Q^* as

$$\begin{aligned} Q^*(s,a) &= \sum_{s' \notin \mathcal{M}} \mathcal{P}_{sa}^{s'}[\mathcal{R}_{sa}^{s'} + \gamma \max_{a'} Q^*(s',a')] + \sum_{s' \in \mathcal{M}} \mathcal{P}_{sa}^{s'}[\mathcal{R}_{sa}^{s'} + \gamma \max_{a'} Q^*(s',a')] \\ &= (1 - \mathcal{P}_{sa}^{\mathcal{M}})\breve{Q}^*(sa) + \sum_{s' \in \mathcal{M}} \mathcal{P}_{sa}^{s'}[\mathcal{R}_{sa}^{s'} + \gamma \max_{a'} Q^*(s',a')]\,. && (32) \end{aligned}$$

Note that it follows from (32), that

$$Q^*(s,a) = \breve{Q}^*(s,a)\,, \qquad \text{if } \mathcal{P}_{sa}^{\mathcal{M}} = 0\,. \tag{33}$$

Convergence of Q_j to Q^* requires convergence of $\breve{Q}_j$ to $\breve{Q}^*$, and vice versa. To deal with this mutual dependence relation, we simultaneously prove their convergence. To achieve this, we define

7. Note that $\hat{\mathcal{R}}_{sa}$ and $\hat{\mathcal{P}}_{sa}^{s'}$ do not converge to $\mathcal{R}_{sa}$ and $\mathcal{P}_{sa}^{s'}$, but to normalized values of these variables.

8. For $\mathcal{P}_{sa}^{\mathcal{M}} = 1$, that is, when all samples are stored by the NTM, $\breve{Q}^*(s,a)$ is not defined. However, in this case, $\breve{Q}(s,a)$ does not receive any updates, nor is it used by any other update. Therefore, we can safely ignore the value $\breve{Q}(s,a)$, and consequently $\breve{Q}^*(s,a)$, if $\mathcal{P}_{sa}^{\mathcal{M}} = 1$.

a function $U : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \to \mathrm{I\!R}$ that encompasses both functions Q and $\breve{Q}$. $\mathcal{B}$ is a set consisting of only two elements: 'mf' and 'bm', which indicate the *Q-value type*. We define U_j as

$$U_j(s,a,b) = \begin{cases} \breve{Q}_j(s,a) & \text{if b = 'mf'} \\ Q_j(s,a) & \text{if b = 'bm'}. \end{cases} \tag{34}$$

Both updates (26) and (27) can now be interpreted as updates of $U_j(s_j,a_j,b_j)$. It follows from (34) that when the model-free update is performed, b_j = 'mf', while for the best-match update b_j = 'bm'.

We will prove convergence of U_j to U_j^*, defined as

$$U^*(s,a,b) = \begin{cases} \breve{Q}^*(s,a) & \text{if b = 'mf'} \\ Q^*(s,a) & \text{if b = 'bm'}. \end{cases}$$

The difficulty with this proof is that we cannot simply apply Lemma 17 (or similar stochastic approximation lemmas), used to prove convergence of BM-LVM, since the $\sum_t(\alpha_t(x_t))^2 < \infty$ condition of Lemma 17 is not met for b = 'bm'. On the other hand, a related lemma can be deduced (see Appendix F), that does not require $\sum_t(\alpha_t(x_t))^2 < \infty$, however, it requires that the contraction condition holds for the value of F_t, instead of its expected value. Hence, also this lemma cannot be directly applied.

To deal with this, we define a related function U'_j, that does comply with the $\sum_t(\alpha_t(x_t))^2 < \infty$ condition, hence we can prove convergence of it to U^* using Lemma 17. On the other hand, the difference between U'_j and U_j complies with all the conditions of Lemma 20, hence we can prove that U_j converges to U'_j using Lemma 20. Adding these two results together, proves the theorem.

We define U'_j as

$$U'_j(s,a,b) = \begin{cases} \breve{Q}'(s,a) & \text{if b = 'mf'} \\ Q'(s,a) & \text{if b = 'bm'}. \end{cases}$$

$\breve{Q}'$ and Q' are updated using the same sample sequence as used for $\breve{Q}$ and Q. The update for $\breve{Q}'$ is

$$\breve{Q}'_{j+1}(s_j,a_j) = (1-\alpha^{s_j a_j})\breve{Q}'_j(s_j,a_j) + \alpha^{s_j a_j}\left[r_{j+1} + \gamma \max_{a'} Q'_j(s'_{j+1},a')\right],$$

while the update for Q' is

$$Q'_{j+1}(s_j,a_j) = \begin{cases} \breve{Q}'_j(s_j,a_j) & \text{if } \mathcal{P}^{\mathcal{M}}_{sa} = 0 \\ (1-\beta^{s_j a_j})Q'_j(s_j,a_j) + \beta^{s_j a_j} Y'_j(s_j,a_j) & \text{if } \mathcal{P}^{\mathcal{M}}_{sa} > 0, \end{cases} \tag{35}$$

with

$$Y'_j(s_j,a_j) = w_0^{s_j,a_j}\breve{Q}'_j(s_j,a_j) + (1-w_0^{s_j a_j})\left[\hat{\mathcal{R}}_{s_j a_j} + \gamma \sum_{s'} \hat{\mathcal{P}}^{s'}_{s_j a_j} \max_{a'} Q'_j(s',a')\right].$$

Note that the only difference with the updates of Q and $\breve{Q}$ is the way Q' is updated for $\mathcal{P}^{\mathcal{M}}_{sa} > 0$. Instead of setting $Q'_{j+1}(s_j,a_j)$ equal to $Y'_j(s_j,a_j)$, it is set equal to a weighted average of $Y'_j(s_j,a_j)$ and $Q'_j(s_j,a_j)$. The weighting is controlled by β_j, which is an arbitrary learning rate with properties $\beta^{sa}_j \in [0,1]$, $\sum_j \beta^{sa}_j = \infty$, $\sum_j(\beta^{sa}_j)^2 < \infty$ w.p.1., and $\beta^{sa}_j = 0$ unless $(s,a) = (s_j,a_j)$ and b_j = 'bm'.[9] Because of this learning rate, Lemma 17 can be used to prove convergence of U'_j to U^*.

9. Note that such a β always exists.

E.2 Convergence of U'_j to U^*

Lemma 18 *$U'_j(s,a,b)$ converges in the limit to $U^*(s,a,b)$ w.p.1.*

Proof We define $\Delta'(s,a,b) = U'_j(s,a,b) - U^*_j(s,a,b)$ and will prove that $\Delta'(s,a,b)$ converges to 0 using Lemma 17. For $b_j =$ 'bm', we use the contraction factor κ^{sa}, defined as

$$\kappa^{sa} = (1 - \mathcal{P}^{\mathcal{M}}_{sa}) + \gamma\,\mathcal{P}^{\mathcal{M}}_{sa}\,. \tag{36}$$

To ensure that $\kappa^{sa} < 1$, $\mathcal{P}^{\mathcal{M}}_{sa}$ has to be larger than 0. Therefore, we exclude (s,a,b) triples for which $b =$ 'bm' $\wedge\, \mathcal{P}^{\mathcal{M}}_{sa} = 0$ from the domain of Δ'. This can be done, because Algorithm 5 states that at least one best-match update occurs in between two model-free updates. Therefore, if $\mathcal{P}^{\mathcal{M}}_{sa} = 0$, $Q'_j(s,a)$ is either equal to $\breve{Q}'_j(s,a)$ or one (model-free) update apart. Since α^{sa}_j converges to 0, it follows that $Q'_j(s,a)$ converges in the limit to $\breve{Q}'_j(s,a)$. Alternatively, we can say

$$Q'_j(s,a) = \breve{Q}'_j(s,a) + c'_j(s,a)\,, \qquad \text{if } \mathcal{P}^{\mathcal{M}}_{sa} = 0\,, \tag{37}$$

with $c'_j(s,a)$ converging to 0 w.p.1.[10] Combining this with (33), the following holds:

$$\lim_{j\to\infty} \breve{Q}'_j(s,a) = \breve{Q}^*(s,a) \quad \Rightarrow \quad \lim_{j\to\infty} Q'_j(s,a) = Q^*(s,a)\,, \qquad \text{if } \mathcal{P}^{\mathcal{M}}_{sa} = 0\,. \tag{38}$$

Note, $\|\breve{Q}'_j - \breve{Q}^*\| \le \|\Delta'_j\|$. However, because of the exclusion of $(s,a,\text{'bm'})$ triples with $\mathcal{P}^{\mathcal{M}}_{sa} = 0$, $\|Q'_j - Q^*\| \le \|\Delta_j\|$ does not hold in general. Instead, the following holds:

$$\begin{aligned}
\|Q'_j - Q^*\| &= \max(\|Q'_j - Q^*\|_{P^{\mathcal{M}}_{sa}>0}, \|Q'_j - Q^*\|_{P^{\mathcal{M}}_{sa}=0}) \\
&\le \max(\|Q'_j - Q^*\|_{P^{\mathcal{M}}_{sa}>0}, \|\breve{Q}'_j - \breve{Q}^*\|_{P^{\mathcal{M}}_{sa}=0} + \|c'_j\|) \\
&\le \max(\|U'_j - U^*\|, \|U'_j - U^*\| + \|c'_j\|) \\
&= \|U'_j - U^*\| + \|c'_j\| \\
&= \|\Delta'\| + \|c'_j\|\,.
\end{aligned}$$

Because of the exclusion of the (s,a,b) triples mentioned above, for all $(s,a,\text{'bm'})$ triples in the domain of Δ'_j, $\mathcal{P}^{\mathcal{M}}_{sa} > 0$.

Δ'_j is updated according to

$$\Delta'_{j+1}(s,a,b) = (1 - \zeta'_j(s,a,b))\Delta'_j(s,a,b) + \zeta'_j(s,a,b)\,F'_j(s,a,b)\,.$$

For $(s,a,b) \neq (s_j,a_j,b_j)$, $\zeta'_j(s,a,b) = 0$ and $F'_j(s,a,b) = 0$. For (s_j,a_j,b_j) the following holds:

$$\zeta'_j(s_j,a_j,b_j) = \begin{cases} \alpha^{s_j a_j}_j & \text{if } b_j = \text{'mf'} \\ \beta^{s_j a_j}_j & \text{if } b_j = \text{'bm'}\,, \end{cases}$$

$$F'_j(s_j,a_j,b_j) = \begin{cases} r_{j+1} + \gamma \max_{a'} Q'_j(s'_{j+1},a') - \breve{Q}^*(s_j,a_j) & \text{if } b_j = \text{'mf'} \\ Y'_j(s_j,a_j) - Q^*(s_j,a_j) & \text{if } b_j = \text{'bm'}\,. \end{cases}$$

10. We use the notational convention to indicate variables that converge to 0 with probability 1 with lowercase, Latin letters: c, d, e, ...

We now prove that Δ'_j converges to zero, by showing the conditions for Lemma 17 hold, using the σ-field P_j, defined as[11]

$$\begin{aligned} P_0 &= \{Q'_0, \breve{Q}'_0, \zeta_0, w_{0,0}, \breve{\mathcal{P}}_0, \breve{\mathcal{R}}_0, s_0, a_0\}\,, \\ P_j &= P_{j-1} \cap \{r_j, s'_j, \zeta_j, w_{0,j}, \breve{\mathcal{P}}_j, \breve{\mathcal{R}}_j, s_j, a_j\}\,. \end{aligned}$$

Conditions 1, 2 and 4 of the Lemma 17 follow from conditions 1,2, and 3 of Theorem 12 and the conditions that hold for β_j^{sa}. Condition 3 of the lemma, we prove below.

For b_j = 'mf', using (31), the following holds:

$$\begin{aligned} |E\{F'_j(s_j,a_j,\text{'mf'})|P_j\}| &= \Big|\sum_{s'\notin\mathcal{M}} \mathcal{P}_{s_ja_j}^{s'}[\mathcal{R}_{s_ja_j}^{s'} + \gamma\max_{a'} Q'_j(s',a')]/(1-\mathcal{P}_{s_ja_j}^{\mathcal{M}}) - \breve{Q}^*(s_j,a_j)\Big| \\ &= \gamma\sum_{s'\notin\mathcal{M}} \mathcal{P}_{s_ja_j}^{s'}\Big|\max_{a'} Q'_j(s',a') - \max_{a'} Q^*(s',a')\Big|/(1-\mathcal{P}_{s_ja_j}^{\mathcal{M}}) \\ &\leq \gamma\|Q'_j - Q^*\| \\ &\leq \gamma\|\Delta'_j\| + \gamma\|c'_j\|\,. \end{aligned} \tag{39}$$

For b_j = 'bm', using (32), we can write

$$\begin{aligned} |F'_j(s_j,a_j,\text{'bm'})| &= |Y'_j(s_j,a_j) - Q^*(s_j,a_j)| \\ &\leq \Big|(1-\mathcal{P}_{s_ja_j}^{\mathcal{M}})(\breve{Q}'_j(s_j,a_j) - \breve{Q}^*(s_j,a_j)) \\ &\qquad + \gamma\sum_{s'\in\mathcal{M}} \mathcal{P}_{s_ja_j}^{s'}[\max_{a'} Q'_j(s',a') - \max_{a'} Q^*(s',a')]\Big| \\ &+ \Big|\Big[w_0^{s_ja_j} - (1-\mathcal{P}_{s_ja_j}^{\mathcal{M}})\Big]\cdot\breve{Q}'_j(s_j,a_j)\Big| \\ &+ \Big|(1-w_0^{s_ja_j})\hat{\mathcal{R}}_{s_ja_j} - \sum_{s'\in\mathcal{M}} \mathcal{P}_{s_ja_j}^{s'}\mathcal{R}_{s_ja_j}^{s'}\Big| \\ &+ \gamma\Big|\sum_{s'\in\mathcal{M}}\Big[(1-w_0^{s_ja_j})\hat{\mathcal{P}}_{s_ja_j}^{s'} - \mathcal{P}_{s_ja_j}^{s'}\Big]\cdot\max_{a'} Q'_j(s',a')\Big|\,. \end{aligned}$$

The sum of the last three terms we call $d_j(s_j,a_j)$. By substituting (28), (29) and (30) in these three terms, it follows that $\lim_{j\to\infty} d_j(s_j,a_j) = 0$. We can further bound $|F'_j(s_j,a_j,\text{'bm'})|$ as follows:

$$\begin{aligned} |F'_j(s_j,a_j,\text{'bm'})| &\leq (1-\mathcal{P}_{s_ja_j}^{\mathcal{M}})\|\breve{Q}_j - \breve{Q}^*\| + \gamma\mathcal{P}_{s_ja_j}^{\mathcal{M}}\|Q_j - Q^*\| + d_j(s_j,a_j) \\ &\leq (1-\mathcal{P}_{s_ja_j}^{\mathcal{M}})\|\Delta'_j\| + \gamma\mathcal{P}_{s_ja_j}^{\mathcal{M}}\left(\|\Delta'_j\| + \|c_j\|\right) + d_j(s_j,a_j) \\ &\leq \Big((1-\mathcal{P}_{s_ja_j}^{\mathcal{M}}) + \gamma\mathcal{P}_{s_ja_j}^{\mathcal{M}}\Big)\cdot\|\Delta'_j\| + \gamma\mathcal{P}_{s_ja_j}^{\mathcal{M}}\|c'_j\| + d_j(s_j,a_j) \\ &= \kappa^{s_ja_j}\|\Delta'_j\| + \gamma\mathcal{P}_{s_ja_j}^{\mathcal{M}}\|c'_j\| + d_j(s_j,a_j)\,. \end{aligned} \tag{40}$$

Note $\|c'_j\|$, as well as $d_j(s_j,a_j)$, converge to 0. Note also that $\kappa^{s_ja_j} < 1$, since $\mathcal{P}_{s_ja_j}^{\mathcal{M}} > 0$ and $\gamma < 1$. From (39) and (40) it follows that the third condition of Lemma 17 is also satisfied. Hence, all conditions hold and Δ'_j converges to 0 w.p.1. Combining this with (38), proves Lemma 18.

■

11. There is no explicit sample related to a best-match update. For consistency, we define $r_j = \emptyset$ and $s'_j = \emptyset$ if $b_{j-1} = \text{'}bm'$.

E.3 Convergence of U_j to U'_j

Lemma 19 *$U_j(s,a,b)$ converges in the limit to $U'_j(s,a,b)$ w.p.1.*

Proof We define $\Delta(s,a,b) = U'_j(s,a,b) - U_j(s,a,b)$ and will prove that $\Delta(s,a,b)$ converges to 0 using Lemma 20. We exclude $(s,a,\text{'bm'})$ triples for which $\mathcal{P}^{\mathcal{M}}_{sa} = 0$ from the domain of Δ. Similar to the reasoning behind (38) and (37), we can deduce

$$Q_j(s,a) = \breve{Q}_j(s,a) + c_j(s,a)\,, \qquad \text{if } \mathcal{P}^{\mathcal{M}}_{sa} = 0\,,$$

with $c_j(s,a)$ converging to 0 in the limit, as well as

$$\lim_{j\to\infty}\Big(\breve{Q}'_j(s,a) - \breve{Q}_j(s,a)\Big) = 0 \quad \Rightarrow \quad \lim_{j\to\infty}\Big(Q'_j(s,a) - Q_j(s,a)\Big) = 0\,, \qquad \text{if } \mathcal{P}^{\mathcal{M}}_{sa} = 0\,. \tag{41}$$

Note, $\|\breve{Q}'_j - \breve{Q}_j\| \le \|\Delta_j\|$. However, $\|Q'_j - Q_j\| \le \|\Delta_j\|$ does not hold in general, because of the exclusion of (s,a,'bm') triples with $\mathcal{P}^{\mathcal{M}}_{sa} = 0$ from the domain of Δ_j. Instead, the following holds:

$$\begin{aligned}
\|Q'_j - Q_j\| &= \max(\|Q'_j - Q_j\|_{P^{\mathcal{M}}_{sa}>0}, \|Q'_j - Q_j\|_{P^{\mathcal{M}}_{sa}=0})\\
&\le \max(\|Q'_j - Q_j\|_{P^{\mathcal{M}}_{sa}>0}, \|\breve{Q}'_j - \breve{Q}_j\|_{P^{\mathcal{M}}_{sa}=0} + \|c_j\| + \|c'_j\|)\\
&\le \max(\|U'_j - U_j\|, \|U'_j - U^*\| + \|c_j\| + \|c'_j\|)\\
&= \|U'_j - U_j\| + \|c_j\| + \|c'_j\|\\
&= \|\Delta'_j\| + c''_j\,,
\end{aligned}$$

with $c''_j = \|c_j\| + \|c'_j\|$ converging to 0 w.p.1.

For $\mathcal{P}^{\mathcal{M}}_{sa} > 0$ we can rewrite (35) as

$$\begin{aligned}
Q'_{j+1}(s_j,a_j) &= (1-\beta^{s_ja_j})Q'_j(s_j,a_j) + \beta^{s_ja_j}Y'_j(s_j,a_j)\\
&= Y'_j(s_j,a_j) + (1-\beta^{s_ja_j})[Q'_j(s_j,a_j) - Y'_j(s_j,a_j)]\,.
\end{aligned}$$

In Section E.2 we proved that $\Delta'_j(s,a,\text{'bm'}) = Q'(s,a) - Q^*(s,a)_j$ converges to 0 w.p.1. On the other hand, it follows from (40), that $F'_j(s_j,a_j,\text{'bm'})$, which is equal to $Y'_j(s_j,a_j) - Q^*(s_j,a_j)$, also converges to 0 w.p.1. Therefore, both $Q'_j(s_j,a_j)$ and $Y'_j(s_j,a_j)$ converge to the same value, so we can write

$$Q'_{j+1}(s_j,a_j) = Y'_j(s_j,a_j) + e_j(s_j,a_j)\,, \qquad \text{if } \mathcal{P}^{\mathcal{M}}_{s_ja_j} > 0\,,$$

with $e_j(s_j,a_j)$ converging to 0 w.p.1.

Δ_j is updated according to

$$\Delta_{j+1}(s,a,b) = (1-\zeta_j(s,a,b))\Delta_j(s,a,b) + \zeta_j(s,a,b)F_j(s,a,b)\,.$$

For $(s,a,b) \ne (s_j,a_j,b_j)$, $\zeta_j(s,a,b) = 0$ and $F_j(s,a,b) = 0$. While for (s_j,a_j,b_j) the following holds:

$$\zeta_j(s_j,a_j,b_j) = \begin{cases} \alpha_j^{s_ja_j} & \text{if } b_j = \text{'mf'}\\ 1 & \text{if } b_j = \text{'bm'}\,,\end{cases}$$

and

$$F_j(s_j,a_j,b_j) = \begin{cases} \gamma \max_{a'} Q'_j(s'_{j+1},a') - \gamma \max_{a'} Q_j(s'_{j+1},a') & \text{if } b_j = \text{'mf'}\\ Y'_j(s_j,a_j) - Y_j(s_j,a_j,b_j) + e_j(s_j,a_j) & \text{if } b_j = \text{'bm'}\,.\end{cases}$$

We now check the three conditions of Lemma 20. Conditions 1 and 2 from the lemma follow from conditions 1 and 2 of Theorem 12. Condition 3, we prove below.

For b_j = 'mf', the following holds:

$$\begin{aligned} |F_j(s_j,a_j,\text{`mf'})| &= \gamma \left| \max_{a'} Q'_j(s'_{j+1},a') - \max_{a'} Q_j(s'_{j+1},a') \right| \\ &\leq \gamma \|Q'_j - Q_j\| \\ &\leq \gamma \|\Delta_j\| + \gamma c''_j, \end{aligned} \tag{42}$$

while for b_j = 'bm', we can write

$$\begin{aligned} |F_j(s_j,a_j,\text{`bm'})| &= |Y'_j(s_j,a_j) - Y_j(s_j,a_j) + e_j(s_j,a_j,b_j)| \\ &\leq w_0^{s_j,a_j} |\breve{Q}'_j(s_j,a_j) - \breve{Q}_j(s_j,a_j)| + |e_j(s_j,a_j)| + \\ &\qquad \gamma (1 - w_0^{s_j a_j}) \sum_{s'} \hat{\mathcal{P}}_{s_j a_j}^{s'} \left| \max_{a'} Q'_j(s',a') - \max_{a'} Q_j(s',a') \right| \\ &\leq w_0^{s_j,a_j} \|\Delta_j\| + \gamma(1 - w_0^{s_j a_j}) \|\Delta_j\| + |e_j(s_j,a_j)| + \gamma(1 - w_0^{s_j a_j}) c''_j \\ &= \left((1 - \mathcal{P}^{\mathcal{M}}_{s_j a_j}) + \gamma \mathcal{P}^{\mathcal{M}}_{s_j a_j} \right) \|\Delta_j\| + |e_j(s_j,a_j)| + \gamma (1 - w_0^{s_j a_j}) c''_j + \\ &\qquad \left(w_0^{s_j,a_j} + \gamma(1 - w_0^{s_j a_j}) - (1 - \mathcal{P}^{\mathcal{M}}_{s_j a_j}) - \gamma \mathcal{P}^{\mathcal{M}}_{s_j a_j} \right) \|\Delta_j\|. \end{aligned}$$

We define

$$\begin{aligned} f_j(s_j,a_j) &= \left(w_0^{s_j,a_j} + \gamma(1 - w_0^{s_j a_j}) - (1 - \mathcal{P}^{\mathcal{M}}_{s_j a_j}) - \gamma \mathcal{P}^{\mathcal{M}}_{s_j a_j} \right) \|\Delta_j\| \\ &\qquad + |e_j(s_j,a_j)| + \gamma(1 - w_0^{s_j a_j}) c''_j. \end{aligned}$$

Note that $\lim_{j\to\infty} f_j = 0$, since e_j and c''_j converge to 0 and $w_0^{s_j,a_j}$ converges to $1 - \mathcal{P}^{\mathcal{M}}_{s_j a_j}$. Using this definition and (36), we can write

$$|F_j(s_j,a_j,\text{`bm'})| \leq \kappa^{s_j a_j} \|\Delta_j\| + f_j(s_j,a_j). \tag{43}$$

Note that $\kappa^{s_j a_j} < 1$. From (42) and (43) it follows that the third condition of Lemma 20 is also satisfied. Hence, all conditions hold and Δ_j converges to 0 w.p.1.Combining this with (41), proves Lemma 19. ■

E.4 Proof of Theorem 12

Because U'_j converges to U^* (Lemma 18) and U_j converges to U'_j (Lemma 19), it follows that also U_j converges to U^*. From this it follows that Q converges to Q^*, proving Theorem 12.

Appendix F. Lemma 20

Lemma 20 *Consider a stochastic process* $(\alpha_t, \Delta_t, F_t)$, $t \geq 0$, *where* $\alpha_t, \Delta_t, F_t : X \to \mathbb{R}$ *satisfy the equations:*

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\Delta_t(x) + \alpha_t(x) F_t(x),$$

where $x \in X$ *and* $t = 0, 1, 2, \ldots$. *Assume that the following conditions hold:*

1. *The set X is finite.*

2. $\alpha_t(x) = [0,1]$, $\sum_t \alpha_t(x) = \infty$.

3. $\|F_t\| \leq \kappa \|\Delta_t\| + c_t$, *where* $\kappa \in [0,1)$ *and* c_t *converges to zero w.p. 1*,

where $\|\cdot\|$ *denotes a maximum norm. Then* Δ_t *converges to zero with probability one.*

Note that this lemma is similar to Lemma 17, but the conditions for the learning rates are less strict ($\sum_t(\alpha_t(x_t))^2 < \infty$ is missing), while the condition for F_t is more strict (condition 3 uses the value of F_t instead of its expected value).

Proof The outline of this proof is that we define a related process Δ'_t that converges to 0 and show that $\|\Delta_t\| \leq \|\Delta'_t\|$ for all t. We will ignore c_t in this proof. This can be safely done, since c_t converges to zero, $\kappa < 1$ and $\sum_t \alpha_t(x) = \infty$ for all x. Therefore, this term is asymptotically unimportant.

We define $\Delta'_0(x) = \|\Delta_0\|$ for all x. For $t > 0$, $\Delta'_t(x)$ is defined as

$$\Delta'_{t+1}(x) = (1 - \beta_t(x))\Delta'_t(x) + \beta_t(x)\kappa\|\Delta'_t\|, \tag{44}$$

with $\beta_t(x) \leq \alpha_t(x)$ and $\beta_t(x) \in [0,1]$, $\sum_t \beta_t(x) = \infty$, $\sum_t(\beta_t(x))^2 < \infty$ w.p.1. It follows from (44) that $\|\Delta'_{t+1}\| \leq \|\Delta'_t\|$. It also follows that if $\Delta'_t(x) \geq \kappa\|\Delta'_t\|$ then $\Delta'_{t+1}(x) \geq \kappa\|\Delta'_t\| \geq \kappa\|\Delta'_{t+1}\|$. And since $\Delta'_0(x) \geq \kappa\|\Delta'_0\|$ it follows that

$$\Delta'_t(x) \geq \kappa\|\Delta'_t\|, \qquad \text{for all } t. \tag{45}$$

Using Lemma 17, it can easily be shown that Δ' converges in the limit to 0 w.p.1.

We now prove that $\|\Delta_t\| \leq \|\Delta'_t\|$ for all t. We start by proving

$$|\Delta_t(x)| \leq \Delta'_t(x) \quad \text{for all } x \quad \Rightarrow \quad |\Delta_{t+1}(x)| \leq \Delta'_{t+1}(x) \quad \text{for all } x. \tag{46}$$

Assuming the left part of (46), for $|\Delta_{t+1}(x)|$ the following holds:

$$\begin{aligned} |\Delta_{t+1}(x)| &\leq (1 - \alpha_t(x))|\Delta_t(x)| + \alpha_t(x)\kappa\|\Delta_t\| \\ &\leq (1 - \alpha_t(x))\Delta'_t(x) + \alpha_t(x)\kappa\|\Delta'_t\|. \end{aligned}$$

Since (45) and $\beta_t(x) \leq \alpha_t(x)$, we can continue as

$$\begin{aligned} |\Delta_{t+1}(x)| &\leq (1 - \beta_t(x))\Delta'_t(x) + \beta_t(x)\kappa\|\Delta'_t\| \\ &\leq \Delta'_{t+1}(x). \end{aligned}$$

This proves (46). And since $|\Delta_0(x)| \leq \Delta'_0(x)$, it follows that $|\Delta_t(x)| \leq \Delta'_t(x)$ holds for all t, and hence, $\|\Delta_t\| \leq \|\Delta'_t\|$ proving the lemma. ■

References

C.G. Atkeson, A.W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1):11–73, 1997.

R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ., 1957.

J. Boyan and A.W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*, 1995.

R.I. Brafman and M. Tennenholtz. R-max: A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

C. Diuk, L. Li, and B.R. Leffler. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(1):503–556, 2005.

T. Jaakkola, M.I. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

L.P. Kaelbling, M.L. Littman, and A.P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

M. Kearns and S. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. *Advances in Neural Information Processing Systems*, 11:996–1002, 1999. ISSN 1049-5258.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232, 2002.

M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1149, 2003.

L.J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3):293–321, 1992.

A. Moore and C. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

M. L. Puterman and M. C. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24:1127–1137, 1978.

G.A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, Tech. rep. CUED/F-INENG/TR166, Cambridge University, 1994.

S. Singh, T. Jaakkola, M.L. Littman, and C. Szepesvari. Convergence results for single-step on-policy reinforecement-learning algorithms. *Machine Learning*, 38:287–308, 2000.

A.L. Strehl and M.L. Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the 22th International Conference on Machine Learning*, pages 856–863, 2005.

A.L. Strehl, L. Li, E. Wiewiora, J. Langford, and M.L. Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 881–888, 2006.

R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1): 9–44, 1988.

R.S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, pages 216–224, 1990.

R.S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1045, 1996.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachussets, 1998.

R.S. Sutton and S.P. Singh. On step-size and bias in temporal-difference learning. In *Proceedings of the 8th Yale Workshop on Adaptive and Learning Systems*, 1994.

C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.

C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):9–44, 1992.

M. Wiering and J. Schmidhuber. Fast online Q(λ). *Machine Learning*, 33:105–115, 1998.

Journal of Machine Learning Research 12 (2011) 2095-2119 Submitted 12/10; Revised 5/11; Published 6/11

Information Rates of Nonparametric Gaussian Process Methods

Aad van der Vaart AAD@FEW.VU.NL
Department of Mathematics
VU University Amsterdam
De Boelelaan 1081
1081 HV Amsterdam
The Netherlands

Harry van Zanten J.H.V.ZANTEN@TUE.NL
Department of Mathematics
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands

Editor: Manfred Opper

Abstract

We consider the quality of learning a response function by a nonparametric Bayesian approach using a Gaussian process (GP) prior on the response function. We upper bound the quadratic risk of the learning procedure, which in turn is an upper bound on the Kullback-Leibler information between the predictive and true data distribution. The upper bound is expressed in small ball probabilities and concentration measures of the GP prior. We illustrate the computation of the upper bound for the Matérn and squared exponential kernels. For these priors the risk, and hence the information criterion, tends to zero for all continuous response functions. However, the rate at which this happens depends on the combination of true response function and Gaussian prior, and is expressible in a certain concentration function. In particular, the results show that for good performance, the regularity of the GP prior should match the regularity of the unknown response function.

Keywords: Bayesian learning, Gaussian prior, information rate, risk, Matérn kernel, squared exponential kernel

1. Introduction

In this introductory section we first recall some important concepts from Gaussian process regression and then outline our main contributions.

1.1 Gaussian Process Regression

Gaussian processes (GP's) have become popular tools for making inference about unknown functions. They are widely used as prior distributions in nonparametric Bayesian learning to predict a response $Y \in \mathcal{Y}$ from a covariate $X \in \mathcal{X}$. In this approach (cf. Rasmussen and Williams, 2006) a response function $f\colon \mathcal{X} \to \mathcal{Y}$ is "a-priori" modelled by the sample path of a Gaussian process. This means that for every finite set of points x_j in $\mathcal{X}$, the prior distribution of the vector $(f(x_1), \dots, f(x_n))$

is multivariate Gaussian. As Gaussian distributions are completely parameterized by their mean and covariance matrix, a GP is completely determined by its *mean function* $m: \mathcal{X} \to \mathbb{R}$ and *covariance kernel* $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, defined as

$$m(x) = \mathrm{E} f(x), \qquad K(x_1, x_2) = \mathrm{cov}\big(f(x_1), f(x_2)\big).$$

The mean function can be any function; the covariance function can be any symmetric, positive semi-definite function. Popular choices are the squared-exponential and Matérn kernels (see Rasmussen and Williams, 2006), or (multiply) integrated Brownian motions (e.g., Wahba, 1978; Van der Vaart and Van Zanten, 2008a). The first two choices are examples of *stationary* GP: the corresponding covariance function has the form $K(x_1, x_2) = K_0(x_1 - x_2)$, for some function K_0 of one argument and hence the distribution of the random function $x \mapsto f(x)$ remains the same under shifting its argument. By Bochner's theorem the stationary covariance functions on $\mathcal{X} = \mathbb{R}^d$ correspond one-to-one to *spectral distributions* (see below for the examples of the squared-exponential and Matérn kernels, or see Rasmussen and Williams, 2006).

In Gaussian process learning the regression function f is modeled as a GP and conditionally on f, observed training data $(X_1, Y_1), \ldots, (X_n, Y_n)$ are viewed as independent pairs that satisfy $Y_i = f(X_i) + \varepsilon_i$, for noise variables ε_i. If g denotes the marginal density of the covariates X_i and for $\mu \in \mathbb{R}$, p_μ denotes the density of $\mu + \varepsilon_i$, then conditional on the GP f the pairs (X_i, Y_i) are independently generated according to the probability density $(x, y) \mapsto p_{f(x)}(y) g(x)$. If the errors are normal with mean 0 and variance σ^2 for instance, we have $p_\mu(y) = (2\pi\sigma^2)^{-1/2} \exp(-(y-\mu)^2/(2\sigma^2))$. By Bayes' rule, the posterior distribution for f given the training data is then given by

$$d\Pi_n(f | X_{1:n}, Y_{1:n}) \propto \prod_{i=1}^n p_{f(X_i)}(Y_i)\, d\Pi(f),$$

where $d\Pi(f)$ refers to the prior distribution, and $Z_{1:n}$ is short for the sequence $Z_1, \ldots, Z_n$. After computation (see for instance Rasmussen and Williams, 2006 for methodology), the posterior distribution may be used to predict new responses from covariate values.

1.2 Quantifying Performance

A common approach to assessing the performance of nonparametric Bayes methods is to assume that the data are in actual fact generated according to a fixed, "true" regression function f_0 and to study how well the posterior distribution, which is a distribution over functions, approximates the target f_0 as the number of training data n tends to infinity.

The distance of the posterior to the truth can be measured in various ways. Seeger et al. (2008) discussed the performance of this method in terms of an information criterion due to Barron (1999). They consider the quantity

$$\mathrm{E}_{f_0} \frac{1}{n} \sum_{i=1}^n KL\Big(p_{f_0(X_i)}, \int p_{f(X_i)}\, d\Pi_{i-1}(f | X_{1:i-1}, Y_{1:i-1})\Big). \tag{1}$$

Here $KL(p, q) = \int \log(p/q)\, dP$ denotes the Kullback-Leibler divergence between two probability densities p and q, so that the terms of the sum are the Kullback-Leibler divergences between the density $y \mapsto p_{f_0(X_i)}(y)$ and the *Bayesian predictive density* $y \mapsto \int p_{f(X_i)}(y)\, d\Pi_{i-1}(f | X_{1:(i-1)}, Y_{1:i-1})$ based on the first $(i-1)$ observations, both evaluated for fixed covariate X_i. The expectation E_{f_0}

on the far left is relative to the distribution of $(X_1,Y_1),\dots,(X_n,Y_n)$. Seeger et al. (2008) obtain a bound on the information criterion (1), which allows them to show for several combinations of true regression functions f_0 and GP priors Π that this tends to zero at a certain rate in the number of observations n.

The information criterion (1) is the Cesàro average of the sequence of *prediction errors*, for $n = 1,2,\dots,$

$$\mathrm{E}_{f_0} KL\Big(p_{f_0(X_{n+1})}, \int p_{f(X_{n+1})}\, d\Pi_n(f|X_{1:n},Y_{1:n})\Big).$$

By concavity of the logarithm and Jensen's inequality (or the convexity of KL in its second argument), these are bounded above by the *risks*

$$\mathrm{E}_{f_0} \int KL\big(p_{f_0(X_{n+1})}, p_{f(X_{n+1})}\big)\, d\Pi_n(f|X_{1:n},Y_{1:n}). \tag{2}$$

The KL divergence between two normal densities with means μ_1 and μ_2 and common variance σ^2 is equal to $(\mu_1-\mu_2)^2/(2\sigma^2)$. Therefore, in the case of normal errors, with p_f the density of the normal distribution with mean f and variance σ^2, the risks reduce to

$$\frac{1}{2\sigma^2}\mathrm{E}_{f_0} \int \|f_0 - f\|_2^2\, d\Pi_n(f|X_{1:n},Y_{1:n}), \tag{3}$$

where $\|\cdot\|_2$ is the L_2-norm relative to the distribution of the covariate X_{n+1}, that is, $\|f\|_2^2 = \int f^2(x)g(x)\,dx$, and σ^2 is the error variance.

Barron (1999) suggested to use the information criterion (1) as a discrepancy measure, because the risks (2) sometimes behave erratically. However, the risks measure the concentration of the full posterior (both location and spread) near the truth, whereas the prediction errors concern the location of the posterior only. Furthermore, taking Cesàro averages may blur discrepancies in the individual prediction errors. We will show that the present situation is in fact *not* one where the risk (2) behaves badly, and this bigger quantity can be bounded instead of the information criterion (1).

If the risk (3) is bounded by ε_n^2 for some sequence $\varepsilon_n \to 0$, then by another application of Jensen's inequality the posterior mean $\mathrm{E}(f|X_{1:n},Y_{1:n}) = \int f\, d\Pi_n(f|X_{1:n},Y_{1:n})$ satisfies

$$\mathrm{E}_{f_0}\big\|\mathrm{E}(f|X_{1:n},Y_{1:n}) - f_0\big\|_2^2 \le \varepsilon_n^2. \tag{4}$$

Thus the posterior distribution induces a "point estimator" that approximates f_0 at the rate same ε_n. It follows that a bound ε_n^2 on the posterior risk (3) must satisfy the same fundamental lower bound as the (quadratic) risk of general nonparametric estimators for the regression function f_0. Such bounds are usually formulated as *minimax* results: for a given point estimator (for example the posterior mean) one takes the maximum (quadratic) risk over all f_0 in a given "a-priori class" of response functions, and shows that this cannot be smaller than some lower bound (see, e.g., Tsybakov, 2009 for a general introduction to this approach). Typical a-priori classes in nonparametric learning are spaces of "smooth" functions. Several variations exist in the precise definition of such spaces, but they have in common a positive parameter β, which measures the extent of the smoothness or "regularity"; this is roughly the number of times that the functions f_0 are differentiable. It is known that if f_0 is defined on a compact subset of $\mathbb{R}^d$ and has regularity $\beta > 0$, then the optimal, minimax rate ε_n is given by (see, e.g., Tsybakov, 2009)

$$\varepsilon_n = n^{-\beta/(2\beta+d)}. \tag{5}$$

It follows that this is also the best possible bound for the risk (3) if f_0 is a β-regular function of d variables. Recent findings in the statistics literature show that for GP priors, it is typically true that this optimal rate can only be attained if the regularity of the GP that is used matches the regularity of f_0 (see Van der Vaart and Van Zanten, 2008a). Using a GP prior that is too rough or too smooth deteriorates the performance of the procedure. Plain consistency however, that is, the existence of *some* sequence ε_n for which (4) holds, typically obtains for *any* f_0 in the support in the prior.

Seeger et al. (2008) considered the asymptotic performance for the Matérn and squared exponential GP priors, but we will argue in the next subsection that using their approach it is not possible to exhibit the interesting facts that optimal rates are obtained by matching regularities and that consistency holds for any f_0 in the support of the prior. In this paper we will derive these results by following a different approach, along the lines of Ghosal et al. (2000) and Van der Vaart and Van Zanten (2008a).

1.3 Role of the RKHS

A key issue is the fact that Seeger et al. (2008) require the true response function f_0 to be in the reproducing kernel Hilbert space (RKHS) of the GP prior. The RKHS of a GP prior with zero mean function and with covariance kernel K can be constructed by first defining the space $\mathbb{H}_0$ consisting of all functions of the form $x \mapsto \sum_{j=1}^{k} c_i K(x, y_i)$. Next, the inner product between two functions in $\mathbb{H}_0$ is defined by

$$\Big\langle \sum c_i K(\cdot, y_i), \sum c'_j K(\cdot, y'_j) \Big\rangle_{\mathbb{H}} = \sum \sum c_i c'_j K(y_i, y'_j),$$

and the associated RKHS-norm by $\|h\|_{\mathbb{H}}^2 = \langle h, h \rangle_{\mathbb{H}}$. Finally, the RKHS $\mathbb{H}$ is defined as the closure of $\mathbb{H}_0$ relative to this norm. Since for all $h \in \mathbb{H}_0$ we have the *reproducing formula*

$$h(x) = \langle h, K(x, \cdot) \rangle_{\mathbb{H}},$$

the RKHS is (or, more precisely, can be identified with) a space of functions on $\mathcal{X}$ and the reproducing formula holds in fact for all $h \in \mathbb{H}$. (For more details, see, e.g., the paper Van der Vaart and Van Zanten, 2008b, which reviews theory on RKHSs that is relevant for Bayesian learning.)

The assumption that $f_0 \in \mathbb{H}$ is very limiting in most cases. The point is that unless the GP prior is a finite-dimensional Gaussian, the RKHS is very small relative to the support of the prior. In the infinite-dimensional case that we are considering here the probability that a draw f from the prior belongs to $\mathbb{H}$ is 0. The reason is that typically, the elements of $\mathbb{H}$ are "smoother" than the draws from the prior. On the other hand, the probability of a draw f falling in a neighbourhood of a given continuous f_0 is typically positive, no matter how small the neighbourhood. (A neighbourhood of f_0 could for instance be defined by all functions with $|f(x) - f_0(x)| < \varepsilon$ for all x, and a given $\varepsilon > 0$.) This means that prior draws can approximate any given continuous function arbitrarily closely, suggesting that the posterior distribution should be able to learn any such function f_0, not just the functions in the RKHS.

Example 1 (Integrated Brownian motion and Matérn kernels) *It is well known that the sample paths $x \mapsto f(x)$ of Brownian motion f have regularity $1/2$. More precisely, for all $\alpha \in (0, 1/2)$ they are almost surely Hölder continuous with exponent α: $\sup_{0 \le x < y \le 1} |f(x) - f(y)|/|x - y|^{\alpha}$ is finite or infinite with probability one depending on whether $\alpha < 1/2$ or $\alpha \ge 1/2$ (see, e.g., Karatzas and Shreve, 1991). Another classical fact is that the RKHS of Brownian motion is the so-called Cameron-Martin space, which consists of functions that have a square integrable derivative (see,* *e.g., Lifshits, 1995). Hence, the functions in the RKHS have regularity* 1*. More generally, it can be shown that draws from a k times integrated Brownian motion have regularity* $k+1/2$*, while elements from its RKHS have regularity* $k+1$ *(cf., e.g., Van der Vaart and Van Zanten, 2008b). Analogous statements hold for the Matérn kernel, see Section 3.1 ahead. All these priors can approximate a continuous function* f_0 *arbitrarily closely on any compact domain: the probability that* $\sup_x |f(x) - f_0(x)| < \varepsilon$ *is positive for any* $\varepsilon > 0$.

We show in this paper that if the true response function f_0 on a compact $\mathcal{X} \subset \mathbb{R}^d$ has regularity β, then for the Matérn kernel with smoothness parameter α the (square) risk (3) decays at the rate $n^{-2\min(\alpha,\beta)/(2\alpha+d)}$. This rate is identical to the optimal rate (5) if and only if $\alpha = \beta$. Because the RKHS of the Matérn (α) prior consists of functions of regularity $\alpha + 1/2$, it contains functions of regularity β only if $\beta \geq \alpha + 1/2$, and this excludes the case $\alpha = \beta$ that the Matérn prior is optimal. Thus if it is assumed a-priori that f_0 is contained in the RKHS, then optimality of Bayesian learning can never be established.

A second drawback of the assumption that $f_0 \in \mathbb{H}$ is that *consistency* (asymptotically correct learning at *some* rate) can be obtained only for a very small class of functions, relative to the support of the GP prior. For instance, Bayesian learning with a Matérn (α) prior is consistent for any continuous true function f_0, not only for f_0 of regularity $\alpha + 1/2$ or higher. For the square-exponential process restricting to $f_0 \in \mathbb{H}$ is even more misleading.

Example 2 (Squared exponential kernel) *For the squared exponential GP on a compact subset of* $\mathbb{R}^d$*, every function h in the RKHS has a Fourier transform* $\hat{h}$ *that satisfies*

$$\int |\hat{h}(\lambda)|^2 e^{c\|\lambda\|^2}\, d\lambda < \infty$$

for some $c > 0$ *(see Van der Vaart and Van Zanten, 2009 and Section 3.2 ahead). In particular, every* $h \in \mathbb{H}$ *can be extended to an analytic (i.e., infinitely often differentiable) function on* $\mathbb{C}^d$.

Hence for the squared exponential kernel, restricting to $f_0 \in \mathbb{H}$ only proves consistency for certain analytic regression functions. However, the support of the process is equal to the space of all continuous functions, and consistency pertains for every continuous regression function f_0.

A third drawback of the restriction to $f_0 \in \mathbb{H}$ is that this is the best possible case for the prior, thus giving an inflated idea of its performance. For instance, the squared exponential process gives very fast learning rates for response functions in its RKHS, but as this is a tiny set of analytic functions, this gives a misleading idea of its performance in genuinely nonparametric situations.

1.4 Contributions

In this paper we present a number of contributions to the study of the performance of GP methods for regression.

Firstly, our results give bounds for the risk (2) instead of the information criterion (1). As argued in Section 1.2 the resulting bounds are stronger.

Secondly, our results are not just valid for functions f_0 in the RKHS of the GP prior, but for all functions in the support of the prior. As explained in the preceding section, this is a crucial difference. It shows that in GP regression we typically have plain consistency for all f_0 in the support of the prior and it allows us to study how the performance depends on the relation between the regularities of the regression function f_0 and typical draws from the prior. We illustrate the general results for the Matérn and squared exponential priors. We present new rate-optimality results for these priors.

A third contribution is that although the concrete GP examples that we consider (Matérn and squared exponential) are stationary, our general results are not limited to stationary processes. The results of Seeger et al. (2008) do concern stationary process and use eigenvalue expansions of the covariance kernels. Underlying our approach are the so-called small deviations behaviour of the Gaussian prior and entropy calculations, following the same basic approach as in our earlier work (Van der Vaart and Van Zanten, 2008a). This allows more flexibility than eigenvalue expansions, which are rarely available and dependent on the covariate distribution. In our approach both stationary and nonstationary prior processes can be considered and it is not necessary to assume a particular relationship between the distribution of the covariates and the prior.

Last but not least, the particular cases of the Matérn and squared exponential kernels that we investigate illustrate that the performance of Bayesian learning methods using GP priors is very sensitive to the fine properties of the priors used. In particular, the relation between the regularity of the response function and the GP used is crucial. Optimal performance is only guaranteed if the regularity of the prior matches the regularity of the unknown function of interest. Serious mismatch leads to (very) slow learning rates. For instance, we show that using the squared-exponential prior, in a situation where a Matérn prior would be appropriate, slows the learning rate from polynomial to logarithmic in n.

1.5 Notations and Definitions

In this section we introduce notation that is used throughout the paper.

1.5.1 SPACES OF SMOOTH FUNCTIONS

As noted in Section 1.2 it is typical to quantify the performance of nonparametric learning procedures relative to a-priori models of smooth functions. The proper definition of "smoothness" or "regularity" depends on the specific situation, but roughly speaking, saying that a function has regularity α means it has α derivatives. In this paper we use two classical notions of finite smoothness: Hölder and Sobolev regularity; and also a scale of infinite smoothness.

For $\alpha > 0$, write $\alpha = m + \eta$, for $\eta \in (0,1]$ and m a nonnegative integer. The *Hölder space* $C^\alpha[0,1]^d$ is the space of all functions whose partial derivatives of orders $(k_1,\ldots,k_d)$ exist for all nonnegative integers $k_1,\ldots,k_d$ such that $k_1+\ldots+k_d \leq m$ and for which the highest order partial derivatives are Lipshitz functions of order η. (A function f is *Lipschitz* of order η if $|f(x)-f(y)| \leq C|x-y|^\eta$, for every x,y; see for instance Van der Vaart and Wellner (1996), Section 2.7.1, for further details on Hölder classes.)

The *Sobolev space* $H^\alpha[0,1]^d$ is the set of functions $f_0\colon [0,1]^d \to \mathbb{R}$ that are restrictions of a function $f_0\colon \mathbb{R}^d \to \mathbb{R}$ with Fourier transform $\hat{f}_0(\lambda) = (2\pi)^{-d}\int e^{i\lambda^T t} f(t)\,dt$ such that

$$\|f_0\|_{\alpha|2}^2 := \int \bigl(1+\|\lambda\|^2\bigr)^\alpha \bigl|\hat{f}_0(\lambda)\bigr|^2\, d\lambda < \infty.$$

Roughly speaking, for integer α, a function belongs to H^α if it has partial derivatives up to order α that are all square integrable. This follows, because the αth derivative of a function f_0 has Fourier transform $\lambda \mapsto (i\lambda)^\alpha \hat{f}_0(\lambda)$, Qualitatively both spaces $H^\alpha[0,1]^d$ and $C^\alpha[0,1]^d$ describe "α-regular" functions. Technically their definitions are different, and so are the resulting sets. There are however many functions in the intersection $H^\alpha[0,1]^d \cap C^\alpha[0,1]^d$ and these are α-regular in both senses at the same time.

We also consider functions that are "infinitely smooth". For $r \geq 1$ and $\lambda > 0$, we define the space $\mathcal{A}^{\gamma,r}(\mathbb{R}^d)$ of functions $f_0 \colon \mathbb{R}^d \to \mathbb{R}$ with Fourier transform $\hat{f}_0$ satisfying

$$\|f_0\|_{\mathcal{A}}^2 := \int e^{\gamma\|\lambda\|^r} |\hat{f}_0|^2(\lambda)\, d\lambda < \infty.$$

This requires exponential decrease of the Fourier transform, in contrast to polynomial decrease for Sobolev smooothness. The functions in $\mathcal{A}^{\gamma,r}(\mathbb{R}^d)$ are infinitely often differentiable and "increasingly smooth" as γ or r increase. They extend to functions that are analytic on a strip in $\mathbb{C}^d$ containing $\mathbb{R}^d$ if $r = 1$, and to entire functions if $r > 1$ (see, e.g., Bauer, 2001, 8.3.5).

1.5.2 General Function Spaces and Norms

For a general metric space $\mathcal{X}$ we denote by $C_b(\mathcal{X})$ the space of bounded, continuous functions on $\mathcal{X}$. If the space $\mathcal{X}$ is compact, for example, $\mathcal{X} = [0,1]^d$, we simply write $C(\mathcal{X})$. The supremum norm of a bounded function f on $\mathcal{X}$ is denoted by $\|f\|_\infty = \sup_{x\in\mathcal{X}} |f(x)|$.

For $x_1, \ldots, x_n \in \mathcal{X}$ and a function $f \colon \mathcal{X} \to \mathbb{R}$ we define the empirical norm $\|f\|_n$ by

$$\|f\|_n = \Big(\frac{1}{n}\sum_{i=1}^n f^2(x_i)\Big)^{1/2}. \tag{6}$$

For m a (Borel) measure on $A \subset \mathbb{R}^d$ we denote by $L_2(m)$ the associated L^2-space, defined by

$$L_2(m) = \Big\{f \colon A \to \mathbb{R} \,\Big|\, \int_A |f(x)|^2\, dm(x) < \infty\Big\}.$$

In a regression setting where the covariates have probability density g on $\mathbb{R}^d$, we denote the corresponding L_2-norm simply by $\|f\|_2$, that is,

$$\|f\|_2 = \int f^2(x) g(x)\, dx.$$

1.5.3 Miscellaneous

The notation $a \lesssim b$ means that $a \leq Cb$ for a universal constant C. We write $a \vee b = \max\{a, b\}$, $a \wedge b = \min\{a, b\}$.

2. General Results

In this section we present general bounds on the posterior risk. The next section treats the special cases of the Matérn and squared exponential kernels. Proofs are deferred to Section 4.

2.1 Fixed Design

In this section we assume that given the function $f \colon \mathcal{X} \to \mathbb{R}$, the data $Y_1, \ldots, Y_n$ are independently generated according to $Y_j = f(x_j) + \varepsilon_j$, for fixed $x_j \in \mathcal{X}$ and independent $\varepsilon_j \sim N(0, \sigma^2)$. Such a fixed design setting occurs when the covariate values in the training data have been set by an experimenter.

For simplicity we assume that $\mathcal{X}$ is a compact metric space, such as a bounded, closed set in $\mathbb{R}^d$, and assume that the true response function f_0 and the support of the GP prior are included in the space $C_b(\mathcal{X})$ of bounded, continuous functions on the metric space $\mathcal{X}$. This enables to formulate the conditions in terms of the *supremum norm* (also called "uniform" norm). Recall that the supremum norm of $f \in C_b(\mathcal{X})$ is given by $\|f\|_\infty = \sup_{x\in\mathcal{X}} |f(x)|$. (Actually Theorem 1 refers to the functions on the design points only and is in terms of the norm (6). The conditions could be formulated in terms of this norm. This would give a stronger result, but its interpretation is hampered by the fact that the norm (6) changes with n.) The RKHS of the GP prior, as defined in Section 1.3, is denoted by $\mathbb{H}$ and the RKHS-norm by $\|\cdot\|_{\mathbb{H}}$.

The following theorem gives an upper bound for the posterior risk. The bound depends on the "true" response function f_0 and the GP prior Π and its RKHS $\mathbb{H}$ through the so-called *concentration function*

$$\phi_{f_0}(\varepsilon) = \inf_{h\in\mathbb{H}:\|h-f_0\|_\infty<\varepsilon} \|h\|_{\mathbb{H}}^2 - \log\Pi\big(f\colon \|f\|_\infty < \varepsilon\big) \tag{7}$$

and the associated function

$$\psi_{f_0}(\varepsilon) = \frac{\phi_{f_0}(\varepsilon)}{\varepsilon^2}. \tag{8}$$

We denote by $\psi_{f_0}^{-1}$ the (generalized) inverse function of the function ψ_{f_0}, that is, $\psi_{f_0}^{-1}(l) = \sup\{\varepsilon > 0\colon \psi_{f_0}(\varepsilon) \geq l\}$.

The concentration function ϕ_{f_0} for a general response function consists of two parts. The second is the small ball exponent $\phi_0(\varepsilon) = -\log\Pi(f\colon \|f\|_\infty < \varepsilon)$, which measures the amount of prior mass in a ball of radius ε around the zero function. As the interest is in small ε this is (the exponent of) the *small ball probability* of the prior. There is a large literature on small ball probabilities of Gaussian distributions. (See Kuelbs and Li, 1993 and Li and Shao, 2001 and references.) This contains both general methods (probabilistic and analytic) for its computation and many examples, stationary and non-stationary. The first part of the definition of $\phi_{f_0}(\varepsilon)$, the infimum, measures the decrease in prior mass if the (small) ball is shifted from the origin to the true parameter f_0. This is not immediately clear from the definition (7), but it can be shown that up to constants, $\phi_{f_0}(\varepsilon)$ equals $-\log\Pi(f\colon \|f-f_0\|_\infty < \varepsilon)$ (see for instance Van der Vaart and Van Zanten, 2008b, Lemma 5.3). The infimum depends on how well f_0 can be approximated by elements h of the RKHS of the prior, and the quality of this approximation is measured by the size of the approximand h in the RKHS-norm. The infimum is finite for every $\varepsilon > 0$ if and only if f_0 is contained in the closure of $\mathbb{H}$ within $C_b(\mathcal{X})$. The latter closure is the support of the prior (Van der Vaart and Van Zanten, 2008b, Lemma 5.1) and in typical examples it is the full space $C_b(\mathcal{X})$.

Our general upper bound for the posterior risk in the fixed design case takes the following form.

Theorem 1 *For $f_0 \in C_b(\mathcal{X})$ it holds that*

$$\mathrm{E}_{f_0}\int \|f-f_0\|_n^2\, d\Pi_n\big(f|Y_{1:n}\big) \lesssim \psi_{f_0}^{-1}(n)^2.$$

For $\psi_{f_0}^{-1}(n) \to 0$ as $n \to \infty$, which is the typical situation, the theorem shows that the posterior distribution contracts at the rate $\psi_{f_0}^{-1}(n)$ around the true response function f_0. To connect to Seeger et al. (2008), we have expressed the contraction using the quadratic risk, but the concentration is actually exponential. In particular, the power 2 can be replaced by any finite power.

From the definitions one can show that (see Lemma 17), whenever $f_0 \in \mathbb{H}$,

$$\psi_{f_0}^{-1}(n) \lesssim \frac{\|f_0\|_{\mathbb{H}}}{\sqrt{n}} + \psi_0^{-1}(n). \tag{9}$$

This relates the theorem to formula (3) in Seeger et al., whose $\log\det(I+cK)$ is replaced by $\psi_0^{-1}(n)^2$. However, the left side $\psi_{f_0}^{-1}(n)$ of the preceding display is finite for every f_0 in the support of the prior, which is typically a much large space than the RKHS (see Section 1.3). For instance, functions f_0 in the RKHS of the squared exponential process are analytic, whereas $\psi_{f_0}^{-1}(n)$ is finite for every continuous function f_0 in that case. Thus the theorem as stated is much more refined than if its upper bound would be replaced by the right side of (9). It is true that $\psi_{f_0}^{-1}(n)$ is smallest if f_0 belongs to the RKHS, but typically the posterior also contracts if this is not the case.

In Sections 3.1 and 3.2 we show how to obtain bounds for the concentration function, and hence a risk bound, for two classes of specific priors: the Matérn class and the squared exponential. Other examples, including non-stationary ones like (multiply) integrated Brownian motion, were considered in Van der Vaart and Van Zanten (2008a), Van der Vaart and Van Zanten (2007) and Van der Vaart and Van Zanten (2009).

2.2 Random Design

In this section we assume that given the function $f\colon [0,1]^d \to \mathbb{R}$ on the d-dimensional unit cube $[0,1]^d$ (or another compact, Lipschitz domain in $\mathbb{R}^d$) the data $(X_1,Y_1),\ldots,(X_n,Y_n)$ are independently generated, X_i having a density g on $[0,1]^d$ that is bounded away from zero and infinity, and $Y_j = f(X_j)+\varepsilon_j$, for errors $\varepsilon_j \sim N(0,\sigma^2)$ that are independent given the X_i's.

We assume that under the GP prior Π the function f is a zero-mean, continuous Gaussian process. The concentration function ϕ_{f_0} and the derived function ψ_{f_0} are defined as before in (7) and (8). Recall that $\|f\|_2$ is the L_2-norm relative to the covariate distribution, that is, $\|f\|_2^2 = \int f^2(x)g(x)\,dx$. The theorem assumes that for some $\alpha > 0$, draws from the prior are α-regular in Hölder sense. This roughly means that α derivatives should exist. See Section 1.5 for the precise definition.

Theorem 2 *Suppose that for some $\alpha > 0$ the prior gives probability one to the Hölder space $C^\alpha[0,1]^d$. For $\psi_{f_0}^{-1}$ the inverse function of ψ_{f_0} and C a constant that depends on the prior and the covariate density, if $\psi_{f_0}^{-1}(n) \le n^{-d/(4\alpha+2d)}$, then*

$$\mathrm{E}_{f_0} \int \|f-f_0\|_2^2 \, d\Pi_n\big(f \mid X_{1:n}, Y_{1:n}\big) \le C\psi_{f_0}^{-1}(n)^2.$$

If, on the other hand, $\psi_{f_0}^{-1}(n) \ge n^{-d/(4\alpha+2d)}$, then the assertion is true with the upper bound $Cn\psi_{f_0}^{-1}(n)^{(4\alpha+4d)/d}$.

Unlike in the case of fixed design treated in Theorem 1, this theorem makes assumptions on the regularity of the prior. This seems unavoidable, because the $\|\cdot\|_2$-risk extrapolates from the observed design points to all points in the support of the covariate density.

In the next section we shall see that a typical rate for estimating a β-smooth response function f_0 is given by

$$\psi_{f_0}^{-1}(n) \sim n^{-(\beta\wedge\alpha)/(2\alpha+d)}.$$

(This reduces to the minimax rate $n^{-\alpha/(2\alpha+d)}$ if and only if $\alpha = \beta$.) In this case $\psi_{f_0}^{-1}(n) \leq n^{-d/(4\alpha+2d)}$ if and only if $\alpha \wedge \beta \geq d/2$. In other words, upper bounds for fixed and random design have exactly the same form if prior and true response are not too rough.

For very rough priors and true response functions, the rate given by the preceding theorem is slower than the rate for deterministic design, and for very rough response functions the theorem may not give a rate at all. The latter seems partly due to using the second moment of the posterior, rather than posterior concentration, although perhaps the theorem can be improved.

3. Results for Concrete Priors

In this section we specialize to two concrete classes of Gaussian process priors, the Matérn class and the squared exponential process.

3.1 Matérn Priors

In this section we compute the risk bounds given by Theorems 1 and 2 for the case of the Matérn kernel. In particular, we show that optimal rates are attained if the smoothness of the prior matches the smoothness of the unknown response function.

The *Matérn priors* correspond to the mean-zero Gaussian processes $W = (W_t : t \in [0,1]^d)$ with covariance function

$$\mathrm{E}W_sW_t = \int_{\mathbb{R}^d} e^{i\lambda^T(s-t)} m(\lambda)\, d\lambda,$$

defined through the *spectral densities* $m: \mathbb{R}^d \to \mathbb{R}$ given by, for $\alpha > 0$,

$$m(\lambda) = \frac{1}{\big(1 + \|\lambda\|^2\big)^{\alpha+d/2}}. \tag{10}$$

The integral can be expressed in certain special functions (see, e.g., Rasmussen and Williams, 2006). This is important for the numerical implementation of the resulting Bayesian procedure, but not useful for our present purpose.

The sample paths of the Matérn process possess the same smoothness in L_2 as the set of functions $e_t(\lambda) = e^{i\lambda^T t}$ in $L_2(m)$. From this it can be seen that the sample paths are k times differentiable in L_2, for k the biggest integer smaller than α, with kth derivative satisfying

$$\mathrm{E}(W_s^{(k)} - W_t^{(k)})^2 \lesssim \|s-t\|^{2(\alpha-k)}.$$

By Kolmogorov's continuity criterion it follows that the sample paths of the kth derivative can be constructed to be Lipshitz of any order strictly smaller than $\alpha - k$. Thus the Matérn process takes its values in $C^{\underline{\alpha}}[0,1]^d$ for any $\underline{\alpha} < \alpha$. Hence in this specific sense it is α-regular.

By Lemma 4.1 of Van der Vaart and Van Zanten (2009) the RKHS $\mathbb{H}$ of the process W is the space of all (real parts of) functions of the form

$$h_\psi(t) = \int e^{i\lambda^T t} \psi(\lambda) m(\lambda)\, d\lambda, \tag{11}$$

for $\psi \in L_2(m)$, and squared RKHS-norm given by

$$\|h_\psi\|_{\mathbb{H}}^2 = \min_{\phi: h_\phi = h_\psi} \int |\phi|^2(\lambda) m(\lambda)\, d\lambda. \tag{12}$$

This characterization is generic for stationary Gaussian processes. The minimum is unnecessary if the spectral density has exponential tails (as in the next section), but is necessary in the present case.

In the following two lemmas we describe the concentration function (7) of the Matérn prior. The small ball probability can be obtained from the preceding characterization of the RKHS, estimates of metric entropy, and general results on Gaussian processes. See Section 4.3 for proofs.

Lemma 3 *For $\|\cdot\|_\infty$ the uniform norm, and C a constant independent of ε,*

$$-\log \mathrm{P}\big(\|W\|_\infty < \varepsilon\big) \leq C\Big(\frac{1}{\varepsilon}\Big)^{d/\alpha}.$$

To estimate the infimum in the definition of the concentration function ϕ_{f_0} for a nonzero response function f_0, we approximate f_0 by elements of the RKHS. The idea is to write f_0 in terms of its Fourier inverse $\hat{f}_0$ as

$$\begin{aligned} f_0(x) &= \int e^{i\lambda^T x}\hat{f}_0(\lambda)\,d\lambda \\ &= \int e^{i\lambda^T x}\frac{\hat{f}_0}{m}(\lambda)\,m(\lambda)\,d\lambda. \end{aligned} \tag{13}$$

If $\hat{f}_0/m$ were contained in $L_2(m)$, then f_0 would be contained in the RKHS, with RKHS-norm bounded by the $L_2(m)$-norm of $\hat{f}_0/m$, that is, the square root of $\int(|\hat{f}_0|^2/m)(\lambda)\,d\lambda$. In general this integral may be infinite, but we can remedy this by truncating the tails of $\hat{f}_0/m$. We then obtain an approximation of f_0 by an element of the RKHS, which is enough to compute the concentration function (8).

A natural a-priori condition on the true response function $f_0\colon [0,1]^d \to \mathbb{R}$ is that this function is contained in a Sobolev space of order β. This space consists roughly of functions that possess β square integrable derivatives. The precise definition is given in Section 1.5.

Lemma 4 *If $f_0 \in C^\beta[0,1]^d \cap H^\beta[0,1]^d$ for $\beta \leq \alpha$, then, for $\varepsilon < 1$, and a constant C depending on f_0 and α,*

$$\inf_{h:\|h-f_0\|_\infty<\varepsilon} \|h\|_{\mathbb{H}}^2 \leq C\Big(\frac{1}{\varepsilon}\Big)^{(2\alpha+d-2\beta)/\beta}.$$

Combination of the two lemmas yields that for $f_0 \in C^\beta[0,1]^d \cap H^\beta[0,1]^d$ for $\beta \leq \alpha$, the concentration function (7) satisfies

$$\phi_{f_0}(\varepsilon) \lesssim \Big(\frac{1}{\varepsilon}\Big)^{(2\alpha+d-2\beta)/\beta} + \Big(\frac{1}{\varepsilon}\Big)^{d/\alpha}.$$

This implies that

$$\psi_{f_0}^{-1}(n) \lesssim \Big(\frac{1}{n}\Big)^{\beta/(2\alpha+d)}.$$

Theorems 1 and 2 imply that the rate of contraction of the posterior distribution is of this order in the case of fixed design, and of this order if $\beta > d/2$ in the case of random design. We summarize these findings in the following theorem.

Theorem 5 *Suppose that we use a Matérn prior with parameter $\alpha > 0$ and $f_0 \in C^\beta[0,1]^d \cap H^\beta[0,1]^d$ for $\beta > 0$. Then in the fixed design case the posterior contracts at the rate $n^{-(\alpha\wedge\beta)/(2\alpha+d)}$. In the random design case this holds as well, provided $\alpha \wedge \beta > d/2$.*

Observe that the optimal rate $n^{-\beta/(2\beta+d)}$ is attained if and only if $\alpha = \beta$. Using a prior that is "rougher" or "smoother" than the truth leads to sub-optimal rates. This is in accordance with the findings for other GP priors in in Van der Vaart and Van Zanten (2008a). It should be remarked here that Theorem 5 only gives an upper bound on the rate of contraction. However, the paper by Castillo (2008) shows that these bounds are typically tight.

3.2 Squared Exponential Kernel

In this section we compute the risk bounds given by Theorems 1 and 2 for the case of the squared exponential kernel.

The *squared exponential process* is the zero-mean Gaussian process with covariance function

$$\mathrm{E}W_sW_t = e^{-\|s-t\|^2}, \qquad s,t \in [0,1]^d.$$

Like the Matérn process the squared exponential process is stationary. Its spectral density is given by

$$m(\lambda) = \frac{1}{2^d \pi^{d/2}} e^{-\|\lambda\|^2/4}. \tag{14}$$

The sample paths of the square exponential process are analytic.

This process was studied already in Van der Vaart and Van Zanten (2007) and Van der Vaart and Van Zanten (2009). The first of the following lemmas is Lemma 4.5 in Van der Vaart and Van Zanten (2009). It deals with the second term in the concentration function (7). As before, let $\|\cdot\|_\infty$ be the uniform norm on the functions $f\colon [0,1]^d \to \mathbb{R}$.

Lemma 6 *There exists a constant C depending only on d such that*

$$-\log \mathrm{P}\Big(\|W\|_\infty \le \varepsilon\Big) \le C\Big(\log\frac{1}{\varepsilon}\Big)^{1+d}.$$

The following lemma concerns the infimum part of the concentration function in the case that the function f_0 belongs to a Sobolev space with regularity β (see Section 1.5).

Lemma 7 *If $f_0 \in H^\beta[0,1]^d$ for $\beta > d/2$, then, for a constant C that depends only on f_0,*

$$\inf_{\|h-f_0\|_\infty \le \varepsilon} \|h\|_{\mathbb{H}}^2 \le \exp\big(C\varepsilon^{-2/(\beta-d/2)}\big).$$

Combination of the preceding two lemmas shows that for a β-regular response function f_0 (in Sobolev sense)

$$\phi_{f_0}(\varepsilon) \lesssim \exp\big(C\varepsilon^{-2/(\beta-d/2)}\big) + \Big(\log\frac{1}{\varepsilon}\Big)^{1+d}.$$

The first term on the right dominates, for any $\beta > 0$. The corresponding rate of contraction satisfies

$$\psi_{f_0}^{-1}(n) \lesssim (1/\log n)^{\beta/2-d/4}.$$

Thus the extreme smoothness of the prior relative to the smoothness of the response function leads to very slow contraction rates for such functions. A remedy for this mismatch is to rescale the sample paths. The length scale of the process can be treated as a hyperparameter and can be endowed with a prior of its own, or can be selected using an empirical Bayes procedure. Van der

Vaart and Van Zanten (2007) and Van der Vaart and Van Zanten (2009) for example show that the prior $x \mapsto f(Ax)$, for f the squared exponential process and A^d an independent Gamma distributed random variable, leads to optimal contraction rates for β-smooth true response functions, for any $\beta > 0$.

Actually, the preceding discussion permits only the derivation of an *upper bound* on the contraction rate. In the next theorem we show that the logarithmic rate is real however. The theorem shows that asymptotically, balls around f_0 of logarithmic radius receive zero posterior mass. The proof, following an idea of Castillo (2008) and given in Section 4.4, is based on the fact that balls of this type also receive very little *prior* mass, essentially because the inequality of the preceding lemma can be reversed.

Theorem 8 *If f_0 is contained in $H^\beta[0,1]^d$ for some $\beta > d/2$, has support within $(0,1)^d$ and possesses a Fourier transform satisfying $|\hat{f}_0(\lambda)| \gtrsim \|\lambda\|^{-k}$ for some $k > 0$ and every $\|\lambda\| \geq 1$, then there exists a constant l such that $\mathrm{E}_{f_0}\Pi\big(f\colon \|f - f_0\|_2 \leq (\log n)^{-l} | X_{1:n}, Y_{1:n}\big) \to 0$.*

As the prior puts all of its mass on analytic functions, perhaps it is not fair to study its performance only for β-regular functions, and it makes sense to study the concentration function also for "supersmooth", analytic response functions as well. The functions in the RKHS of the squared exponential process are examples of supersmooth functions, and for those functions we obtain the rate $\psi_0^{-1}(n)$ determined by the (centered) small ball probability only. In view of Lemma 6 this is a $1/\sqrt{n}$-rate up to a logarithmic factor.

The following lemma deals with the infimum part of the concentration function in the case that that the function f_0 is supersmooth. Recall the definition of the space $\mathcal{A}^{\gamma,r}(\mathbb{R}^d)$ of analytic functions given in Section 1.5.

Lemma 9
- *If f_0 is the restriction to $[0,1]^d$ of an element of $\mathcal{A}^{\gamma,r}(\mathbb{R}^d)$, for $r > 2$, or for $r \geq 2$ with $\gamma \geq 4$, then $f_0 \in \mathbb{H}$.*
- *If f_0 is the restriction to $[0,1]^d$ of an element of $\mathcal{A}^{\gamma,r}(\mathbb{R}^d)$ for $r < 2$, then there exist a constant C depending on f_0 such that*

$$\inf_{\|h-w\|_\infty \leq \varepsilon} \|h\|_{\mathbb{H}}^2 \leq C e^{\left(\log(1/\varepsilon)\right)^{2/r}/(4\gamma^{2/r})}.$$

Combination of Lemmas 6 and 9 with the general theorems yields the following result.

Theorem 10 *Suppose that we use a squared exponential prior and f_0 is the restriction to $[0,1]^d$ of an element of $\mathcal{A}^{\gamma,r}(\mathbb{R}^d)$, for $r \geq 1$ and $\gamma > 0$. Then both in the fixed and the random design cases the posterior contracts at the rate $(\log n)^{1/r}/\sqrt{n}$.*

Observe that the rate that we get in the last theorem is up to a logarithmic factor equal to the rate $1/\sqrt{n}$ at which the posterior typically contracts for parametric models (cf., the Bernstein-von Mises theorem, for example, Van der Vaart, 1998). This "almost parametric rate" is explainable from the fact that spaces of analytic functions are only slightly bigger than finite-dimensional spaces in terms of their metric entropy (see Kolmogorov and Tihomirov, 1961).

Together, Theorems 8 and 10 give the same general message for the squared exponential kernel as Theorem 5 does for the Matérn kernel: fast convergence rates are only attained if the smoothness of the prior matches the smoothness of the response function f_0. However, generally the assumption of existence of infinitely many derivatives of a true response function ($f_0 \in \mathcal{A}^{g,r}(\mathbb{R}^d)$) is considered too strong to define a test case for nonparametric learning. If this assumption holds, then the response function f_0 can be recovered at a very fast rate, but this is poor evidence of good performance, as only few functions satisfy the assumption. Under the more truly "nonparametric assumption" that f_0 is β-regular, the performance of the squared-exponential prior is disastrous (unless the length scale is changed appropriately in a data-dependent way).

4. Proofs

This section contains the proofs of the presented results.

4.1 Proof of Theorem 1

The proof of Theorem 1 is based on estimates of the prior mass near the true parameter f_0 and on the metric entropy of the support of the prior. This is expressed in the following proposition.

We use the notation $D(\varepsilon, \mathcal{A}, d)$ for the ε-packing number of the metric space $(\mathcal{A}, d)$: the maximal number of points in $\mathcal{A}$ such that every pair has distance at least ε relative to d.

Proposition 11 *Suppose that for some $\varepsilon > 0$ with $\sqrt{n}\varepsilon \geq 1$ and for every $r > 1$ there exists a set $\mathcal{F}_r$ such that*

$$D\big(\varepsilon, \mathcal{F}_r, \|\cdot\|_n\big) \leq e^{n\varepsilon^2 r^2}, \qquad (15)$$
$$\Pi(\mathcal{F}_r) \geq 1 - e^{-2n\varepsilon^2 r^2}.$$

Furthermore, suppose that

$$\Pi\big(f\colon \|f - f_0\|_n \leq \varepsilon\big) \geq e^{-n\varepsilon^2}. \qquad (16)$$

Then

$$P_{n,f_0} \int \|f - f_0\|_n^l \, d\Pi_n\big(f | Y_{1:n}\big) \lesssim \varepsilon^l.$$

For $\theta \in \mathbb{R}^n$ let $P_{n,\theta}$ be the normal distribution $N_n(\theta, I)$. In the following three lemmas let $\|\cdot\|$ be the Euclidean norm on $\mathbb{R}^n$.

Lemma 12 *For any $\theta_0, \theta_1 \in \mathbb{R}^n$, there exists a test ϕ based on $Y \sim N_n(\theta, I)$ such that, for every $\theta \in \mathbb{R}^n$ with $\|\theta - \theta_1\| \leq \|\theta_0 - \theta_1\|/2$,*

$$P_{n,\theta_0}\phi \vee P_{n,\theta}(1 - \phi) \leq e^{-\|\theta_0 - \theta_1\|^2/8}.$$

Proof For simplicity of notation we can choose $\theta_0 = 0$. If $\|\theta - \theta_1\| \leq \|\theta_1\|/2$, then $\|\theta\| \geq \|\theta_1\|/2$ and hence $\langle \theta, \theta_1 \rangle = \big(\|\theta\|^2 + \|\theta_1\|^2 - \|\theta - \theta_1\|^2\big)/2 \geq \|\theta_1\|^2/2$. Therefore, the test $\phi = 1_{\theta_1^T Y > D\|\theta_1\|}$ satisfies, with Φ the standard normal cdf,

$$P_{n,\theta_0}\phi = 1 - \Phi(D),$$
$$P_{n,\theta}(1 - \phi) = \Phi\big((D\|\theta_1\| - \langle \theta, \theta_1 \rangle)/\|\theta_1\|\big) \leq \Phi(D - \rho),$$

for $\rho = \|\theta_1\|/2$. The infimum over D of $\big(1 - \Phi(D)\big) + \Phi(D - \rho)$ is attained for $D = \rho/2$, for which $D - \rho = -\rho/2$. We substitute this in the preceding display and use the bound $1 - \Phi(x) \leq e^{-x^2/2}$, valid for $x \geq 0$. ■

Let $D(\varepsilon, \Theta)$ be the maximal number of points that can be placed inside the set $\Theta \subset \mathbb{R}^n$ such that any pair has Euclidean distance at least ε.

Lemma 13 *For any $\Theta \subset \mathbb{R}^n$ there exists a test ϕ based on $Y \sim N_n(\theta, I)$ with, for any $r > 1$ and every integer $j \geq 1$,*

$$P_{n,\theta_0}\phi \leq 9D(r/2, \Theta)\exp(-r^2/8),$$
$$\sup_{\theta \in \Theta: \|\theta - \theta_0\| \geq jr} P_{n,\theta}(1-\phi) \leq \exp(-j^2 r^2/8).$$

Proof The set Θ can be partitioned into the shells

$$C_{j,r} = \big\{\theta \in \Theta: jr \leq \|\theta - \theta_0\| < (j+1)r\big\}.$$

We place in each of these shells a maximal collection Θ_j of points that are $jr/2$-separated, and next construct a test ϕ_j as the maximum of all the tests as in the preceding lemma attached to one of these points. The number of points is equal to $D(jr/2, C_{j,r})$. Every $\theta \in C_{j,r}$ is in a ball of radius $jr/2$ of some point $\theta_1 \in \Theta_j$ and satisfies $\|\theta - \theta_1\| \leq jr/2 \leq \|\theta_0 - \theta_1\|/2$, since $\theta_1 \in C_{j,r}$. Hence each test satisfies the inequalities of the preceding lemma. It follows that

$$P_{n,\theta_0}\phi_j \leq D(jr/2, C_{j,r})e^{-j^2r^2/8},$$
$$\sup_{\theta \in C_{j,r}} P_{n,\theta}(1-\phi_j) \leq e^{-j^2r^2/8}.$$

Finally, we construct ϕ as the supremum over all tests ϕ_j, for $j \geq 1$. We note that $\sum_{j\geq 1} D(jr/2, C_{j,r})e^{-j^2r^2/8} \leq D(r/2, \Theta)e^{-r^2/8}/(1-e^{-r^2/8})$, and $1/(1-e^{-1/8}) \approx 8.510$. ■

Lemma 14 *For any probability distribution Π on $\mathbb{R}^n$ and $x > 0$,*

$$P_{n,\theta_0}\Big(\int \frac{p_{n,\theta}}{p_{n,\theta_0}}\, d\Pi(\theta) \leq e^{-\sigma_0^2/2 - \|\mu_0\|x}\Big) \leq e^{-x^2/2},$$

for $\mu_0 = \int (\theta - \theta_0)\, d\Pi(\theta)$ and $\sigma_0^2 = \int \|\theta - \theta_0\|^2\, d\Pi(\theta)$. Consequently, for any probability distribution Π on $\mathbb{R}^n$ and any $r > 0$,

$$P_{n,\theta_0}\Big(\int \frac{p_{n,\theta}}{p_{n,\theta_0}}\, d\Pi(\theta) \geq e^{-r^2}\Pi\big(\theta: \|\theta - \theta_0\| < r\big)\Big) \geq 1 - e^{-r^2/8}.$$

Proof Under θ_0 the variable $\int \log(p_{n,\theta}/p_{n,\theta_0})\, d\Pi(\theta) = \mu_0^T(Y - \theta_0) - \sigma_0^2/2$ is normally distributed with mean $-\sigma_0^2/2$ and variance $\|\mu_0\|^2$. Therefore, the event B_n that this variable is smaller than $-\sigma_0^2/2 - \|\mu_0\|x$ has probability bounded above by $\Phi(-x) \leq e^{-x^2/2}$. By Jensen's inequality applied to the logarithm, the event in the left side of the lemma is contained in B_n.

To prove the second assertion we first restrict the integral $\int p_{n,\theta}/p_{n,\theta_0}\, d\Pi(\theta)$ to the ball $\{\theta: \|\theta - \theta_0\| \leq r\}$, which makes it smaller. Next we divide by $\Pi\big(\theta: \|\theta - \theta_0\| < r\big)$ to renormalize Π to a probability measure on this ball, and apply the first assertion with this renormalized measure Π. The relevant characteristics of the renormalized measure satisfy $\|\mu_0\| \le r$ and $\sigma_0^2 \le r^2$. Therefore the assertion follows upon choosing $x = r/2$. ■

Proof [Proof of Proposition 11] For any event $\mathcal{A}$, any test ϕ and any $r > 1$, the expected value $P_{n,f_0}\Pi\big(f: \|f - f_0\|_n > 4\varepsilon r | Y_{1:n}\big)$ is bounded by $A + B + C + D$, for

$$\begin{aligned}
A &= P_{n,f_0}\phi,\\
B &= P_{n,f_0}(\mathcal{A}^c)\\
C &= P_{n,f_0}\Pi_n\big(f \notin \mathcal{F}_r | Y_{1:n}\big)1_{\mathcal{A}},\\
D &= P_{n,f_0}\Pi_n\big(f \in \mathcal{F}_r: \|f - f_0\|_n > 4\varepsilon r | Y_{1:n}\big)(1-\phi)1_{\mathcal{A}}.
\end{aligned}$$

For the test ϕ given by Lemma 13 with Θ the set of all vectors $\big(f(x_1), \dots, f(x_n)\big)$ as f ranges over $\mathcal{F}_r$, with θ_0 this vector at $f = f_0$, and with r taken equal to $4\sqrt{n}\varepsilon r$, we obtain, for $4\sqrt{n}\varepsilon r > 1$,

$$A \le 9D(2\sqrt{n}\varepsilon r, \Theta)e^{-2n\varepsilon^2 r^2} \le 9e^{-n\varepsilon^2 r^2}.$$

In view of Lemma 14 applied with r equal to $\sqrt{n}\varepsilon r$, there exists an event $\mathcal{A}$ such that

$$B \le e^{-n\varepsilon^2 r^2/8},$$

while on the event $\mathcal{A}$,

$$\int \frac{p_{n,f}}{p_{n,f_0}}\, d\Pi(f) \ge e^{-n\varepsilon^2 r^2}\Pi\big(f: \|f - f_0\|_n < \varepsilon r\big) \ge e^{-n\varepsilon^2(r^2+1)}.$$

It follows that on the event $\mathcal{A}$, for any set $\mathcal{B}$,

$$\Pi_n(\mathcal{B}|Y_{1:n}) \le e^{n\varepsilon^2(r^2+1)} \int_{\mathcal{B}} p_{n,f}/p_{n,f_0}\, d\Pi(f).$$

Therefore, in view of the fact that $P_{n,f_0}(p_{n,f}/p_{n.f_0}) \le 1$, we obtain,

$$\begin{aligned}
C &\le e^{n\varepsilon^2(r^2+1)} P_{n,f_0} \int_{\mathcal{F}_r^c} p_{n,f}/p_{n,f_0}\, d\Pi(f)\\
&\le e^{n\varepsilon^2(r^2+1)}\Pi(\mathcal{F}_r^c) \le e^{-n\varepsilon^2(r^2-1)}. \qquad (17)
\end{aligned}$$

Finally, in view of the fact that $P_{n,f_0}(p_{n,f}/p_{n.f_0})(1-\phi) \le P_{n,f}(1-\phi)$, which is bounded above by $e^{-2j^2 n\varepsilon^2 r^2}$ for f contained in $C_{j,r} := \{f \in \mathcal{F}_{n,r}: 4j\varepsilon r \le \|f - f_0\|_n < 4(j+1)\varepsilon r\}$ by the second inequality in Lemma 13, we obtain, again using Fubini's theorem,

$$\begin{aligned}
D &\le e^{n\varepsilon^2(r^2+1)} \sum_{j\ge1} P_{n,f_0}(1-\phi)\int_{C_{j,r}} p_{n,f}/p_{n,f_0}\, d\Pi(f)\\
&\le e^{n\varepsilon^2(r^2+1)} \sum_{j\ge1} e^{-2j^2 n\varepsilon^2 r^2} \le 9e^{-n\varepsilon^2(r^2-1)},
\end{aligned}$$

for $n\varepsilon^2 r^2 \ge 1/16$, as $1/(1-e^{-1/8}) \approx 8.5$.

Finally we write

$$\begin{aligned}
P_{n,f_0} \int \|f-f_0\|_n^l \, d\Pi_n\big(f|\,Y_{1:n}\big) \\
&= P_{n,f_0} \int_0^\infty l r^{l-1} \Pi_n\big(\|f-f_0\|_n > 4\varepsilon r|\,Y_{1:n}\big)\, dr\, (4\varepsilon)^l \\
&\le (8\varepsilon)^l + (4\varepsilon)^l P_{n,f_0} \int_2^\infty l r^{l-1} (A+B+C+D)(r)\, dr.
\end{aligned}$$

Inserting the bound on $A+B+C+D$ obtained previously we see that the integral is bounded by $10\int_2^\infty (e^{-r^2/8}+e^{-(r^2-1)})\,dr<\infty$. ∎

Proof [Proof of Theorem 1] Theorem 1 is a specialization of Proposition 11 to Gaussian priors, where the conditions of the proposition are reexpressed in terms of the concentration function ϕ_{f_0} of the prior. The details are the same as in Van der Vaart and Van Zanten (2008a).

First we note that $\varepsilon := 2\psi_{f_0}^{-1}(n)$ satisfies $\phi_{f_0}(\varepsilon/2) \le n\varepsilon^2/4 \le n\varepsilon^2$. It is shown in Kuelbs et al. (1994) (or see Lemma 5.3 in Van der Vaart and Van Zanten, 2008b) that the concentration function ϕ_{f_0} determines the small ball probabilities around f_0, in the sense that, for the given ε,

$$\Pi\big(f: \|f-f_0\|_\infty < \varepsilon\big) \ge e^{-n\varepsilon^2}. \tag{18}$$

Because $\|\cdot\|_n \le \|\cdot\|_\infty$, it follows that (16) is satisfied.

For $\mathbb{H}_1$ and $\mathbb{B}_1$ the unit balls of the RKHS and $\mathbb{B}$ and $M_r = -2\Phi^{-1}(e^{-n\varepsilon^2 r^2})$, we define sets $\mathcal{F}_r = \varepsilon\mathbb{B}_1 + M_r\mathbb{H}_1$. By Borell's inequality (see Borell, 2008, or Theorem 5.1 in Van der Vaart and Van Zanten, 2008b) these sets have prior probability $\Pi(\mathcal{F}_r)$ bounded below by $1-\Phi(\alpha+M_r)$, for Φ the standard normal distribution function and α the solution to the equation $\Phi(\alpha) = \Pi\big(f: \|f\|_\infty < \varepsilon\big) = e^{-\phi_o(\varepsilon)}$. Because $\Phi(\alpha) \ge e^{-n\varepsilon^2} \ge e^{-n\varepsilon^2 r^2}$, we have $\alpha + M_r \ge -\Phi^{-1}(e^{-n\varepsilon^2 r^2})$. We conclude that $\Pi(\mathcal{F}_r) \ge 1 - e^{-n\varepsilon^2 r^2}$.

It is shown in the proof of Theorem 2.1 of Van der Vaart and Van Zanten (2008a) that the sets $\mathcal{F}_r$ also satisfy the entropy bound (15), for the norm $\|\cdot\|_\infty$, and hence certainly for $\|\cdot\|_n$. ∎

4.2 Proof of Theorem 2

For a function $f: [0,1]^d \to \mathbb{R}$ and $\alpha > 0$ let $\|f\|_{\alpha|\infty}$ be the Besov norm of regularity α measured using the $L_\infty - L_\infty$-norms (see (19) below). This is bounded by the Hölder norm of order α (see for instance Cohen et al., 2001 for details).

Lemma 15 *Let $\mathcal{X} = [0,1]^d$ and suppose that the density of the covariates is bounded below by a constant c. Then $\|f\|_\infty \lesssim c^{-2\alpha/(2\alpha+d)} \|f\|_{\alpha|\infty}^{d/(2\alpha+d)} \|f\|_2^{2\alpha/(2\alpha+d)}$, for any function $f: [0,1]^d \to \mathbb{R}$.*

Proof We can assume without loss of generality that the covariate distribution is the uniform distribution. We can write the function as the Fourier series $f = \sum_{j=0}^\infty \sum_k \sum_v \beta_{j,k,v} e_{j,k,v}$ relative to a basis $(e_{j,k,v})$ of orthonormal wavelets in $L_2(\mathbb{R}^d)$. (Here k runs for each fixed j through an index set for of the order $O(2^{jd})$ translates, and v runs through $\{0,1\}^d$ when $j=0$ and $\{0,1\}^d \setminus \{0\}$ when $j \ge 1$.)

For wavelets constructed from suitable scaling functions, the various norms of f can be expressed in the coefficients through (up to constants, see for instance Cohen et al., 2001, Section 2)

$$\begin{aligned}
\|f\|_2 &= \Big(\sum_j \sum_k \sum_v \beta_{j,k,v}^2\Big)^{1/2}, \\
\|f\|_\infty &\le \sum_j \max_k \max_v |\beta_{j,k,v}| 2^{jd/2}, \\
\|f\|_{\alpha|\infty} &= \sup_j \max_k \max_v |\beta_{j,k,v}| 2^{j(\alpha+d/2)}. \qquad (19)
\end{aligned}$$

For given J let $f_J = \sum_{j\le J}\sum_k \sum_v \beta_{j,k,v} e_{j,k,v}$ be the projection of f on the base elements of resolution level bounded by J. Then

$$\begin{aligned}
\|f-f_J\|_\infty &\le \sum_{j>J} \max_k \max_v |\beta_{j,k,v}| 2^{jd/2} \\
&\le \sum_{j>J} 2^{-j(\alpha+d/2)} \|f\|_{\alpha|\infty} 2^{jd/2} \le 2^{-J\alpha}\|f\|_{\alpha|\infty}.
\end{aligned}$$

Furthermore, by the Cauchy-Schwarz inequality,

$$\begin{aligned}
\|f_J\|_\infty &\le \sum_{j\le J} \max_k \max_v |\beta_{j,k,v}| 2^{jd/2} \\
&\le \Big(\sum_{j\le J} \max_k \max_v \beta_{j,k,v}^2\Big)^{1/2} \Big(\sum_{j\le J} 2^{jd}\Big)^{1/2} \\
&\le \|f\|_2 2^{Jd/2},
\end{aligned}$$

where in the last inequality we have bounded the maximum over (k,v) by the sum.

Combining the two preceding displays we see that $\|f\|_\infty \le 2^{-J\alpha}\|f\|_{\alpha|\infty} + \|f\|_2 2^{Jd/2}$. We finish the proof by choosing J to balance the two terms on the right. ■

Proof [Proof of Theorem 2] Let $\varepsilon = 2\psi_{f_0}^{-1}(n)$ so that $\phi_{f_0}(\varepsilon/2) \le n\varepsilon^2$ and (18) holds. By the definition of ϕ_{f_0} there exists an element f_ε of the RKHS of the prior with $\|f_\varepsilon - f_0\|_\infty \le \varepsilon/2$ and $\|f_\varepsilon\|_{\mathbb{H}}^2 \le \phi_{f_0}(\varepsilon/2) \le n\varepsilon^2$. Because $\|f_\varepsilon - f_0\|_2 \le \|f_\varepsilon - f_0\|_\infty \le \varepsilon$, the posterior second moments of $\|f-f_\varepsilon\|_2$ and $\|f-f_0\|_2$ are within a multiple of ε^2, and hence it suffices to bound the former of the two.

For any positive constants γ,τ, any $\eta \ge \varepsilon$, and any events $\mathcal{A}_r$ we can bound

$$\begin{aligned}
&\frac{1}{\eta^2} \mathrm{E}_{f_0} \int \|f-f_\varepsilon\|_2^2 \, d\Pi(f|X_{1:n},Y_{1:n}) \\
&\qquad = \mathrm{E}_{f_0} \int_0^\infty r \Pi\big(f\colon \|f-f_\varepsilon\|_2 > \eta r | X_{1:n}, Y_{1:n}\big)\, dr
\end{aligned}$$

by $I+II+III+IV$, for

$$\begin{aligned}
I &= \mathrm{E}_{f_0}\int_0^\infty r\Pi\big(f\colon 2\|f-f_\varepsilon\|_n > \eta r | X_{1:n}, Y_{1:n}\big)\,dr,\\
II &= \mathrm{E}_{f_0}\int_0^\infty r1_{\mathcal{A}_r^c}\,dr,\\
III &= \mathrm{E}_{f_0}\int_0^\infty r1_{\mathcal{A}_r}\Pi\big(\|f\|_{\alpha|\infty} > \tau\sqrt{n}\eta r^\gamma | X_{1:n}, Y_{1:n}\big)\,dr,\\
IV &= \mathrm{E}_{f_0}\int_0^\infty r1_{\mathcal{A}_r}\Pi\big(f\colon \|f-f_\varepsilon\|_2 > \eta r \geq 2\|f-f_\varepsilon\|_n,\\
&\qquad\qquad \|f\|_{\alpha|\infty} \leq \tau\sqrt{n}\eta r^\gamma | X_{1:n}, Y_{1:n}\big)\,dr.
\end{aligned}$$

The term I is the quadratic risk in terms of the empirical norm, centered at f_ε. Conditioned on the design points and centered at f_0 this was seen to be bounded in the previous section (as $\eta \geq \varepsilon$), uniformly in the design points. Because $\|f_0 - f_\varepsilon\|_\infty \leq \varepsilon$, the term I is bounded by a constant.

In view of Lemma 14, with r of the lemma equal to $\sqrt{n}\varepsilon r^\gamma$, there exist events $\mathcal{A}_r$ such that

$$II \leq \int_0^\infty re^{-n\varepsilon^2 r^{2\gamma}/8}\,dr \lesssim 1,$$

while on the event $\mathcal{A}_r$,

$$\begin{aligned}
\int \frac{p_{n,f}}{p_{n,f_0}}\,d\Pi(f) &\geq e^{-n\varepsilon^2 r^{2\gamma}}\Pi\big(f\colon \|f-f_0\|_n < \varepsilon r^\gamma\big)\\
&\geq e^{-n\varepsilon^2(r^{2\gamma}+1)}, \qquad (20)
\end{aligned}$$

by (18) and because $\|\cdot\|_n \leq \|\cdot\|_\infty$.

Because the prior Π is concentrated on the functions with $\|f\|_{\alpha|\infty} < \infty$ by assumption, it can be viewed as the distribution of a Gaussian random element with values in the Hölder space $C^\alpha[0,1]^d$. It follows that $\tau^2 \colon= 16\int \|f\|_{\alpha|\infty}^2\,d\Pi(f)$ is finite, and $\Pi\big(f\colon \|f\|_{\alpha|\infty} > \tau x\big) \leq e^{-2x^2}$, for every $x > 0$, by Borell's inequality (e.g., Van der Vaart and Wellner, 1996, A.2.1.). By the same argument as used to obtain (17) in the proof of Proposition 11, we see that

$$\begin{aligned}
III &\leq 1 + \int_1^\infty re^{n\varepsilon^2(r^{2\gamma}+1)}\Pi\big(f\colon \|f\|_{\alpha|\infty} > \tau\sqrt{n}\eta r^\gamma\big)\,dr\\
&\leq 1 + \int_1^\infty re^{n\varepsilon^2(r^{2\gamma}+1)}e^{-2n\eta^2 r^{2\gamma}}\,dr \lesssim 2.
\end{aligned}$$

It remains to prove that IV is bounded as well.

The squared empirical norm $\|f-f_\varepsilon\|_n^2$ is the average of the independent random variables $(f - f_\varepsilon)^2(X_i)$, which have expectation $\|f-f_\varepsilon\|_2^2$, and variance bounded by $P(f-f_\varepsilon)^4 \leq \|f-f_\varepsilon\|_2^2\|f - f_\varepsilon\|_\infty^2$. Therefore, we can apply Bernstein's inequality (see, e.g., Lemma 2.2.9 in Van der Vaart and Wellner, 1996) to see that

$$\mathrm{P}\big(\|f-f_\varepsilon\|_2 \geq 2\|f-f_\varepsilon\|_n\big) \leq e^{-(n/5)\|f-f_\varepsilon\|_2^2/\|f-f_\varepsilon\|_\infty^2}.$$

The unit ball of the RKHS of a GP f is always contained in c times the unit ball of the Banach space on which it is supported, for $c^2 = \mathrm{E}\|f\|^2$, where $\|\cdot\|$ is the norm of the Banach space (see, e.g., Van der Vaart and Van Zanten, 2008b), formula (2.5)). An equivalent statement is that the Banach norm $\|f\|$ of an element of the RKHS is bounded above by c times its RKHS-norm. Because Π is concentrated on $C^\alpha[0,1]^d$, we can apply this general fact with $\|\cdot\|$ the α-Hölder norm, and conclude that the α-Hölder norm of an element of the RKHS is bounded above by $\tau/4$ times its RKHS-norm, for $\tau/4$ the second moment of the prior norm defined previously. In particular $\|f_\varepsilon\|_{\alpha|\infty} \leq \tau\|f_\varepsilon\|_{\mathbb{H}} \leq \tau\sqrt{n}\varepsilon$. Therefore, for f in the set $\mathcal{F}$ of functions with $\|f\|_{\alpha|\infty} \leq \tau\sqrt{n}\varepsilon r^\gamma$, we have $\|f - f_\varepsilon\|_{\alpha|\infty} \leq 2\tau\sqrt{n}\varepsilon r^\gamma$, whence by Lemma 15 for $f \in \mathcal{F}$ we can replace $\|f - f_\varepsilon\|_\infty$ in the preceding display by $c(2\tau\sqrt{n}\varepsilon r^\gamma)^{d/(2\alpha+d)}\|f - f_\varepsilon\|_2^{2\alpha/(2\alpha+d)}$, for a constant c depending on the covariate density. We then have

$$\begin{aligned}
&\mathrm{E}_{f_0}\Pi\big(f \in \mathcal{F}: \|f - f_\varepsilon\|_2 > \eta r \geq 2\|f - f_\varepsilon\|_n\big)\\
&\quad\leq \int_{f\in\mathcal{F}:\|f-f_\varepsilon\|_2>\eta r} \mathrm{P}\big(\|f - f_\varepsilon\|_2 \geq 2\|f - f_\varepsilon\|_n\big)\, d\Pi(f)\\
&\quad\leq \int_{\|f-f_\varepsilon\|_2>\eta r} \exp\Big(-\frac{n}{5c^2}\Big(\frac{\|f - f_\varepsilon\|_2}{2\tau\sqrt{n}\varepsilon r^\gamma}\Big)^{2d/(2\alpha+d)}\Big)\, d\Pi(f)\\
&\quad\leq \exp\Big(-Cn^{2\alpha/(2\alpha+d)}(\eta r^{1-\gamma}/\varepsilon)^{2d/(2\alpha+d)}\Big),
\end{aligned}$$

for $1/C = 5c^2(2\tau)^{2d/(2\alpha+d)}$. Substitution of this bound and the lower bound (20) in IV yields

$$IV \leq 1 + \int_1^\infty r e^{n\varepsilon^2(r^{2\gamma}+1)} e^{-Cn^{2\alpha/(2\alpha+d)}(\eta r^{1-\gamma}/\varepsilon)^{2d/(2\alpha+d)}}\, dr.$$

For $Cn^{2\alpha/(2\alpha+d)}(\eta/\varepsilon)^{2d/(2\alpha+d)} \geq n\varepsilon^2$ this is finite if $\gamma > 0$ is chosen sufficiently small. Equivalently, IV is bounded if $\eta \gtrsim \sqrt{n}\varepsilon^{(2\alpha+2d)/d}$.

We must combine this with the requirement made at the beginning of the proof that $\eta \geq \varepsilon \geq 2\psi_{f_0}^{-1}(n)$. If $\varepsilon \leq n^{-d/(4\alpha+2d)}$, then $\sqrt{n}\varepsilon^{(2\alpha+2d)/d} \leq \varepsilon$ and hence the requirement $\eta \gtrsim \sqrt{n}\varepsilon^{(2\alpha+2d)/d}$ is satisfied for $\eta = \varepsilon$. Otherwise, we choose $\eta \sim \sqrt{n}\varepsilon^{(2\alpha+2d)/d} \gg \varepsilon$. In both cases we have proved that the posterior second moment has mean bounded by a multiple of η^2. ■

4.3 Proofs for Section 3

Proof [Proof of Lemma 3] The Fourier transform of h_ψ given in (11) is, up to constants, the function $\phi = \psi m$, and for ψ the minimal choice as in (12) this function satisfies (cf., (10))

$$\int \big|\phi(\lambda)\big|^2 \big(1 + \|\lambda\|^2\big)^{\alpha+d/2}\, d\lambda = \|h_\psi\|_{\mathbb{H}}^2.$$

In other words, the unit ball $\mathbb{H}_1$ of the RKHS is contained in a Sobolev ball of order $\alpha + d/2$. (See Section 1.5 for the definition of Sobolev spaces.) The metric entropy relative to the uniform norm of such a Sobolev ball is bounded by a constant times $(1/\varepsilon)^{d/(\alpha+d/2)}$ (see Theorem 3.3.2 on p. 105 in Edmunds and Triebel, 1996). The lemma next follows from the results of Kuelbs and Li (1993) and Li and Linde (1998) that characterize the small ball probability in terms of the entropy of the RKHS-unit ball. ■

Proof [Proof of Lemma 4] Let $\kappa:\mathbb{R}\to\mathbb{R}$ be a function with a real, symmetric Fourier transform $\hat\kappa$, which equals $1/(2\pi)$ in a neighborhood of 0 and which has compact support. From $\hat\kappa(\lambda)=(2\pi)^{-1}\int e^{i\lambda t}\kappa(t)\,dt$ it then follows that $\int\kappa(t)\,dt=1$ and $\int(it)^k\kappa(t)\,dt=0$ for $k\geq 1$. For $t=(t_1,\ldots,t_d)$, define $\phi(t)=\kappa(t_1)\cdots\kappa(t_d)$. Then ϕ integrates to 1, has finite absolute moments of all orders, and vanishing moments of all orders bigger than 0.

For $\sigma>0$ set $\phi_\sigma(x)=\sigma^{-d}\phi(x/\sigma)$ and $h=\phi_\sigma * f_0$. Because ϕ is a higher order kernel, standard arguments from the theory of kernel estimation shows that $\|f_0-\phi_\sigma*f_0\|_\infty\lesssim\sigma^\beta$.

The Fourier transform of h is the function $\lambda\mapsto\hat h(\lambda)=\hat\phi(\sigma\lambda)\hat f_0(\lambda)$, and therefore (12) and (13) show that

$$\begin{aligned}\|h\|_{\mathbb{H}}^2&\lesssim\int\left|\hat\phi(\sigma\lambda)\hat f_0(\lambda)\right|^2\frac{1}{m(\lambda)}\,d\lambda\\&\lesssim\sup_\lambda\Big[\big(1+\|\lambda\|^2\big)^{\alpha+d/2-\beta}\left|\hat\phi(\sigma\lambda)\right|^2\Big]\|f_0\|_{\beta|2}^2\\&\lesssim C(\sigma)\sup_\lambda\Big[\big(1+\|\lambda\|^2\big)^{\alpha+d/2-\beta}\left|\hat\phi(\lambda)\right|^2\Big]\|f_0\|_{\beta|2}^2.\end{aligned}$$

for

$$C(\sigma)=\sup_\lambda\Big(\frac{1+\|\lambda\|^2}{1+\|\sigma\lambda\|^2}\Big)^{\alpha+d/2-\beta}\lesssim\Big(\frac{1}{\sigma}\Big)^{2\alpha+d-2\beta},$$

if $\sigma\leq 1$. The assertion of the lemma follows upon choosing $\sigma\sim\varepsilon^{1/\beta}$. ■

Proof [Proof of Lemma 7] For given $K>0$ let $\psi(\lambda)=(\hat f_0/m)(\lambda)1_{\|\lambda\|\leq K}$. The function h_ψ defined by (11) with m given in (14) satisfies

$$\begin{aligned}\|h_\psi-f_0\|_\infty&\leq\int_{\|\lambda\|>K}|\hat f_0(\lambda)|\,d\lambda\\&\leq\|f_0\|_{\beta|2}\Big(\int_{\|\lambda\|>K}\big(1+\|\lambda\|^2\big)^{-\beta}\,d\lambda\Big)^{1/2}\\&\lesssim\|f_0\|_{\beta|2}\frac{1}{K^{\beta-d/2}}.\end{aligned}$$

Furthermore, the squared RKHS-norm of h_ψ is given by

$$\begin{aligned}\|h_\psi\|_{\mathbb{H}}^2&=\int_{\|\lambda\|\leq K}\frac{|\hat f_0|^2}{m}(\lambda)\,d\lambda\\&\leq\sup_{\|\lambda\|\leq K}m(\lambda)^{-1}\big(1+\|\lambda\|^2\big)^{-\beta}\|f_0\|_{\beta|2}^2\\&\lesssim e^{K^2/4}\|f_0\|_{\beta|2}^2.\end{aligned}$$

We conclude the proof by choosing $K\sim\varepsilon^{-1/(\beta-d/2)}$. ■

Proof [Proof of 9] The first assertion is proved in Van der Vaart and Van Zanten (2009), Lemma 4.4. The second assertion is proved in the same way as Lemma 7, where this time, with $\|f_0\|_{\mathcal{A}}$ the norm of f_0 in $\mathcal{A}^{\gamma,r}(\mathbb{R}^d)$,

$$\begin{aligned}\|h_\psi - f_0\|_\infty^2 &\le \int_{\|\lambda\|>K} e^{-\gamma\|\lambda\|^r}\,d\lambda \|f_0\|_{\mathcal{A}}^2\\ &\le e^{-\gamma K^r} K^{-r+1}\|f_0\|_{\mathcal{A}}^2,\\ \|h_\psi\|_{\mathbb{H}}^2 &\le \sup_{\|\lambda\|\le K} e^{\|\lambda\|^2/4-\gamma\|\lambda\|^r}\|f_0\|_{\mathcal{A}}^2 \le e^{K^2/4}\|f_0\|_{\mathcal{A}}^2.\end{aligned}$$

We finish by choosing $K \sim \big(\gamma^{-1}\log(1/\varepsilon)\big)^{1/r}$. ■

4.4 Miscellaneous Results

Proof [Proof of Theorem 8] We start by proving the following lower bound on the concentration function: there exists $b, v > 0$ such that for $\varepsilon \downarrow 0$,

$$\begin{aligned}\phi_{f_0}(\varepsilon) &\ge \inf_{\psi:\|h_\psi - f_0\|_2<\varepsilon} \|h_\psi\|_{\mathbb{H}}^2 \\ &\gtrsim \exp\big(b\varepsilon^{-v}\big).\end{aligned} \tag{21}$$

For given $\varepsilon > 0$ let h_ψ be a function in the RKHS of the form (11) such that $\|h_\psi - f_0\|_2 < \varepsilon$. Let r be a function which is equal to 1 on the support of f_0, has itself support within $[0,1]$ and Fourier transform with exponentially small tails: $|\hat{r}(\lambda)\exp(|\lambda|^u) \to 0$ as $|\lambda| \to \infty$, for some $u > 0$. (Such a function exists for $u < 1$.) Then $h_\psi r$ has support inside $[0,1]$ and $f_0 r = f_0$, so that $\|h_\psi r - f_0\|_{2,\mathbb{R}} \le \|h_\psi - f_0\|_2$, where $\|\cdot\|_{2,\mathbb{R}}$ is the norm of $L_2(\mathbb{R}^d)$ and $\|\cdot\|_2$ the norm of $L_2[0,1]^d$. The function $h_\psi r$ has Fourier transform $(\psi m) * \hat{r}$, and hence by Parseval's identity $\big\|(\psi m) * \hat{r} - \hat{f}_0\big\|_{2,\mathbb{R}} < \varepsilon$. Therefore, for $K > 0$ and χ_K the indicator of the set $\{\lambda \in \mathbb{R}^d : \|\lambda\| > K\}$,

$$\big\|(\psi m) * \hat{r}\chi_{2K}\big\|_{2,\mathbb{R}} \ge \|\hat{f}_0\chi_{2K}\|_{2,\mathbb{R}} - \varepsilon \ge c(1/K)^{k-d/2} - \varepsilon,$$

by the assumption on $\hat{f}_0$, for some constant c. By Lemma 16 with $A = K/2$ and $2K$ instead of K, it follows that

$$\|\psi m\chi_K\|_{2,\mathbb{R}}\|\hat{r}(1-\chi_K)\|_{1,\mathbb{R}} \ge c(1/K)^{k-d/2} - \varepsilon - \|\psi m\|_{2,\mathbb{R}}\|\hat{r}\chi_K\|_1.$$

In view of (12) we have that $\|h_\psi\|_{\mathbb{H}} = \|\psi\sqrt{m}\|_{2,\mathbb{R}}$ and hence $\|\psi m\chi_K\|_{2,\mathbb{R}} \le \sqrt{m(K)}\|h_\psi\|_{\mathbb{H}}$, and $\|\psi m\|_{2,\mathbb{R}} \le \|h_\psi\|_{\mathbb{H}}$. Combining this with the preceding display we see that

$$\big(\|\hat{r}(1-\chi_K)\|_{1,\mathbb{R}}\sqrt{m(K)} + \|\hat{r}\chi_K\|_1\big)\|h_\psi\|_{\mathbb{H}} \ge c(1/K)^{k-d/2} - \varepsilon = \varepsilon,$$

for $K = (c/2\varepsilon)^{1/(k-d/2)}$. Here $\|\hat{r}(1-\chi_K)\|_{1,\mathbb{R}}\sqrt{m(K)}$ is of the order $\exp(-K^2/4)$, in view of the definition (14) of m and the fact that $\hat{r}$ is integrable, and $\|\hat{r}\chi_K\|_1$ is of the order $\exp(-dK^u)$, by construction. The proof of (21) is complete upon substituting $K = (c/2\varepsilon)^{1/(k-d/2)}$ and rearranging the preceding display.

The prior mass of a ball of radius ε around f_0 is bounded below by $e^{-\phi_{f_0}(\varepsilon/2)}$ and bounded above by $e^{-\phi_{f_0}(\varepsilon)}$, where we can use any norm. In view of (21) and Lemmas 6 and 7 we conclude that there exist constants such

$$\exp\bigl(-e^{a\varepsilon^{-u}}\bigr) \le \Pi\bigl(f\colon \|f-f_0\|_\infty < \varepsilon\bigr),$$
$$\Pi\bigl(f\colon \|f-f_0\|_2 < \varepsilon\bigr) \le \exp\bigl(-e^{b\varepsilon^{-v}}\bigr).$$

By choosing η_n, ε_n such that $a\varepsilon_n^{-u} = \log n^s$ and $b\eta_n^{-v} = \log n^t$, we obtain that

$$\frac{\Pi\bigl(f\colon \|f-f_0\|_2 < \eta_n\bigr)}{\Pi\bigl(f\colon \|f-f_0\|_\infty < \varepsilon_n\bigr)} \le \exp(-n^t + n^s) \ll e^{-2n\varepsilon_n^2},$$

if $t > 1 \vee s$. It then follows that $\mathrm{E}_{f_0}\Pi\bigl(f\colon \|f-f_0\|_2 < \eta_n | X_{1:n}, Y_{1:n}\bigr) \to 0$, by the same argument as given to prove (17). ■

If the convolution of a function f with a light-tailed function g has heavy tails, then f itself must have heavy tails. The following quantitative version of this principle underlies the preceding proof.

Lemma 16 *For arbitrary functions $f, g\colon \mathbb{R} \to \mathbb{R}$, χ_K the indicator function of $\{\lambda \in \mathbb{R}^d\colon \|\lambda\| > K\}$, and $0 < A < K$,*

$$\|f\chi_{K-A}\|_2 \|g(1-\chi_A)\|_1 \ge \|(f*g)\chi_K\|_2 - \|f\|_2 \|g\chi_A\|_1.$$

Proof For f_t the function $\lambda \mapsto f(\lambda - t)$, we have $\|f_t\chi_K\|_2 \le \|f\chi_{K-A}\|$ if $\|t\| \le A$, and $\|f_t\chi_K\|_2 \le \|f\|_2$ for every t. Therefore

$$\Bigl\|\int f_t\chi_K\, g(t)\, dt\Bigr\|_2 \le \int \|f_t\chi_K\|_2\, |g(t)|\, dt$$
$$\le \|f\chi_{K-A}\|_2 \int_{\|t\|\le A} |g(t)|\, dt + \|f\|_2 \int_{\|t\|>A} |g(t)|\, dt.$$

It suffices to arrange this inequality. ■

Lemma 17 *For ψ_{f_0} defined by (8) and $f_0 \in \mathbb{H}$ we have (9).*

Proof Because the function ψ_{f_0} is decreasing, the relation $\psi_{f_0}(\varepsilon) \le n$ for some ε implies that $\psi_{f_0}^{-1}(n) \le \varepsilon$. Consequently, if $\tilde\psi_{f_0}$ is an upper bound on ψ_{f_0}, then $\tilde\psi_{f_0}(\varepsilon) \le n$ for some ε implies that $\psi_{f_0}^{-1}(n) \le \varepsilon$. If $f_0 \in \mathbb{H}$, then we can choose $h = f_0$ in the infimum in the definition of ϕ_{f_0}, and hence we obtain

$$\phi_{f_0}(\varepsilon) \le \|f_0\|_{\mathbb{H}}^2 + \phi_0(\varepsilon).$$

If both $\|f_0\|_{\mathbb{H}}^2 \le n\varepsilon^2/2$ and $\psi_0(\varepsilon) \le n\varepsilon^2/2$, then $\tilde\psi_{f_0}(\varepsilon) \le n$ and hence $\psi_{f_0}^{-1}(n) \le \varepsilon$. ■

References

A. R. Barron. Information-theoretic characterization of Bayes performance and the choice of priors in parametric and nonparametric problems. In *Bayesian Statistics, 6 (Alcoceber, 1998)*, pages 27–52. Oxford Univ. Press, New York, 1999.

H. Bauer. *Measure and integration theory*, volume 26 of *de Gruyter Studies in Mathematics*. Walter de Gruyter & Co., Berlin, 2001.

C. Borell. Inequalities of the Brunn-Minkowski type for Gaussian measures. *Probab. Theory Related Fields*, 140(1-2):195–205, 2008.

I. Castillo. Lower bounds for posterior rates with Gaussian process priors. *Electron. J. Stat.*, 2: 1281–1299, 2008.

A. Cohen, W. Dahmen, I. Daubechies, and R. DeVore. Tree approximation and optimal encoding. *Appl. Comput. Harmon. Anal.*, 11(2):192–226, 2001.

D. E. Edmunds and H. Triebel. *Function Spaces, Entropy Numbers, Differential Operators*, volume 120 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 1996.

S. Ghosal, J. K. Ghosh, and A. W. van der Vaart. Convergence rates of posterior distributions. *Ann. Statist.*, 28(2):500–531, 2000.

I. Karatzas and S. E. Shreve. *Brownian Motion and Stochastic Calculus*, 2nd edition. Springer-Verlag, New York, 1991.

J. Kuelbs and W. V. Li. Metric entropy and the small ball problem for Gaussian measures. *J. Funct. Anal.*, 116(1):133–157, 1993.

J. Kuelbs, W. V. Li, and W. Linde. The Gaussian measure of shifted balls. *Probab. Theory Related Fields*, 98(2):143–162, 1994.

A. N. Kolmogorov and V. M. Tihomirov. ε-entropy and ε-capacity of sets in functional space. *Amer. Math. Soc. Transl. (2)*, 17: 277–364, 1961

W. V. Li and Q.-M. Shao. Gaussian processes: inequalities, small ball probabilities and applications. In *Stochastic Processes: Theory and Methods*, volume 19 of *Handbook of Statist.*, pages 533–597. North-Holland, Amsterdam, 2001.

W. V. Li and W. Linde. Existence of small ball constants for fractional Brownian motions. *C. R. Acad. Sci. Paris Sér. I Math.*, 326(11):1329–1334, 1998.

M. A. Lifshits. *Gaussian Random Functions*. Kluwer Academic Publishers, Dordrecht, 1995.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine learning*. MIT Press, Cambridge, MA, 2006.

M. W. Seeger, S. M. Kakade, and D. P. Foster. Information consistency of nonparametric Gaussian process methods. *IEEE Trans. Inform. Theory*, 54(5):2376–2382, 2008.

A. B. Tsybakov., *Introduction to Nonparametric Estimation*. Springer, New York, 2009.

A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, Cambridge, 1998.

A. W. van der Vaart and J. H. van Zanten. Bayesian inference with rescaled Gaussian process priors. *Electron. J. Stat.*, 1:433–448 (electronic), 2007.

A. W. van der Vaart and J. H. van Zanten. Rates of contraction of posterior distributions based on Gaussian process priors. *Ann. Statist.*, 36(3):1435–1463, 2008a.

A. W. van der Vaart and J. H. van Zanten. Reproducing kernel Hilbert spaces of Gaussian priors. In *Pushing the Limits of Contemporary Statistics: Contributions in Honor of Jayanta K. Ghosh*, volume 3 of *Inst. Math. Stat. Collect.*, pages 200–222. Inst. Math. Statist., Beachwood, OH, 2008b.

A. W. van der Vaart and J. H. van Zanten. Adaptive Bayesian estimation using a Gaussian random field with inverse gamma bandwidth. *Ann. Statist.*, 37(5B):2655–2675, 2009.

A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer Series in Statistics. Springer-Verlag, New York, 1996.

G. Wahba. Improper priors, spline smoothing and the problem of guarding against model errors in regression. *J. Roy. Statist. Soc. Ser. B*, 40(3): 364–372, 1978.

Journal of Machine Learning Research 12 (2011) 2121-2159 Submitted 3/10; Revised 3/11; Published 7/11

Adaptive Subgradient Methods for Online Learning and Stochastic Optimization[*]

John Duchi JDUCHI@CS.BERKELEY.EDU
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720 USA

Elad Hazan EHAZAN@IE.TECHNION.AC.IL
Technion - Israel Institute of Technology
Technion City
Haifa, 32000, Israel

Yoram Singer SINGER@GOOGLE.COM
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA

Editor: Tong Zhang

Abstract

We present a new family of subgradient methods that dynamically incorporate knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning. Metaphorically, the adaptation allows us to find needles in haystacks in the form of very predictive but rarely seen features. Our paradigm stems from recent advances in stochastic optimization and online learning which employ proximal functions to control the gradient steps of the algorithm. We describe and analyze an apparatus for adaptively modifying the proximal function, which significantly simplifies setting a learning rate and results in regret guarantees that are provably as good as the best proximal function that can be chosen in hindsight. We give several efficient algorithms for empirical risk minimization problems with common and important regularization functions and domain constraints. We experimentally study our theoretical analysis and show that adaptive subgradient methods outperform state-of-the-art, yet non-adaptive, subgradient algorithms.

Keywords: subgradient methods, adaptivity, online learning, stochastic convex optimization

1. Introduction

In many applications of online and stochastic learning, the input instances are of very high dimension, yet within any particular instance only a few features are non-zero. It is often the case, however, that infrequently occurring features are highly informative and discriminative. The informativeness of rare features has led practitioners to craft domain-specific feature weightings, such as TF-IDF (Salton and Buckley, 1988), which pre-emphasize infrequently occurring features. We use this old idea as a motivation for applying modern learning-theoretic techniques to the problem of online and stochastic learning, focusing concretely on (sub)gradient methods.

$*$. A preliminary version of this work was published in COLT 2010.

Standard stochastic subgradient methods largely follow a predetermined procedural scheme that is oblivious to the characteristics of the data being observed. In contrast, our algorithms dynamically incorporate knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning. Informally, our procedures give frequently occurring features very low learning rates and infrequent features high learning rates, where the intuition is that each time an infrequent feature is seen, the learner should "take notice." Thus, the adaptation facilitates finding and identifying very predictive but comparatively rare features.

1.1 The Adaptive Gradient Algorithm

Before introducing our adaptive gradient algorithm, which we term ADAGRAD, we establish notation. Vectors and scalars are lower case italic letters, such as $x \in \mathcal{X}$. We denote a sequence of vectors by subscripts, that is, $x_t, x_{t+1}, \ldots$, and entries of each vector by an additional subscript, for example, $x_{t,j}$. The subdifferential set of a function f evaluated at x is denoted $\partial f(x)$, and a particular vector in the subdifferential set is denoted by $f'(x) \in \partial f(x)$ or $g_t \in \partial f_t(x_t)$. When a function is differentiable, we write $\nabla f(x)$. We use $\langle x, y\rangle$ to denote the inner product between x and y. The Bregman divergence associated with a strongly convex and differentiable function ψ is

$$B_\psi(x,y) = \psi(x) - \psi(y) - \langle \nabla\psi(y), x - y\rangle \ .$$

We also make frequent use of the following two matrices. Let $g_{1:t} = [g_1 \ \cdots \ g_t]$ denote the matrix obtained by concatenating the subgradient sequence. We denote the ith row of this matrix, which amounts to the concatenation of the ith component of each subgradient we observe, by $g_{1:t,i}$. We also define the outer product matrix $G_t = \sum_{\tau=1}^t g_\tau g_\tau^\top$.

Online learning and stochastic optimization are closely related and basically interchangeable (Cesa-Bianchi et al., 2004). In order to keep our presentation simple, we confine our discussion and algorithmic descriptions to the online setting with the regret bound model. In online learning, the learner repeatedly predicts a point $x_t \in \mathcal{X} \subseteq \mathbb{R}^d$, which often represents a weight vector assigning importance values to various features. The learner's goal is to achieve low regret with respect to a static predictor x^* in the (closed) convex set $\mathcal{X} \subseteq \mathbb{R}^d$ (possibly $\mathcal{X} = \mathbb{R}^d$) on a sequence of functions $f_t(x)$, measured as

$$R(T) = \sum_{t=1}^T f_t(x_t) - \inf_{x\in X} \sum_{t=1}^T f_t(x) \ .$$

At every timestep t, the learner receives the (sub)gradient information $g_t \in \partial f_t(x_t)$. Standard subgradient algorithms then move the predictor x_t in the opposite direction of g_t while maintaining $x_{t+1} \in \mathcal{X}$ via the projected gradient update (e.g., Zinkevich, 2003)

$$x_{t+1} = \Pi_{\mathcal{X}}\left(x_t - \eta g_t\right) = \operatorname*{argmin}_{x\in\mathcal{X}} \|x - (x_t - \eta g_t)\|_2^2 \ .$$

In contrast, let the Mahalanobis norm $\|\cdot\|_A = \sqrt{\langle \cdot, A\cdot\rangle}$ and denote the projection of a point y onto $\mathcal{X}$ according to A by $\Pi_{\mathcal{X}}^A(y) = \operatorname{argmin}_{x\in\mathcal{X}} \|x - y\|_A = \operatorname{argmin}_{x\in\mathcal{X}} \langle x - y, A(x-y)\rangle$. Using this notation, our generalization of standard gradient descent employs the update

$$x_{t+1} = \Pi_{\mathcal{X}}^{G_t^{1/2}}\left(x_t - \eta G_t^{-1/2} g_t\right) \ .$$

The above algorithm is computationally impractical in high dimensions since it requires computation of the root of the matrix G_t, the outer product matrix. Thus we specialize the update to

$$x_{t+1} = \Pi_{\mathcal{X}}^{\text{diag}(G_t)^{1/2}}\left(x_t - \eta \,\text{diag}(G_t)^{-1/2} g_t\right) . \tag{1}$$

Both the inverse and root of $\text{diag}(G_t)$ can be computed in linear time. Moreover, as we discuss later, when the gradient vectors are sparse the update above can often be performed in time proportional to the support of the gradient. We now elaborate and give a more formal discussion of our setting.

In this paper we consider several different online learning algorithms and their stochastic convex optimization counterparts. Formally, we consider online learning with a sequence of composite functions ϕ_t. Each function is of the form $\phi_t(x) = f_t(x) + \varphi(x)$ where f_t and φ are (closed) convex functions. In the learning settings we study, f_t is either an instantaneous loss or a stochastic estimate of the objective function in an optimization task. The function φ serves as a fixed regularization function and is typically used to control the complexity of x. At each round the algorithm makes a prediction $x_t \in \mathcal{X}$ and then receives the function f_t. We define the regret with respect to the fixed (optimal) predictor x^* as

$$R_\phi(T) \triangleq \sum_{t=1}^{T} [\phi_t(x_t) - \phi_t(x^*)] = \sum_{t=1}^{T} [f_t(x_t) + \varphi(x_t) - f_t(x^*) - \varphi(x^*)] \ . \tag{2}$$

Our goal is to devise algorithms which are guaranteed to suffer asymptotically sub-linear regret, namely, $R_\phi(T) = o(T)$.

Our analysis applies to related, yet different, methods for minimizing the regret (2). The first is Nesterov's primal-dual subgradient method (2009), and in particular Xiao's (2010) extension, regularized dual averaging, and the follow-the-regularized-leader (FTRL) family of algorithms (see for instance Kalai and Vempala, 2003; Hazan et al., 2006). In the primal-dual subgradient method the algorithm makes a prediction x_t on round t using the average gradient $\bar{g}_t = \frac{1}{t}\sum_{\tau=1}^{t} g_\tau$. The update encompasses a trade-off between a gradient-dependent linear term, the regularizer φ, and a strongly-convex term ψ_t for well-conditioned predictions. Here ψ_t is the *proximal* term. The update amounts to solving

$$x_{t+1} = \underset{x\in\mathcal{X}}{\text{argmin}} \left\{ \eta \left\langle \bar{g}_t, x \right\rangle + \eta\varphi(x) + \frac{1}{t}\psi_t(x) \right\} \ , \tag{3}$$

where η is a fixed step-size and $x_1 = \text{argmin}_{x\in\mathcal{X}} \varphi(x)$. The second method similarly has numerous names, including proximal gradient, forward-backward splitting, and composite mirror descent (Tseng, 2008; Duchi et al., 2010). We use the term composite mirror descent. The composite mirror descent method employs a more immediate trade-off between the current gradient g_t, φ, and staying close to x_t using the proximal function ψ,

$$x_{t+1} = \underset{x\in\mathcal{X}}{\text{argmin}} \left\{ \eta \left\langle g_t, x \right\rangle + \eta\varphi(x) + B_{\psi_t}(x, x_t) \right\} \ . \tag{4}$$

Our work focuses on temporal adaptation of the proximal function in a data driven way, while previous work simply sets $\psi_t \equiv \psi$, $\psi_t(\cdot) = \sqrt{t}\psi(\cdot)$, or $\psi_t(\cdot) = t\psi(\cdot)$ for some fixed ψ.

We provide formal analyses equally applicable to the above two updates and show how to automatically choose the function ψ_t so as to achieve asymptotically small regret. We describe and analyze two algorithms. Both algorithms use squared Mahalanobis norms as their proximal functions, setting $\psi_t(x) = \langle x, H_t x\rangle$ for a symmetric matrix $H_t \succeq 0$. The first uses diagonal matrices while the second constructs full dimensional matrices. Concretely, for some small fixed $\delta \geq 0$ (specified later, though in practice δ can be set to 0) we set

$$H_t = \delta I + \operatorname{diag}(G_t)^{1/2} \;\; \text{(Diagonal)} \quad \text{and} \quad H_t = \delta I + G_t^{1/2} \;\; \text{(Full)} \,. \tag{5}$$

Plugging the appropriate matrix from the above equation into ψ_t in (3) or (4) gives rise to our ADAGRAD family of algorithms. Informally, we obtain algorithms which are similar to second-order gradient descent by constructing approximations to the Hessian of the functions f_t, though we use roots of the matrices.

1.2 Outline of Results

We now outline our results, deferring formal statements of the theorems to later sections. Recall the definitions of $g_{1:t}$ as the matrix of concatenated subgradients and G_t as the outer product matrix in the prequel. The ADAGRAD algorithm with full matrix divergences entertains bounds of the form

$$R_\phi(T) = O\left(\|x^*\|_2 \operatorname{tr}(G_T^{1/2})\right) \quad \text{and} \quad R_\phi(T) = O\left(\max_{t \leq T} \|x_t - x^*\|_2 \operatorname{tr}(G_T^{1/2})\right).$$

We further show that

$$\operatorname{tr}\left(G_T^{1/2}\right) = d^{1/2} \sqrt{\inf_S \left\{ \sum_{t=1}^T \langle g_t, S^{-1} g_t \rangle \;:\; S \succeq 0, \operatorname{tr}(S) \leq d \right\}} \,.$$

These results are formally given in Theorem 7 and its corollaries. When our proximal function $\psi_t(x) = \langle x, \operatorname{diag}(G_t)^{1/2} x \rangle$ we have bounds attainable in time at most linear in the dimension d of our problems of the form

$$R_\phi(T) = O\left(\|x^*\|_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2\right) \quad \text{and} \quad R_\phi(T) = O\left(\max_{t \leq T} \|x_t - x^*\|_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2\right).$$

Similar to the above, we will show that

$$\sum_{i=1}^d \|g_{1:T,i}\|_2 = d^{1/2} \sqrt{\inf_s \left\{ \sum_{t=1}^T \langle g_t, \operatorname{diag}(s)^{-1} g_t \rangle \;:\; s \succeq 0, \langle \mathbb{1}, s \rangle \leq d \right\}} \,.$$

We formally state the above two regret bounds in Theorem 5 and its corollaries.

Following are a simple example and corollary to Theorem 5 to illustrate one regime in which we expect substantial improvements (see also the next subsection). Let $\varphi \equiv 0$ and consider Zinkevich's online gradient descent algorithm. Given a compact convex set $\mathcal{X} \subseteq \mathbb{R}^d$ and sequence of convex functions f_t, Zinkevich's algorithm makes the sequence of predictions $x_1, \ldots, x_T$ with $x_{t+1} = \Pi_{\mathcal{X}}(x_t - (\eta/\sqrt{t}) g_t)$. If the diameter of $\mathcal{X}$ is bounded, thus $\sup_{x,y \in \mathcal{X}} \|x - y\|_2 \leq D_2$, then online gradient descent, with the optimal choice in *hindsight* for the stepsize η (see the bound (7) in Section 1.4), achieves a regret bound of

$$\sum_{t=1}^T f_t(x_t) - \inf_{x \in \mathcal{X}} \sum_{t=1}^T f_t(x) \leq \sqrt{2} D_2 \sqrt{\sum_{t=1}^T \|g_t\|_2^2} \,. \tag{6}$$

When $\mathcal{X}$ is bounded via $\sup_{x,y \in \mathcal{X}} \|x - y\|_\infty \leq D_\infty$, the following corollary is a simple consequence of our Theorem 5.

Corollary 1 *Let the sequence $\{x_t\} \subset \mathbb{R}^d$ be generated by the update (4) and assume that $\max_t \|x^* - x_t\|_\infty \le D_\infty$. Using stepsize $\eta = D_\infty/\sqrt{2}$, for any x^*, the following bound holds.*

$$R_\phi(T) \le \sqrt{2d} D_\infty \sqrt{\inf_{s \succeq 0, \langle 1, s\rangle \le d} \sum_{t=1}^T \|g_t\|^2_{\operatorname{diag}(s)^{-1}}} = \sqrt{2} D_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2 \ .$$

The important feature of the bound above is the infimum under the square root, which allows us to perform better than simply using the identity matrix, and the fact that the stepsize is easy to set a priori. For example, if the set $\mathcal{X} = \{x : \|x\|_\infty \le 1\}$, then $D_2 = 2\sqrt{d}$ while $D_\infty = 2$, which suggests that if we are learning a dense predictor over a box, the adaptive method should perform well. Indeed, in this case we are guaranteed that the bound in Corollary 1 is better than (6) as the identity matrix belongs to the set over which we take the infimum.

To conclude the outline of results, we would like to point to two relevant research papers. First, Zinkevich's regret bound is tight and cannot be improved in a minimax sense (Abernethy et al., 2008). Therefore, improving the regret bound requires further reasonable assumptions on the input space. Second, in a independent work, performed concurrently to the research presented in this paper, McMahan and Streeter (2010) study *competitive ratios*, showing guaranteed improvements of the above bounds relative to families of online algorithms.

1.3 Improvements and Motivating Example

As mentioned in the prequel, we expect our adaptive methods to outperform standard online learning methods when the gradient vectors are sparse. We give empirical evidence supporting the improved performance of the adaptive methods in Section 6. Here we give a few abstract examples that show that for sparse data (input sequences where g_t has many zeros) the adaptive methods herein have better performance than non-adaptive methods. In our examples we use the hinge loss, that is,

$$f_t(x) = [1 - y_t \langle z_t, x\rangle]_+ \ ,$$

where y_t is the label of example t and $z_t \in \mathbb{R}^d$ is the data vector.

For our first example, which was also given by McMahan and Streeter (2010), consider the following sparse random data scenario, where the vectors $z_t \in \{-1, 0, 1\}^d$. Assume that at in each round t, feature i appears with probability $p_i = \min\{1, ci^{-\alpha}\}$ for some $\alpha \in (1, \infty)$ and a dimension-independent constant c. Then taking the expectation of the gradient terms in the bound in Corollary 1, we have

$$\mathbb{E} \sum_{i=1}^d \|g_{1:T,i}\|_2 = \sum_{i=1}^d \mathbb{E}\left[\sqrt{|\{t : |g_{t,i}| = 1\}|}\right] \le \sum_{i=1}^d \sqrt{\mathbb{E}|\{t : |g_{t,i}| = 1\}|} = \sum_{i=1}^d \sqrt{p_i T}$$

by Jensen's inequality. In the rightmost sum, we have $c\sum_{i=1}^d i^{-\alpha/2} = O(\log d)$ for $\alpha \ge 2$, and $\sum_{i=1}^d i^{-\alpha/2} = O(d^{1-\alpha/2})$ for $\alpha \in (1,2)$. If the domain $\mathcal{X}$ is a hypercube, say $\mathcal{X} = \{x : \|x\|_\infty \le 1\}$, then in Corollary 1 $D_\infty = 2$, and the regret of ADAGRAD is $O(\max\{\log d, d^{1-\alpha/2}\}\sqrt{T})$. For contrast, the standard regret bound (6) for online gradient descent has $D_2 = 2\sqrt{d}$ and $\|g_t\|_2^2 \ge 1$, yielding best case regret $O(\sqrt{dT})$. So we see that in this sparse yet heavy tailed feature setting, ADAGRAD's regret guarantee can be exponentially smaller in the dimension d than the non-adaptive regret bound.

Our remaining examples construct a sparse sequence for which there is a perfect predictor that the adaptive methods learn after d iterations, while standard online gradient descent (Zinkevich,

2003) suffers significantly higher loss. We assume the domain $\mathcal{X}$ is compact, so that for online gradient descent we set $\eta_t = \eta/\sqrt{t}$, which gives the optimal $O(\sqrt{T})$ regret (the setting of η does not matter to the adversary we construct).

1.3.1 DIAGONAL ADAPTATION

Consider the diagonal version of our proposed update (4) with $\mathcal{X} = \{x : \|x\|_\infty \le 1\}$. Evidently, we can take $D_\infty = 2$, and this choice simply results in the update $x_{t+1} = x_t - \sqrt{2}\,\mathrm{diag}(G_t)^{-1/2} g_t$ followed by projection (1) onto $\mathcal{X}$ for ADAGRAD (we use a pseudo-inverse if the inverse does not exist). Let e_i denote the ith unit basis vector, and assume that for each t, $z_t = \pm e_i$ for some i. Also let $y_t = \mathrm{sign}(\langle 1, z_t \rangle)$ so that there exists a perfect classifier $x^* = 1 \in \mathcal{X} \subset \mathbb{R}^d$. We initialize x_1 to be the zero vector. Fix some $\varepsilon > 0$, and on rounds rounds $t = 1, \ldots, \eta^2/\varepsilon^2$, set $z_t = e_1$. After these rounds, simply choose $z_t = \pm e_i$ for index $i \in \{2, \ldots, d\}$ chosen at random. It is clear that the update to parameter x_i at these iterations is different, and amounts to

$$x_{t+1} = x_t + e_i \ \ \text{ADAGRAD} \qquad x_{t+1} = \left[x_t + \frac{\eta}{\sqrt{t}} \right]_{[-1,1]^d} \ \ \text{(Gradient Descent)}\,.$$

(Here $[\cdot]_{[-1,1]^d}$ denotes the truncation of the vector to $[-1,1]^d$). In particular, after suffering $d-1$ more losses, ADAGRAD has a perfect classifier. However, on the remaining iterations gradient descent has $\eta/\sqrt{t} \le \varepsilon$ and thus evidently suffers loss at least $d/(2\varepsilon)$. Of course, for small ε, we have $d/(2\varepsilon) \gg d$. In short, ADAGRAD achieves constant regret per dimension while online gradient descent can suffer arbitrary loss (for unbounded t). It seems quite silly, then, to use a global learning rate rather than one for each feature.

Full Matrix Adaptation. We use a similar construction to the diagonal case to show a situation in which the full matrix update from (5) gives substantially lower regret than stochastic gradient descent. For full divergences we set $\mathcal{X} = \{x : \|x\|_2 \le \sqrt{d}\}$. Let $V = [v_1 \ \ldots \ v_d] \in \mathbb{R}^{d \times d}$ be an orthonormal matrix. Instead of having z_t cycle through the unit vectors, we make z_t cycle through the v_i so that $z_t = \pm v_i$. We let the label $y_t = \mathrm{sign}(\langle 1, V^\top z_t \rangle) = \mathrm{sign}\left(\sum_{i=1}^d \langle v_i, z_t \rangle \right)$. We provide an elaborated explanation in Appendix A. Intuitively, with $\psi_t(x) = \langle x, H_t x \rangle$ and H_t set to be the full matrix from (5), ADAGRAD again needs to observe each orthonormal vector v_i only once while stochastic gradient descent's loss can be made $\Omega(d/\varepsilon)$ for any $\varepsilon > 0$.

1.4 Related Work

Many successful algorithms have been developed over the past few years to minimize regret in the online learning setting. A modern view of these algorithms casts the problem as the task of following the (regularized) leader (see Rakhlin, 2009, and the references therein) or FTRL in short. Informally, FTRL methods choose the best decision in hindsight at every iteration. Verbatim usage of the FTRL approach fails to achieve low regret, however, adding a proximal[1] term to the past predictions leads to numerous low regret algorithms (Kalai and Vempala, 2003; Hazan and Kale, 2008; Rakhlin, 2009). The proximal term strongly affects the performance of the learning algorithm. Therefore, adapting the proximal function to the characteristics of the problem at hand is desirable.

Our approach is thus motivated by two goals. The first is to generalize the agnostic online learning paradigm to the meta-task of specializing an algorithm to fit a particular data set. Specifically,

1. The proximal term is also referred to as regularization in the online learning literature. We use the phrase proximal term in order to avoid confusion with the statistical regularization function φ.

we change the proximal function to achieve performance guarantees which are competitive with the best proximal term found in hindsight. The second, as alluded to earlier, is to automatically adjust the learning rates for online learning and stochastic gradient descent on a per-feature basis. The latter can be very useful when our gradient vectors g_t are sparse, for example, in a classification setting where examples may have only a small number of non-zero features. As we demonstrated in the examples above, it is rather deficient to employ exactly the same learning rate for a feature seen hundreds of times and for a feature seen only once or twice.

Our techniques stem from a variety of research directions, and as a byproduct we also extend a few well-known algorithms. In particular, we consider variants of the follow-the-regularized leader (FTRL) algorithms mentioned above, which are kin to Zinkevich's lazy projection algorithm. We use Xiao's recently analyzed regularized dual averaging (RDA) algorithm (2010), which builds upon Nesterov's (2009) primal-dual subgradient method. We also consider forward-backward splitting (FOBOS) (Duchi and Singer, 2009) and its composite mirror-descent (proximal gradient) generalizations (Tseng, 2008; Duchi et al., 2010), which in turn include as special cases projected gradients (Zinkevich, 2003) and mirror descent (Nemirovski and Yudin, 1983; Beck and Teboulle, 2003). Recent work by several authors (Nemirovski et al., 2009; Juditsky et al., 2008; Lan, 2010; Xiao, 2010) considered efficient and robust methods for stochastic optimization, especially in the case when the expected objective f is smooth. It may be interesting to investigate adaptive metric approaches in smooth stochastic optimization.

The idea of adapting first order optimization methods is by no means new and can be traced back at least to the 1970s with the work on space dilation methods of Shor (1972) and variable metric methods, such as the BFGS family of algorithms (e.g., Fletcher, 1970). This prior work often assumed that the function to be minimized was differentiable and, to our knowledge, did not consider stochastic, online, or composite optimization. In her thesis, Nedić (2002) studied variable metric subgradient methods, though it seems difficult to derive explicit rates of convergence from the results there, and the algorithms apply only when the constraint set $\mathcal{X} = \mathbb{R}^d$. More recently, Bordes et al. (2009) proposed a Quasi-Newton stochastic gradient-descent procedure, which is similar in spirit to our methods. However, their convergence results assume a smooth objective with positive definite Hessian bounded away from 0. Our results apply more generally.

Prior to the analysis presented in this paper for online and stochastic optimization, the strongly convex function ψ in the update equations (3) and (4) either remained intact or was simply multiplied by a time-dependent scalar throughout the run of the algorithm. Zinkevich's projected gradient, for example, uses $\psi_t(x) = \|x\|_2^2$, while RDA (Xiao, 2010) employs $\psi_t(x) = \sqrt{t}\psi(x)$ where ψ is a strongly convex function. The bounds for both types of algorithms are similar, and both rely on the norm $\|\cdot\|$ (and its associated dual $\|\cdot\|_*$) with respect to which ψ is strongly convex. Mirror-descent type first order algorithms, such as projected gradient methods, attain regret bounds of the form (Zinkevich, 2003; Bartlett et al., 2007; Duchi et al., 2010)

$$R_\phi(T) \le \frac{1}{\eta} B_\psi(x^*, x_1) + \frac{\eta}{2} \sum_{t=1}^{T} \left\| f_t'(x_t) \right\|_*^2 . \tag{7}$$

Choosing $\eta \propto 1/\sqrt{T}$ gives $R_\phi(T) = O(\sqrt{T})$. When $B_\psi(x, x^*)$ is bounded for all $x \in \mathcal{X}$, we choose step sizes $\eta_t \propto 1/\sqrt{t}$ which is equivalent to setting $\psi_t(x) = \sqrt{t}\psi(x)$. Therefore, no assumption on the time horizon is necessary. For RDA and follow-the-leader algorithms, the bounds are similar

(Xiao, 2010, Theorem 3):

$$R_\phi(T) \le \sqrt{T}\psi(x^*) + \frac{1}{2\sqrt{T}} \sum_{t=1}^{T} \left\| f_t'(x_t) \right\|_*^2 . \tag{8}$$

The problem of adapting to data and obtaining tighter data-dependent bounds for algorithms such as those above is a natural one and has been studied in the mistake-bound setting for online learning in the past. A framework that is somewhat related to ours is the confidence weighted learning scheme by Crammer et al. (2008) and the adaptive regularization of weights algorithm (AROW) of Crammer et al. (2009). These papers provide mistake-bound analyses for second-order algorithms, which in turn are similar in spirit to the second-order Perceptron algorithm (Cesa-Bianchi et al., 2005). The analyses by Crammer and colleagues, however, yield mistake bounds dependent on the runs of the individual algorithms and are thus difficult to compare with our regret bounds.

AROW maintains a mean prediction vector $\mu_t \in \mathbb{R}^d$ and a covariance matrix $\Sigma_t \in \mathbb{R}^{d\times d}$ over μ_t as well. At every step of the algorithm, the learner receives a pair (z_t, y_t) where $z_t \in \mathbb{R}^d$ is the tth example and $y_t \in \{-1, +1\}$ is the label. Whenever the predictor μ_t attains a margin value smaller than 1, AROW performs the update

$$\beta_t = \frac{1}{\langle z_t, \Sigma_t z_t\rangle + \lambda}, \quad \alpha_t = [1 - y_t \langle z_t, \mu_t\rangle]_+ ,$$
$$\mu_{t+1} = \mu_t + \alpha_t \Sigma_t y_t z_t, \quad \Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t x_t x_t^\top \Sigma_t. \tag{9}$$

In the above scheme, one can force Σ_t to be diagonal, which reduces the run-time and storage requirements of the algorithm but still gives good performance (Crammer et al., 2009). In contrast to AROW, the ADAGRAD algorithm uses the *root* of the inverse covariance matrix, a consequence of our formal analysis. Crammer et al.'s algorithm and our algorithms have similar run times, generally linear in the dimension d, when using diagonal matrices. However, when using full matrices the runtime of AROW algorithm is $O(d^2)$, which is faster than ours as it requires computing the root of a matrix.

In concurrent work, McMahan and Streeter (2010) propose and analyze an algorithm which is very similar to some of the algorithms presented in this paper. Our analysis builds on recent advances in online learning and stochastic optimization (Duchi et al., 2010; Xiao, 2010), whereas McMahan and Streeter use first-principles to derive their regret bounds. As a consequence of our approach, we are able to apply our analysis to algorithms for composite minimization with a known additional objective term φ. We are also able to generalize and analyze both the mirror descent and dual-averaging family of algorithms. McMahan and Streeter focus on what they term the *competitive ratio*, which is the ratio of the worst case regret of the adaptive algorithm to the worst case regret of a non-adaptive algorithm with the best proximal term ψ chosen in hindsight. We touch on this issue briefly in the sequel, but refer the interested reader to McMahan and Streeter (2010) for this alternative elegant perspective. We believe that both analyses shed insights into the problems studied in this paper and complement each other.

There are also other lines of work on adaptive gradient methods that are not directly related to our work but nonetheless relevant. Tighter regret bounds using the variation of the cost functions f_t were proposed by Cesa-Bianchi et al. (2007) and derived by Hazan and Kale (2008). Bartlett et al. (2007) explore another adaptation technique for η_t where they adapt the step size to accommodate both strongly and weakly convex functions. Our approach differs from previous approaches as it does not focus on a particular loss function or mistake bound. Instead, we view the problem of adapting the proximal function as a meta-learning problem. We then obtain a bound comparable to the bound obtained using the best proximal function chosen in hindsight.

2. Adaptive Proximal Functions

Examining the bounds (7) and (8), we see that most of the regret depends on dual norms of $f_t'(x_t)$, and the dual norms in turn depend on the choice of ψ. This naturally leads to the question of whether we can modify the proximal term ψ along the run of the algorithm in order to lower the contribution of the aforementioned norms. We achieve this goal by keeping second order information about the sequence f_t and allow ψ to vary on each round of the algorithms.

We begin by providing two corollaries based on previous work that give the regret of our base algorithms when the proximal function ψ_t is allowed to change. These corollaries are used in the sequel in our regret analysis. We assume that ψ_t is monotonically non-decreasing, that is, $\psi_{t+1}(x) \geq \psi_t(x)$. We also assume that ψ_t is 1-strongly convex with respect to a time-dependent semi-norm $\|\cdot\|_{\psi_t}$. Formally, ψ is 1-strongly convex with respect to $\|\cdot\|_{\psi}$ if

$$\psi(y) \geq \psi(x) + \langle \nabla\psi(x), y - x\rangle + \frac{1}{2}\|x-y\|_{\psi}^2 .$$

Strong convexity is guaranteed if and only if $B_{\psi_t}(x,y) \geq \frac{1}{2}\|x-y\|_{\psi_t}^2$. We also denote the dual norm of $\|\cdot\|_{\psi_t}$ by $\|\cdot\|_{\psi_t^*}$. For completeness, we provide the proofs of following two results in Appendix F, as they build straightforwardly on work by Duchi et al. (2010) and Xiao (2010). For the primal-dual subgradient update, the following bound holds.

Proposition 2 *Let the sequence $\{x_t\}$ be defined by the update (3). For any $x^* \in \mathcal{X}$,*

$$\sum_{t=1}^{T} f_t(x_t) + \varphi(x_t) - f_t(x^*) - \varphi(x^*) \leq \frac{1}{\eta}\psi_T(x^*) + \frac{\eta}{2}\sum_{t=1}^{T}\left\|f_t'(x_t)\right\|_{\psi_{t-1}^*}^2 . \tag{10}$$

For composite mirror descent algorithms a similar result holds.

Proposition 3 *Let the sequence $\{x_t\}$ be defined by the update (4). Assume w.l.o.g. that $\varphi(x_1) = 0$. For any $x^* \in \mathcal{X}$,*

$$\begin{aligned}&\sum_{t=1}^{T} f_t(x_t) + \varphi(x_t) - f_t(x^*) - \varphi(x^*)\\ &\leq \frac{1}{\eta}B_{\psi_1}(x^*, x_1) + \frac{1}{\eta}\sum_{t=1}^{T-1}\left[B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1})\right] + \frac{\eta}{2}\sum_{t=1}^{T}\left\|f_t'(x_t)\right\|_{\psi_t^*}^2 .\end{aligned} \tag{11}$$

The above corollaries allow us to prove regret bounds for a family of algorithms that iteratively modify the proximal functions ψ_t in attempt to lower the regret bounds.

```
INPUT: $\eta > 0, \delta \geq 0$
VARIABLES: $s \in \mathbb{R}^d, H \in \mathbb{R}^{d \times d}, g_{1:t,i} \in \mathbb{R}^t$ for $i \in \{1, \dots, d\}$
INITIALIZE $x_1 = 0, g_{1:0} = []$
FOR $t = 1$ to $T$
    Suffer loss $f_t(x_t)$
    Receive subgradient $g_t \in \partial f_t(x_t)$ of $f_t$ at $x_t$
    UPDATE $g_{1:t} = [g_{1:t-1}\ g_t], s_{t,i} = \|g_{1:t,i}\|_2$
    SET $H_t = \delta I + \mathrm{diag}(s_t), \psi_t(x) = \frac{1}{2}\langle x, H_t x\rangle$

    Primal-Dual Subgradient Update (3):
    $x_{t+1} = \operatorname*{argmin}_{x \in X} \left\{ \eta \left\langle \frac{1}{t} \sum_{\tau=1}^{t} g_\tau, x \right\rangle + \eta\varphi(x) + \frac{1}{t}\psi_t(x) \right\}.$

    Composite Mirror Descent Update (4):
    $x_{t+1} = \operatorname*{argmin}_{x \in X} \left\{ \eta\langle g_t, x\rangle + \eta\varphi(x) + B_{\psi_t}(x, x_t) \right\}.$
```

Figure 1: ADAGRAD with diagonal matrices

3. Diagonal Matrix Proximal Functions

We begin by restricting ourselves to using diagonal matrices to define matrix proximal functions and (semi)norms. This restriction serves a two-fold purpose. First, the analysis for the general case is somewhat complicated and thus the analysis of the diagonal restriction serves as a proxy for better understanding. Second, in problems with high dimension where we expect this type of modification to help, maintaining more complicated proximal functions is likely to be prohibitively expensive. Whereas earlier analysis requires a learning rate to slow changes between predictors x_t and x_{t+1}, we will instead automatically grow the proximal function we use to achieve asymptotically low regret. To remind the reader, $g_{1:t,i}$ is the ith row of the matrix obtained by concatenating the subgradients from iteration 1 through t in the online algorithm.

To provide some intuition for the algorithm we show in Algorithm 1, let us examine the problem

$$\min_s \sum_{t=1}^{T} \sum_{i=1}^{d} \frac{g_{t,i}^2}{s_i} \quad \text{s.t. } s \succeq 0,\ \langle 1, s\rangle \leq c\ .$$

This problem is solved by setting $s_i = \|g_{1:T,i}\|_2$ and scaling s so that $\langle s, 1\rangle = c$. To see this, we can write the Lagrangian of the minimization problem by introducing multipliers $\lambda \succeq 0$ and $\theta \geq 0$ to get

$$\mathcal{L}(s, \lambda, \theta) = \sum_{i=1}^{d} \frac{\|g_{1:T,i}\|_2^2}{s_i} - \langle \lambda, s\rangle + \theta(\langle 1, s\rangle - c).$$

Taking partial derivatives to find the infimum of $\mathcal{L}$, we see that $-\|g_{1:T,i}\|_2^2 / s_i^2 - \lambda_i + \theta = 0$, and complementarity conditions on $\lambda_i s_i$ (Boyd and Vandenberghe, 2004) imply that $\lambda_i = 0$. Thus we have $s_i = \theta^{-\frac{1}{2}} \|g_{1:T,i}\|_2$, and normalizing appropriately using θ gives that $s_i = c\,\|g_{1:T,i}\|_2 / \sum_{j=1}^{d} \|g_{1:T,j}\|_2$.

As a final note, we can plug s_i into the objective above to see

$$\inf_s \left\{ \sum_{t=1}^T \sum_{i=1}^d \frac{g_{t,i}^2}{s_i} \ : \ s \succeq 0, \langle 1, s \rangle \le c \right\} = \frac{1}{c} \left(\sum_{i=1}^d \|g_{1:T,i}\|_2 \right)^2 . \tag{12}$$

Let $\operatorname{diag}(v)$ denote the diagonal matrix with diagonal v. It is natural to suspect that for s achieving the infimum in Equation (12), if we use a proximal function similar to $\psi(x) = \langle x, \operatorname{diag}(s) x \rangle$ with associated squared dual norm $\|x\|_{\psi^*}^2 = \langle x, \operatorname{diag}(s)^{-1} x \rangle$, we should do well lowering the gradient terms in the regret bounds (10) and (11).

To prove a regret bound for our Algorithm 1, we note that both types of updates suffer losses that include a term depending solely on the gradients obtained along their run. The following lemma is applicable to both updates, and was originally proved by Auer and Gentile (2000), though we provide a proof in Appendix C. McMahan and Streeter (2010) also give an identical lemma.

Lemma 4 *Let $g_t = f_t'(x_t)$ and $g_{1:t}$ and s_t be defined as in Algorithm 1. Then*

$$\sum_{t=1}^T \langle g_t, \operatorname{diag}(s_t)^{-1} g_t \rangle \le 2 \sum_{i=1}^d \|g_{1:T,i}\|_2 .$$

To obtain a regret bound, we need to consider the terms consisting of the dual-norm of the subgradient in the regret bounds (10) and (11), which is $\|f_t'(x_t)\|_{\psi_t^*}^2$. When $\psi_t(x) = \langle x, (\delta I + \operatorname{diag}(s_t)) x \rangle$, it is easy to see that the associated dual-norm is

$$\|g\|_{\psi_t^*}^2 = \langle g, (\delta I + \operatorname{diag}(s_t))^{-1} g \rangle .$$

From the definition of s_t in Algorithm 1, we clearly have $\|f_t'(x_t)\|_{\psi_t^*}^2 \le \langle g_t, \operatorname{diag}(s_t)^{-1} g_t \rangle$. Note that if $s_{t,i} = 0$ then $g_{t,i} = 0$ by definition of $s_{t,i}$. Thus, for any $\delta \ge 0$, Lemma 4 implies

$$\sum_{t=1}^T \|f_t'(x_t)\|_{\psi_t^*}^2 \le 2 \sum_{i=1}^d \|g_{1:T,i}\|_2 . \tag{13}$$

To obtain a bound for a primal-dual subgradient method, we set $\delta \ge \max_t \|g_t\|_\infty$, in which case $\|g_t\|_{\psi_{t-1}^*}^2 \le \langle g_t, \operatorname{diag}(s_t)^{-1} g_t \rangle$, and we follow the same lines of reasoning to achieve the inequality (13).

It remains to bound the various Bregman divergence terms for Corollary 3 and the term $\psi_T(x^*)$ for Corollary 2. We focus first on the composite mirror-descent update. Examining the bound (11) and Algorithm 1, we notice that

$$\begin{aligned} B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1}) &= \frac{1}{2} \langle x^* - x_{t+1}, \operatorname{diag}(s_{t+1} - s_t)(x^* - x_{t+1}) \rangle \\ &\le \frac{1}{2} \max_i (x_i^* - x_{t+1,i})^2 \|s_{t+1} - s_t\|_1 . \end{aligned}$$

Since $\|s_{t+1} - s_t\|_1 = \langle s_{t+1} - s_t, 1 \rangle$ and $\langle s_T, 1 \rangle = \sum_{i=1}^d \|g_{1:T,i}\|_2$, we have

$$\begin{aligned} \sum_{t=1}^{T-1} B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1}) \ &\le \ \frac{1}{2} \sum_{t=1}^{T-1} \|x^* - x_{t+1}\|_\infty^2 \langle s_{t+1} - s_t, 1 \rangle \\ &\le \ \frac{1}{2} \max_{t \le T} \|x^* - x_t\|_\infty^2 \sum_{i=1}^d \|g_{1:T,i}\|_2 - \frac{1}{2} \|x^* - x_1\|_\infty^2 \langle s_1, 1 \rangle \ . \end{aligned} \tag{14}$$

We also have

$$\psi_T(x^*) = \delta \|x^*\|_2^2 + \langle x^*, \mathrm{diag}(s_T)x^*\rangle \le \delta \|x^*\|_2^2 + \|x^*\|_\infty^2 \sum_{i=1}^d \|g_{1:T,i}\|_2 .$$

Combining the above arguments with Corollaries 2 and 3, and using (14) with the fact that $B_{\psi_1}(x^*, x_1) \le \frac{1}{2}\|x^* - x_1\|_\infty^2 \langle 1, s_1\rangle$, we have proved the following theorem.

Theorem 5 *Let the sequence $\{x_t\}$ be defined by Algorithm 1. For x_t generated using the primal-dual subgradient update (3) with $\delta \ge \max_t \|g_t\|_\infty$, for any $x^* \in \mathcal{X}$,*

$$R_\phi(T) \le \frac{\delta}{\eta}\|x^*\|_2^2 + \frac{1}{\eta}\|x^*\|_\infty^2 \sum_{i=1}^d \|g_{1:T,i}\|_2 + \eta \sum_{i=1}^d \|g_{1:T,i}\|_2 .$$

For x_t generated using the composite mirror-descent update (4), for any $x^* \in \mathcal{X}$

$$R_\phi(T) \le \frac{1}{2\eta}\max_{t\le T}\|x^* - x_t\|_\infty^2 \sum_{i=1}^d \|g_{1:T,i}\|_2 + \eta \sum_{i=1}^d \|g_{1:T,i}\|_2 .$$

The above theorem is a bit unwieldy. We thus perform a few algebraic simplifications to get the next corollary, which has a more intuitive form. Let us assume that $\mathcal{X}$ is compact and set $D_\infty = \sup_{x\in\mathcal{X}} \|x - x^*\|_\infty$. Furthermore, define

$$\gamma_T \triangleq \sum_{i=1}^d \|g_{1:T,i}\|_2 = \inf_s \left\{ \sum_{t=1}^T \left\langle g_t, \mathrm{diag}(s)^{-1} g_t \right\rangle : \langle 1, s\rangle \le \sum_{i=1}^d \|g_{1:T,i}\|_2 ,\ s \succeq 0 \right\} .$$

Also w.l.o.g. let $0 \in \mathcal{X}$. The following corollary is immediate (this is equivalent to Corollary 1, though we have moved the $\sqrt{d}$ term in the earlier bound).

Corollary 6 *Assume that D_∞ and γ_T are defined as above. For $\{x_t\}$ generated by Algorithm 1 using the primal-dual subgradient update (3) with $\eta = \|x^*\|_\infty$, for any $x^* \in \mathcal{X}$ we have*

$$R_\phi(T) \le 2\|x^*\|_\infty \gamma_T + \delta \frac{\|x^*\|_2^2}{\|x^*\|_\infty} \le 2\|x^*\|_\infty \gamma_T + \delta \|x^*\|_1 .$$

Using the composite mirror descent update (4) to generate $\{x_t\}$ and setting $\eta = D_\infty/\sqrt{2}$, we have

$$R_\phi(T) \le \sqrt{2} D_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2 = \sqrt{2} D_\infty \gamma_T .$$

We now give a short derivation of Corollary 1 from the introduction: use Theorem 5, Corollary 6, and the fact that

$$\inf_s \left\{ \sum_{t=1}^T \sum_{i=1}^d \frac{g_{t,i}^2}{s_i} \ :\ s \succeq 0, \langle 1, s\rangle \le d \right\} = \frac{1}{d}\left(\sum_{i=1}^d \|g_{1:T,i}\|_2 \right)^2 .$$

as in (12) in the beginning of Section 3. Plugging the γ_T term in from Corollary 6 and multiplying D_∞ by $\sqrt{d}$ completes the proof of the corollary.

As discussed in the introduction, Algorithm 1 should have lower regret than non-adaptive algorithms on sparse data, though this depends on the geometry of the underlying optimization space $\mathcal{X}$. For example, suppose that our learning problem is a logistic regression with 0/1-valued features. Then the gradient terms are likewise based on 0/1-valued features and sparse, so the gradient terms in the bound $\sum_{i=1}^{d} \|g_{1:T,i}\|_2$ should all be much smaller than $\sqrt{T}$. If some features appear much more frequently than others, then the infimal representation of γ_T and the infimal equality in Corollary 1 show that we have significantly lower regret by using higher learning rates for infrequent features and lower learning rates on commonly appearing features. Further, if the optimal predictor is relatively dense, as is often the case in predictions problems with sparse inputs, then $\|x^*\|_\infty$ is the best p-norm we can have in the regret.

More precisely, McMahan and Streeter (2010) show that if $\mathcal{X}$ is contained within an ℓ_∞ ball of radius R and contains an ℓ_∞ ball of radius r, then the bound in the above corollary is within a factor of $\sqrt{2}R/r$ of the regret of the best diagonal proximal matrix, chosen in hindsight. So, for example, if $\mathcal{X} = \{x \in \mathbb{R}^d : \|x\|_p \le C\}$, then $R/r = d^{1/p}$, which shows that the domain $\mathcal{X}$ does effect the guarantees we can give on optimality of ADAGRAD.

4. Full Matrix Proximal Functions

In this section we derive and analyze new updates when we estimate a full matrix for the divergence ψ_t instead of a diagonal one. In this generalized case, we use the root of the matrix of outer products of the gradients that we have observed to update our parameters. As in the diagonal case, we build on intuition garnered from an optimization problem, and in particular, we seek a matrix S which is the solution to the following minimization problem:

$$\min_S \sum_{t=1}^{T} \left\langle g_t, S^{-1} g_t \right\rangle \ \text{ s.t. } S \succeq 0, \ \operatorname{tr}(S) \le c \ . \tag{15}$$

The solution is obtained by defining $G_t = \sum_{\tau=1}^{t} g_\tau g_\tau^\top$ and setting S to be a normalized version of the root of G_T, that is, $S = c\, G_T^{1/2} / \operatorname{tr}(G_T^{1/2})$. For a proof, see Lemma 15 in Appendix E, which also shows that when G_T is not full rank we can instead use its pseudo-inverse. If we iteratively use divergences of the form $\psi_t(x) = \left\langle x, G_t^{1/2} x \right\rangle$, we might expect as in the diagonal case to attain low regret by collecting gradient information. We achieve our low regret goal by employing a similar doubling lemma to Lemma 4 and bounding the gradient norm terms. The resulting algorithm is given in Algorithm 2, and the next theorem provides a quantitative analysis of the brief motivation above.

Theorem 7 *Let G_t be the outer product matrix defined above and the sequence $\{x_t\}$ be defined by Algorithm 2. For x_t generated using the primal-dual subgradient update of (3) and $\delta \ge \max_t \|g_t\|_2$, for any $x^* \in \mathcal{X}$*

$$R_\phi(T) \le \frac{\delta}{\eta} \|x^*\|_2^2 + \frac{1}{\eta} \|x^*\|_2^2 \operatorname{tr}(G_T^{1/2}) + \eta \operatorname{tr}(G_T^{1/2}).$$

For x_t generated with the composite mirror-descent update of (4), if $x^* \in \mathcal{X}$ and $\delta \ge 0$

$$R_\phi(T) \le \frac{\delta}{\eta} \|x^*\|_2^2 + \frac{1}{2\eta} \max_{t \le T} \|x^* - x_t\|_2^2 \operatorname{tr}(G_T^{1/2}) + \eta \operatorname{tr}(G_T^{1/2}).$$

```
INPUT: $\eta > 0, \delta \ge 0$
VARIABLES: $S_t \in \mathbb{R}^{d\times d}, H_t \in \mathbb{R}^{d\times d}, G_t \in \mathbb{R}^{d\times d}$
INITIALIZE $x_1 = 0, S_0 = 0, H_0 = 0, G_0 = 0$
FOR $t = 1$ to $T$
    Suffer loss $f_t(x_t)$
    Receive subgradient $g_t \in \partial f_t(x_t)$ of $f_t$ at $x_t$
    UPDATE $G_t = G_{t-1} + g_t g_t^\top, S_t = G_t^{\frac{1}{2}}$
    SET $H_t = \delta I + S_t, \psi_t(x) = \frac{1}{2}\langle x, H_t x\rangle$

    Primal-Dual Subgradient Update ((3)):
      $x_{t+1} = \operatorname*{argmin}_{x\in X} \left\{ \eta \left\langle \frac{1}{t}\sum_{\tau=1}^{t} g_\tau, x \right\rangle + \eta\varphi(x) + \frac{1}{t}\psi_t(x) \right\}.$

    Composite Mirror Descent Update ((4)):
      $x_{t+1} = \operatorname*{argmin}_{x\in X} \left\{ \eta\langle g_t, x\rangle + \eta\varphi(x) + B_{\psi_t}(x, x_t) \right\}.$
```

Figure 2: ADAGRAD with full matrices

Proof To begin, we consider the difference between the divergence terms at time $t+1$ and time t from the regret (11) in Corollary 3. Let $\lambda_{\max}(M)$ denote the largest eigenvalue of a matrix M. We have

$$\begin{aligned} B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1}) &= \frac{1}{2}\left\langle x^* - x_{t+1}, ({G_{t+1}}^{1/2} - {G_t}^{1/2})(x^* - x_{t+1})\right\rangle \\ &\le \frac{1}{2}\|x^* - x_{t+1}\|_2^2 \lambda_{\max}(G_{t+1}^{1/2} - G_t^{1/2}) \le \frac{1}{2}\|x^* - x_{t+1}\|_2^2 \operatorname{tr}(G_{t+1}^{1/2} - G_t^{1/2}) \ . \end{aligned}$$

For the last inequality we used the fact that the trace of a matrix is equal to the sum of its eigenvalues along with the property ${G_{t+1}}^{1/2} - {G_t}^{1/2} \succeq 0$ (see Lemma 13 in Appendix B) and therefore $\operatorname{tr}(G_{t+1}^{1/2} - G_t^{1/2}) \ge \lambda_{\max}(G_{t+1}^{1/2} - G_t^{1/2})$. Thus, we get

$$\sum_{t=1}^{T-1} B_{\psi_{t+1}}(x^*, x_{t+1}) - B_{\psi_t}(x^*, x_{t+1}) \le \frac{1}{2}\sum_{t=1}^{T-1} \|x^* - x_{t+1}\|_2^2 \left(\operatorname{tr}(G_{t+1}^{1/2}) - \operatorname{tr}(G_t^{1/2})\right) \ .$$

Now we use the fact that G_1 is a rank 1 PSD matrix with non-negative trace to see that

$$\begin{aligned} &\sum_{t=1}^{T-1} \|x^* - x_{t+1}\|_2^2 \left(\operatorname{tr}(G_{t+1}^{1/2}) - \operatorname{tr}(G_t^{1/2})\right) \\ &\qquad \le \max_{t\le T} \|x^* - x_t\|_2^2 \operatorname{tr}({G_T}^{1/2}) - \|x^* - x_1\|_2^2 \operatorname{tr}(G_1^{1/2}) \ . \qquad (16) \end{aligned}$$

It remains to bound the gradient terms common to all our bounds. We use the following three lemmas, which essentially directly applicable. We prove the first two in Appendix D.

Lemma 8 *Let $B \succeq 0$ and $B^{-1/2}$ denote the root of the inverse of B when $B \succ 0$ and the root of the pseudo-inverse of B otherwise. For any ν such that $B - \nu gg^\top \succeq 0$ the following inequality holds.*

$$2\operatorname{tr}((B - \nu gg^\top)^{1/2}) \le 2\operatorname{tr}(B^{1/2}) - \nu \operatorname{tr}(B^{-1/2} gg^\top) \ .$$

Lemma 9 *Let* $\delta \geq \|g\|_2$ *and* $A \succeq 0$, *then* $\langle g, (\delta I + A^{1/2})^{-1} g\rangle \leq \left\langle g, \left((A + gg^\top)^\dagger\right)^{1/2} g\right\rangle$.

Lemma 10 *Let* $S_t = G_t{}^{1/2}$ *be as defined in Algorithm 2 and* $A^\dagger$ *denote the pseudo-inverse of* A. *Then*

$$\sum_{t=1}^{T} \left\langle g_t, S_t^\dagger g_t \right\rangle \leq 2 \sum_{t=1}^{T} \left\langle g_t, S_T^\dagger g_t \right\rangle = 2\operatorname{tr}(G_T{}^{1/2}) .$$

Proof We prove the lemma by induction. The base case is immediate, since we have

$$\left\langle g_1, (G_1^\dagger)^{1/2} g_1 \right\rangle = \frac{\langle g_1, g_1\rangle}{\|g_1\|_2} = \|g_1\|_2 \leq 2\|g_1\|_2 .$$

Now, assume the lemma is true for $T-1$, so from the inductive assumption we get

$$\sum_{t=1}^{T} \left\langle g_t, S_t^\dagger g_t \right\rangle \leq 2 \sum_{t=1}^{T-1} \left\langle g_t, S_{T-1}^\dagger g_t \right\rangle + \left\langle g_T, S_T^\dagger g_T \right\rangle .$$

Since S_{T-1} does not depend on t we can rewrite $\sum_{t=1}^{T-1} \left\langle g_t, S_{T-1}^\dagger g_t \right\rangle$ as

$$\operatorname{tr}\left(S_{T-1}^\dagger, \sum_{t=1}^{T-1} g_t g_t^\top \right) = \operatorname{tr}((G_{T-1}^\dagger)^{1/2} G_{T-1}) ,$$

where the right-most equality follows from the definitions of S_t and G_t. Therefore, we get

$$\begin{aligned} \sum_{t=1}^{T} \left\langle g_t, S_t^\dagger g_t \right\rangle &\leq 2\operatorname{tr}((G_{T-1}^\dagger)^{1/2} G_{T-1}) + \left\langle g_T, (G_T^\dagger)^{1/2} g_T \right\rangle \\ &= 2\operatorname{tr}(G_{T-1}^{1/2}) + \left\langle g_T, (G_T^\dagger)^{1/2} g_T \right\rangle . \end{aligned}$$

Using Lemma 8 with the substitution $B = G_T$, $\nu = 1$, and $g = g_t$ lets us exploit the concavity of the function $\operatorname{tr}(A^{1/2})$ to bound the above sum by $2\operatorname{tr}(G_T^{1/2})$. ▲

We can now finalize our proof of the theorem. As in the diagonal case, we have that the squared dual norm (seminorm when $\delta = 0$) associated with ψ_t is

$$\|x\|_{\psi_t^*}^2 = \langle x, (\delta I + S_t)^{-1} x\rangle .$$

Thus it is clear that $\|g_t\|_{\psi_t^*}^2 \leq \left\langle g_t, S_t^\dagger g_t \right\rangle$. For the dual-averaging algorithms, we use Lemma 9 above show that $\|g_t\|_{\psi_{t-1}^*}^2 \leq \left\langle g_t, S_t^\dagger g_t \right\rangle$ so long as $\delta \geq \|g_t\|_2$. Lemma 10's doubling inequality then implies that

$$\sum_{t=1}^{T} \left\|f_t'(x_t)\right\|_{\psi_t^*}^2 \leq 2\operatorname{tr}(G_T^{1/2}) \quad \text{and} \quad \sum_{t=1}^{T} \left\|f_t'(x_t)\right\|_{\psi_{t-1}^*}^2 \leq 2\operatorname{tr}(G_T^{1/2}) \tag{17}$$

for the mirror-descent and primal-dual subgradient algorithm, respectively.

To finish the proof, Note that $B_{\psi_1}(x^*, x_1) \leq \frac{1}{2}\|x^* - x_1\|_2^2 \operatorname{tr}(G_1^{1/2})$ when $\delta = 0$. By combining this with the first of the bounds (17) and the bound (16) on $\sum_{t=1}^{T-1} B_{\psi_{t+1}}(x^*, x^{t+1}) - B_{\psi_t}(x^*, x^{t+1})$, Corollary 3 gives the theorem's statement for the mirror-descent family of algorithms. Combining the fact that $\sum_{t=1}^{T} \|f_t'(x_t)\|_{\psi_{t-1}^*}^2 \le 2\operatorname{tr}(G_T^{1/2})$ and the bound (16) with Corollary 2 gives the desired bound on $R_\phi(T)$ for the primal-dual subgradient algorithms, which completes the proof of the theorem. ■

As before, we can give a corollary that simplifies the bound implied by Theorem 7. The infimal equality in the corollary uses Lemma 15 in Appendix B. The corollary underscores that for learning problems in which there is a rotation U of the space for which the gradient vectors g_t have small inner products $\langle g_t, Ug_t\rangle$ (essentially a sparse basis for the g_t) then using full-matrix proximal functions can attain significantly lower regret.

Corollary 11 *Assume that $\varphi(x_1) = 0$. Then the regret of the sequence $\{x_t\}$ generated by Algorithm 2 when using the primal-dual subgradient update with $\eta = \|x^*\|_2$ is*

$$R_\phi(T) \le 2\|x^*\|_2 \operatorname{tr}(G_T^{1/2}) + \delta\|x^*\|_2 \ .$$

Let $\mathcal{X}$ be compact set so that $\sup_{x\in\mathcal{X}} \|x - x^*\|_2 \le D$. Taking $\eta = D/\sqrt{2}$ and using the composite mirror descent update with $\delta = 0$, we have

$$R_\phi(T) \le \sqrt{2}D\operatorname{tr}(G_T^{1/2}) = \sqrt{2d}D\sqrt{\inf_S\left\{\sum_{t=1}^{T} g_t^\top S^{-1} g_t \ : \ S \succeq 0, \operatorname{tr}(S) \le d\right\}} \ .$$

5. Derived Algorithms

In this section, we derive updates using concrete regularization functions φ and settings of the domain $\mathcal{X}$ for the ADAGRAD framework. We focus on showing how to solve Equations (3) and (4) with the diagonal matrix version of the algorithms we have presented. We focus on the diagonal case for two reasons. First, the updates often take closed-form in this case and carry some intuition. Second, the diagonal case is feasible to implement in very high dimensions, whereas the full matrix version is likely to be confined to a few thousand dimensions. We also discuss how to efficiently compute the updates when the gradient vectors are sparse.

We begin by noting a simple but useful fact. Let G_t denote either the outer product matrix of gradients or its diagonal counterpart and let $H_t = \delta I + G_t^{1/2}$, as usual. Simple algebraic manipulations yield that each of the updates (3) and (4) in the prequel can be written in the following form (omitting the stepsize η):

$$x_{t+1} = \operatorname*{argmin}_{x\in\mathcal{X}} \left\{\langle u, x\rangle + \varphi(x) + \frac{1}{2}\langle x, H_t x\rangle\right\} \ . \tag{18}$$

In particular, at time t for the RDA update, we have $u = \eta t g_t$. For the composite gradient update (4),

$$\eta\langle g_t, x\rangle + \frac{1}{2}\langle x - x_t, H_t(x - x_t)\rangle = \langle \eta g_t - H_t x_t, x\rangle + \frac{1}{2}\langle x, H_t x\rangle + \frac{1}{2}\langle x_t, H_t x_t\rangle$$

so that $u = \eta g_t - H_t x_t$. We now derive algorithms for solving the general update (18). Since most of the derivations are known, we generally provide only the closed-form solutions or algorithms for the solutions in the remainder of the subsection, deferring detailed derivations to Appendix G for the interested reader.

5.1 ℓ_1-regularization

We begin by considering how to solve the minimization problems necessary for Algorithm 1 with diagonal matrix divergences and $\varphi(x) = \lambda \|x\|_1$. We consider the two updates we proposed and denote the ith diagonal element of the matrix $H_t = \delta I + \text{diag}(s_t)$ from Algorithm 1 by $H_{t,ii} = \delta + \|g_{1:t,i}\|_2$. For the primal-dual subgradient update, the solution to (3) amounts to the following simple update for $x_{t+1,i}$:

$$x_{t+1,i} = \text{sign}\left(-\bar{g}_{t,i}\right) \frac{\eta t}{H_{t,ii}} \left[|\bar{g}_{t,i}| - \lambda\right]_+ . \tag{19}$$

Comparing the update (19) to the standard dual averaging update (Xiao, 2010), which is

$$x_{t+1,i} = \text{sign}\left(-\bar{g}_{t,i}\right) \eta \sqrt{t} \left[|\bar{g}_{t,i}| - \lambda\right]_+ ,$$

it is clear that the difference distills to the step size employed for each coordinate. Our generalization of RDA yields a dedicated step size for each coordinate inversely proportional to the time-based norm of the coordinate in the sequence of gradients. Due to the normalization by this term the step size scales *linearly* with t, so when $H_{t,ii}$ is small, gradient information on coordinate i is quickly incorporated.

The composite mirror-descent update (4) has a similar form that essentially amounts to iterative shrinkage and thresholding, where the shrinkage differs per coordinate:

$$x_{t+1,i} = \text{sign}\left(x_{t,i} - \frac{\eta}{H_{t,ii}} g_{t,i}\right) \left[\left|x_{t,i} - \frac{\eta}{H_{t,ii}} g_{t,i}\right| - \frac{\lambda\eta}{H_{t,ii}}\right]_+ .$$

We compare the actual performance of the newly derived algorithms to previously studied versions in the next section.

For both updates it is clear that we can perform "lazy" computation when the gradient vectors are sparse, a frequently occurring setting when learning for instance from text corpora. Suppose that from time step t_0 through t, the ith component of the gradient is 0. Then we can evaluate the above updates on demand since $H_{t,ii}$ remains intact. For composite mirror-descent, at time t when $x_{t,i}$ is needed, we update

$$x_{t,i} = \text{sign}(x_{t_0,i}) \left[|x_{t_0,i}| - \frac{\lambda\eta}{H_{t_0,ii}}(t - t_0)\right]_+ .$$

Even simpler just in time evaluation can be performed for the the primal-dual subgradient update. Here we need to keep an unnormalized version of the average $\bar{g}_t$. Concretely, we keep track of $u_t = t\bar{g}_t = \sum_{\tau=1}^{t} g_\tau = u_{t-1} + g_t$, then use the update (19):

$$x_{t,i} = \text{sign}(-u_{t,i}) \frac{\eta t}{H_{t,ii}} \left[\frac{|u_{t,i}|}{t} - \lambda\right]_+ ,$$

where H_t can clearly be updated lazily in a similar fashion.

5.2 ℓ_1-ball Projections

We next consider the setting in which $\varphi \equiv 0$ and $\mathcal{X} = \{x : \|x\|_1 \le c\}$, for which it is straightforward to adapt efficient solutions to continuous quadratic knapsack problems (Brucker, 1984). We

```
INPUT: $v \succeq 0, a \succeq 0, c \geq 0$.
IF $\sum_i v_i \leq c$ RETURN $z^* = v$
SORT $v_i/a_i$ into $\mu = [v_{i_j}/a_{i_j}]$ s.t. $v_{i_j}/a_{i_j} \geq v_{i_{j+1}}/a_{i_{j+1}}$
SET $\rho := \max\left\{\rho : \sum_{j=1}^{\rho} a_{i_j} v_{i_j} - \frac{v_{i_\rho}}{a_{i_\rho}} \sum_{j=1}^{\rho} a_{i_j}^2 < c\right\}$
SET $\theta = \frac{\sum_{j=1}^{\rho} a_{i_j} v_{i_j} - c}{\sum_{j=1}^{\rho} a_{i_j}^2}$
RETURN $z^*$ where $z_i^* = [v_i - \theta a_i]_+$.
```

Figure 3: Project $v \succeq 0$ to $\{z : \langle a, z\rangle \leq c, z \succeq 0\}$.

use the matrix $H_t = \delta I + \operatorname{diag}(G_t)^{1/2}$ from Algorithm 1. We provide a brief derivation sketch and an $O(d \log d)$ algorithm in this section. First, we convert the problem (18) into a projection problem onto a scaled ℓ_1-ball. By making the substitutions $z = H^{1/2}x$ and $A = H^{-1/2}$, it is clear that problem (18) is equivalent to

$$\min_z \left\| z + H^{-1/2} u \right\|_2^2 \quad \text{s.t.} \quad \|Az\|_1 \leq c \; .$$

Now, by appropriate choice of $v = -H^{-1/2}u = -\eta t H_t^{-1/2} g_t$ for the primal-dual update (3) and $v = H_t^{1/2} x_t - \eta H_t^{-1/2} g_t$ for the mirror-descent update (4), we arrive at the problem

$$\min_z \frac{1}{2} \|z - v\|_2^2 \quad \text{s.t.} \quad \sum_{i=1}^{d} a_i |z_i| \leq c \; . \tag{20}$$

We can clearly recover x_{t+1} from the solution z^* to the projection (20) via $x_{t+1} = H_t^{-1/2} z^*$.

By the symmetry of the objective (20), we can assume without loss of generality that $v \succeq 0$ and constrain $z \succeq 0$, and a bit of manipulation with the Lagrangian (see Appendix G) for the problem shows that the solution z^* has the form

$$z_i^* = \begin{cases} v_i - \theta^* a_i & \text{if } v_i \geq \theta^* a_i \\ 0 & \text{otherwise} \end{cases}$$

for some $\theta^* \geq 0$. The algorithm in Figure 3 constructs the optimal θ and returns z^*.

5.3 ℓ_2 Regularization

We now turn to the case where $\varphi(x) = \lambda \|x\|_2$ while $\mathcal{X} = \mathbb{R}^d$. This type of regularization is useful for zeroing multiple weights in a group, for example in multi-task or multiclass learning (Obozinski et al., 2007). Recalling the general proximal step (18), we must solve

$$\min_x \; \langle u, x\rangle + \frac{1}{2}\langle x, Hx\rangle + \lambda \|x\|_2 \; . \tag{21}$$

There is no closed form solution for this problem, but we give an efficient bisection-based procedure for solving (21). We start by deriving the dual. Introducing a variable $z = x$, we get the equivalent problem of minimizing $\langle u, x\rangle + \frac{1}{2}\langle x, Hx\rangle + \lambda\|z\|_2$ subject to $x = z$. With Lagrange multipliers α for the equality constraint, we obtain the Lagrangian

$$\mathcal{L}(x, z, \alpha) = \langle u, x\rangle + \frac{1}{2}\langle x, Hx\rangle + \lambda \|z\|_2 + \langle \alpha, x - z\rangle \; .$$

```
INPUT: $u \in \mathbb{R}^d, H \succeq 0, \lambda > 0$.
IF $\|u\|_2 \le \lambda$
  RETURN $x = 0$
SET $v = H^{-1}u$, $\theta_{\max} = \|v\|_2/\lambda - 1/\sigma_{\min}(H)$
   $\theta_{\min} = \|v\|_2/\lambda - 1/\sigma_{\max}(H)$
WHILE $\theta_{\max} - \theta_{\min} > \varepsilon$
  SET $\theta = (\theta_{\max} + \theta_{\min})/2$, $\alpha(\theta) = -(H^{-1} + \theta I)^{-1}v$
  IF $\|\alpha(\theta)\|_2 > \lambda$
    SET $\theta_{\min} = \theta$
  ELSE
    SET $\theta_{\max} = \theta$
RETURN $x = -H^{-1}(u + \alpha(\theta))$
```

Figure 4: Minimize $\langle u,x\rangle + \frac{1}{2}\langle x,Hx\rangle + \lambda\|x\|_2$

Taking the infimum of $\mathcal{L}$ with respect to the primal variables x and z, we see that the infimum is attained at $x = -H^{-1}(u+\alpha)$. Coupled with the fact that $\inf_z \lambda\|z\|_2 - \langle\alpha,z\rangle = -\infty$ unless $\|\alpha\|_2 \le \lambda$, in which case the infimum is 0, we arrive at the dual form

$$\inf_{x,z} \mathcal{L}(x,z,\alpha) = \begin{cases} -\frac{1}{2}\left\langle u+\alpha, H^{-1}(u+\alpha)\right\rangle & \text{if } \|\alpha\|_2 \le \lambda \\ -\infty & \text{otherwise.} \end{cases}$$

Setting $v = H^{-1}u$, we further distill the dual to

$$\min_\alpha \ \langle v,\alpha\rangle + \frac{1}{2}\left\langle \alpha, H^{-1}\alpha\right\rangle \ \text{ s.t. } \|\alpha\|_2 \le \lambda\,. \tag{22}$$

We can solve problem (22) efficiently using a bisection search of its equivalent representation in Lagrange form,

$$\min_\alpha \ \langle v,\alpha\rangle + \frac{1}{2}\left\langle \alpha, H^{-1}\alpha\right\rangle + \frac{\theta}{2}\|\alpha\|_2^2\ ,$$

where $\theta > 0$ is an unknown scalar. The solution to the latter as a function of θ is clearly $\alpha(\theta) = -(H^{-1}+\theta I)^{-1}v = -(H^{-1}+\theta I)^{-1}H^{-1}u$. Since $\|\alpha(\theta)\|_2$ is monotonically decreasing in θ (consider the the eigen-decomposition of the positive definite H^{-1}), we can simply perform a bisection search over θ, checking at each point whether $\|\alpha(\theta)\|_2 \gtrless \lambda$.

To find initial upper and lower bounds on θ, we note that

$$(1/\sigma_{\max}(H)+\theta)^{-1}\|v\|_2 \le \|\alpha(\theta)\|_2 \le (1/\sigma_{\min}(H)+\theta)^{-1}\|v\|_2$$

where $\sigma_{\max}(H)$ denotes the maximum singular value of H and $\sigma_{\min}(H)$ the minimum. To guarantee $\|\alpha(\theta_{\max})\|_2 \le \lambda$, we thus set $\theta_{\max} = \|v\|_2/\lambda - 1/\sigma_{\max}(H)$. Similarly, for $\theta_{\min}$ we see that so long as $\theta \ge \|v\|_2/\lambda - 1/\sigma_{\min}(H)$ we have $\|\alpha(\theta)\|_2 \ge \lambda$. The fact that $\partial\|x\|_2 = \{z : \|z\|_2 \le 1\}$ when $x = 0$ implies that the solution for the original problem (21) is $x = 0$ if and only if $\|u\|_2 \le \lambda$. We provide pseudocode for solving (21) in Algorithm 4.

5.4 ℓ_∞ Regularization

We again let $\mathcal{X} = \mathbb{R}^d$ but now choose $\varphi(x) = \lambda \|x\|_\infty$. This type of update, similarly to ℓ_2, zeroes groups of variables, which is handy in finding structurally sparse solutions for multitask or multiclass problems. Solving the ℓ_∞ regularized problem amounts to

$$\min_x \ \langle u,x\rangle + \frac{1}{2}\langle x,Hx\rangle + \lambda \|x\|_\infty \ . \tag{23}$$

The dual of this problem is a modified ℓ_1-projection problem. As in the case of ℓ_2 regularization, we introduce an equality constrained variable $z = x$ with associated Lagrange multipliers $\alpha \in \mathbb{R}^d$ to obtain

$$\mathcal{L}(x,z,\alpha) = \langle u,x\rangle + \frac{1}{2}\langle x,Hx\rangle + \lambda \|z\|_\infty + \langle \alpha, x - z\rangle \ .$$

Performing identical manipulations to the ℓ_2 case, we take derivatives and get that $x = -H^{-1}(u+\alpha)$ and, similarly, unless $\|\alpha\|_1 \le \lambda$, $\inf_z \mathcal{L}(x,z,\alpha) = -\infty$. Thus the dual problem for (23) is

$$\max_\alpha \ -\frac{1}{2}(u+\alpha)H^{-1}(u+\alpha) \ \text{ s.t. } \ \|\alpha\|_1 \le \lambda \ .$$

When H is diagonal we can find the optimal α^* using the generalized ℓ_1-projection in Algorithm 3, then reconstruct the optimal x via $x = -H^{-1}(u+\alpha^*)$.

5.5 Mixed-norm Regularization

Finally, we combine the above results to show how to solve problems with matrix-valued inputs $X \in \mathbb{R}^{d\times k}$, where $X = [\bar{x}_1 \ \cdots \ \bar{x}_d]^\top$. We consider mixed-norm regularization, which is very useful for encouraging sparsity across several tasks (Obozinski et al., 2007). Now φ is an ℓ_1/ℓ_p norm, that is, $\varphi(X) = \lambda \sum_{i=1}^d \|\bar{x}_i\|_p$. By imposing an ℓ_1-norm over p-norms of the rows of X, entire rows are nulled at once.

When $p \in \{2,\infty\}$ and the proximal H in (18) is diagonal, the previous algorithms can be readily used to solve the mixed norm problems. We simply maintain diagonal matrix information for each of the rows x_i of X separately, then solve one of the previous updates for each row independently. We use this form of regularization in our experiments with multiclass prediction problems in the next section.

6. Experiments

We performed experiments with several real world data sets with different characteristics: the ImageNet image database (Deng et al., 2009), the Reuters RCV1 text classification data set (Lewis et al., 2004), the MNIST multiclass digit recognition problem, and the census income data set from the UCI repository (Asuncion and Newman, 2007). For uniformity across experiments, we focus on the completely online (fully stochastic) optimization setting, in which at each iteration the learning algorithm receives a single example. We measure performance using two metrics: the online loss or error and the test set performance of the predictor the learning algorithm outputs at the end of a single pass through the training data. We also give some results that show how imposing sparsity constraints (in the form of ℓ_1 and mixed-norm regularization) affects the learning algorithm's performance. One benefit of the ADAGRAD framework is its ability to straightforwardly generalize to

	RDA	FB	ADAGRAD-RDA	ADAGRAD-FB	PA	AROW
ECAT	.051 (.099)	.058 (.194)	**.044** (**.086**)	**.044** (.238)	.059	.049
CCAT	.064 (.123)	.111 (.226)	**.053** (**.105**)	**.053** (.276)	.107	.061
GCAT	.046 (.092)	.056 (.183)	**.040** (**.080**)	**.040** (.225)	.066	.044
MCAT	.037 (.074)	.056 (.146)	.035 (**.063**)	**.034** (.176)	.053	.039

Table 1: Test set error rates and proportion non-zero (in parenthesis) on Reuters RCV1.

domain constraints $\mathcal{X} \neq \mathbb{R}^d$ and arbitrary regularization functions φ, in contrast to previous adaptive online algorithms.

We experiment with RDA (Xiao, 2010), FOBOS(Duchi and Singer, 2009), adaptive RDA, adaptive FOBOS, the Passive-Aggressive (PA) algorithm (Crammer et al., 2006), and AROW (Crammer et al., 2009). To remind the reader, PA is an online learning procedure with the update

$$x_{t+1} = \underset{x}{\operatorname{argmin}} \left[1 - y_t \langle z_t, x \rangle\right]_+ + \frac{\lambda}{2} \|x - x_t\|_2^2 \ ,$$

where λ is a regularization parameter. PA's update is similar to the update employed by AROW (see (9)), but the latter maintains second order information on x. By using a representer theorem it is also possible to derive efficient updates for PA and AROW when the loss is the logistic loss, $\log(1 + \exp(-y_t \langle z_t, x_t \rangle))$. We thus we compare the above six algorithms using both hinge and logistic loss.

6.1 Text Classification

The Reuters RCV1 data set consists of a collection of approximately 800,000 text articles, each of which is assigned multiple labels. There are 4 high-level categories, Economics, Commerce, Medical, and Government (ECAT, CCAT, MCAT, GCAT), and multiple more specific categories. We focus on training binary classifiers for each of the four major categories. The input features we use are 0/1 bigram features, which, post word stemming, give data of approximately 2 million dimensions. The feature vectors are very sparse, however, and most examples have fewer than 5000 non-zero features.

We compare the twelve different algorithms mentioned in the prequel as well as variants of FOBOS and RDA with ℓ_1-regularization. We summarize the results of the ℓ_1-regularized runs as well as AROW and PA in Table 1. The results for both hinge and logistic losses are qualitatively and quantitatively very similar, so we report results only for training with the hinge loss in Table 1. Each row in the table represents the average of four different experiments in which we hold out 25% of the data for a test set and perform an online pass on the remaining 75% of the data. For RDA and FOBOS, we cross-validate the stepsize parameter η by simply running multiple passes and then choosing the output of the learner that had the fewest mistakes during training. For PA and AROW we choose λ using the same approach. We use the same regularization multiplier on the ℓ_1 term for RDA and FOBOS, selected so that RDA achieved approximately 10% non-zero predictors.

It is evident from the results presented in Table 1 that the adaptive algorithms (AROW and ADAGRAD) are far superior to non-adaptive algorithms in terms of error rate on test data. The ADAGRAD algorithms naturally incorporate sparsity as well since they are run with ℓ_1-regularization, though RDA has significantly higher sparsity levels (PA and AROW do not have any sparsity). Furthermore, although omitted from the table to avoid clutter, in *every* test with the RCV1 corpus, the

Alg.	Avg. Prec.	P@1	P@3	P@5	P@10	Prop. nonzero
ADAGRAD RDA	**0.6022**	0.8502	0.8307	0.8130	0.7811	**0.7267**
AROW	0.5813	**0.8597**	**0.8369**	**0.8165**	**0.7816**	1.0000
PA	0.5581	0.8455	0.8184	0.7957	0.7576	1.0000
RDA	0.5042	0.7496	0.7185	0.6950	0.6545	0.8996

Table 2: Test set precision for ImageNet

adaptive algorithms outperformed the non-adaptive algorithms. Moreover, both ADAGRAD-RDA and ADAGRAD-Fobos outperform AROW on all the classification tasks. Unregularized RDA and FOBOS attained similar results as did the ℓ_1-regularized variants (of course without sparsity), but we omit the results to avoid clutter and because they do not give much more understanding.

6.2 Image Ranking

ImageNet (Deng et al., 2009) consists of images organized according to the nouns in the WordNet hierarchy, where each noun is associated on average with more than 500 images collected from the web. We selected 15,000 important nouns from the hierarchy and conducted a large scale image ranking task for *each* noun. This approach is identical to the task tackled by Grangier and Bengio (2008) using the Passive-Aggressive algorithm. To solve this problem, we train 15,000 ranking machines using Grangier and Bengio's visterms features, which represent patches in an image with 79-dimensional sparse vectors. There are approximately 120 patches per image, resulting in a 10,000-dimensional feature space.

Based on the results in the previous section, we focus on four algorithms for solving this task: AROW, ADAGRAD with RDA updates and ℓ_1-regularization, vanilla RDA with ℓ_1, and Passive-Aggressive. We use the ranking hinge loss, which is $[1 - \langle x, z_1 - z_2 \rangle]_+$ when z_1 is ranked above z_2. We train a ranker x_c for each of the image classes individually, cross-validating the choice of initial stepsize for each algorithm on a small held-out set. To train an individual ranker for class c, at each step of the algorithm we randomly sample a positive image z_1 for the category c and an image z_2 from the training set (which with high probability is a negative example for class c) and perform an update on the example $z_1 - z_2$. We let each algorithm take 100,000 such steps for each image category, we train four sets of rankers with each algorithm, and the training set includes approximately 2 million images.

For evaluation, we use a distinct test set of approximately 1 million images. To evaluate a set of rankers, we iterate through all 15,000 classes in the data set. For each class we take all the positive image examples in the test set and sample 10 times as many negative image examples. Following Grangier and Bengio, we then rank the set of positive and negative images and compute precision-at-k for $k = \{1, \ldots, 10\}$ and the average precision for each category. The precision-at-k is defined as the proportion of examples ranked in the top k for a category c that actually belong to c, and the average precision is the average of the precisions at each position in which a relevant picture appears. Letting $\mathrm{Pos}(c)$ denote the positive examples for category c and $p(i)$ denote the position of the ith returned picture in list of images sorted by inner product with x_c, the average precision is

$$\frac{1}{|\mathrm{Pos}(c)|} \sum_{i=1}^{|\mathrm{Pos}(c)|} \frac{i}{p(i)} .$$

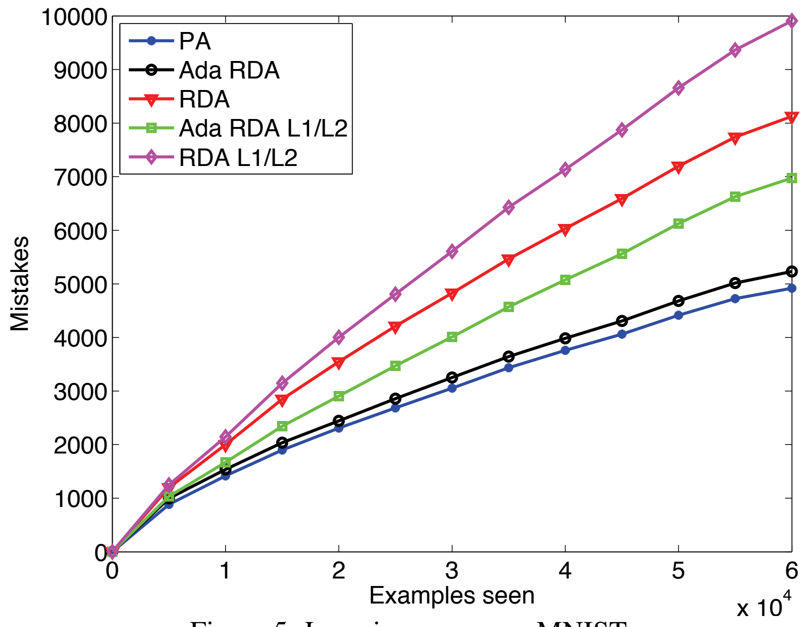


Figure 5: Learning curves on MNIST

We compute the mean of each measurement across all classes, performing this twelve times for each of the sets of rankers trained. Table 2 summarizes our results. We do not report variance as the variance was on the order of 10^{-5} for each algorithm. One apparent characteristic to note from the table is that ADAGRAD RDA achieves higher levels of sparsity than the other algorithms—using only 73% of the input features it achieves very high performance. Moreover, it outperforms all the algorithms in average precision. AROW has better results than the other algorithms in terms of precision-at-k for $k \leq 10$, though ADAGRAD's performance catches up to and eventually surpasses AROW's as k grows.

6.3 Multiclass Optical Character Recognition

In the well-known MNIST multiclass classification data set, we are given 28×28 pixel images a_i, and the learner's task is to classify each image as a digit in $\{0,\ldots,9\}$. Linear classifiers do not work well on a simple pixel-based representation. Thus we learn classifiers built on top of a kernel machine with Gaussian kernels, as do Duchi and Singer (2009), which gives a different (and non-sparse) structure to the feature space in contrast to our previous experiments. In particular, for the ith example and jth feature, the feature value is $z_{ij} = K(a_i,a_j) \triangleq \exp\left(-\frac{1}{2\sigma^2}\left\|a_i - a_j\right\|_2^2\right)$. We use a support set of approximately 3000 images to compute the kernels and trained multiclass predictors, which consist of one vector $x_c \in \mathbb{R}^{3000}$ for each class c, giving a 30,000 dimensional problem. There is no known multiclass AROW algorithm. We therefore compare adaptive RDA with and without mixed-norm ℓ_1/ℓ_2 and ℓ_1/ℓ_∞ regularization (see Section 5.5), RDA, and multiclass Passive Aggressive to one another using the multiclass hinge loss (Crammer et al., 2006). For each algorithm we used the first 5000 of 60,000 training examples to choose the stepsize η (for RDA) and λ (for PA).

In Figure 5, we plot the learning curves (cumulative mistakes made) of multiclass PA, RDA, RDA with ℓ_1/ℓ_2 regularization, adaptive RDA, and adaptive RDA with ℓ_1/ℓ_2 regularization (ℓ_1/ℓ_∞

	Test error rate	Prop. nonzero
PA	0.062	1.000
Ada-RDA	0.066	1.000
RDA	0.108	1.000
Ada-RDA $\lambda = 5 \cdot 10^{-4}$	0.100	0.569
RDA $\lambda = 5 \cdot 10^{-4}$	0.138	0.878
Ada-RDA $\lambda = 10^{-3}$	0.137	0.144
RDA $\lambda = 10^{-3}$	0.192	0.532

Table 3: Test set error rates and sparsity proportions on MNIST. The scalar λ is the multiplier on the ℓ_1/ℓ_2 regularization term.

is similar). From the curves, we see that Adaptive RDA seems to have similar performance to PA, and the adaptive versions of RDA are vastly superior to their non-adaptive counterparts. Table 3 further supports this, where we see that the adaptive RDA algorithms outperform their non-adaptive counterparts both in terms of sparsity (the proportion of non-zero rows) and test set error rates.

6.4 Income Prediction

The KDD census income data set from the UCI repository (Asuncion and Newman, 2007) contains census data extracted from 1994 and 1995 population surveys conducted by the U.S. Census Bureau. The data consists of 40 demographic and employment related variables which are used to predict whether a respondent has income above or below \$50,000. We quantize each feature into bins (5 per feature for continuous features) and take products of features to give a 4001 dimensional feature space with 0/1 features. The data is divided into a training set of 199,523 instances and test set of 99,762 test instances.

As in the prequel, we compare AROW, PA, RDA, and adaptive RDA with and without ℓ_1-regularization on this data set. We use the first 10,000 examples of the training set to select the step size parameters λ for AROW and PA and η for RDA. We perform ten experiments on random shuffles of the training data. Each experiment consists of a training pass through some proportion of the data (.05, .1, .25, .5, or the entire training set) and computing the test set error rate of the learned predictor. Table 4 and Figure 6 summarize the results of these experiments. The variance of the test error rates is on the order of 10^{-6} so we do not report it. As earlier, the table and figure make it clear that the adaptive methods (AROW and ADAGRAD-RDA) give better performance than non-adaptive methods. Further, as detailed in the table, the ADAGRAD methods can give extremely sparse predictors that still give excellent test set performance. This is consistent with the experiments we have seen to this point, where ADAGRAD gives sparse but highly accurate predictors.

6.5 Experiments with Sparsity-Accuracy Tradeoffs

In our final set of experiments, we investigate the tradeoff between the level of sparsity and the classification accuracy for the ADAGRAD-RDA algorithms. Using the same experimental setup as for the initial text classification experiments described in Section 6.1, we record the average test-set performance of ADAGRAD-RDA versus the proportion of features that are non-zero in the predictor ADAGRAD outputs after a single pass through the training data. To achieve this, we run

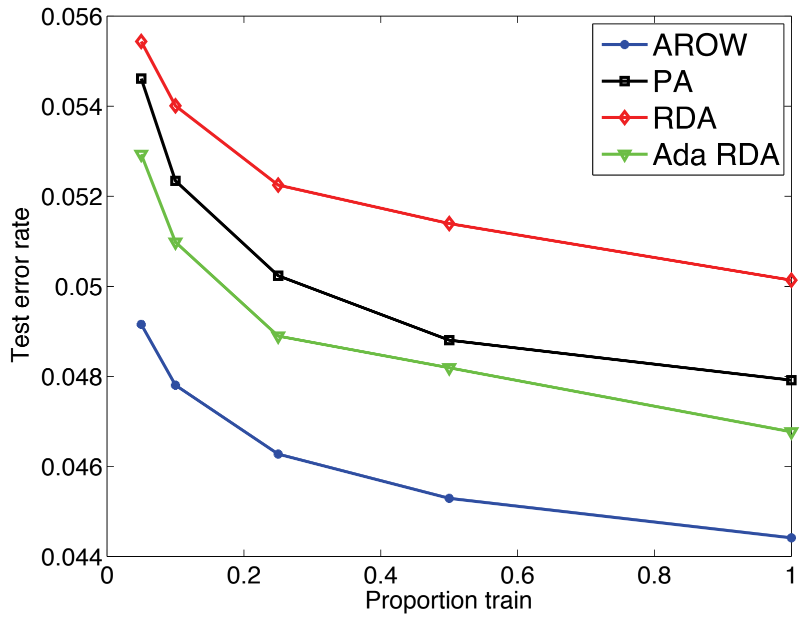


Figure 6: Test set error rates as function of proportion of training data seen on Census Income data set.

Prop. Train	0.05	0.10	0.25	0.50	1.00
AROW	0.049	0.048	0.046	0.045	0.044
PA	0.055	0.052	0.050	0.049	0.048
RDA	0.055	0.054	0.052	0.051	0.050
Ada-RDA	0.053	0.051	0.049	0.048	0.047
ℓ_1 RDA	0.056 (0.075)	0.054 (0.066)	0.053 (0.058)	0.052 (0.053)	0.051 (0.050)
ℓ_1 Ada-RDA	0.052 (0.062)	0.051 (0.053)	0.050 (0.044)	0.050 (0.040)	0.049 (0.037)

Table 4: Test set error rates as function of proportion of training data seen (proportion of non-zeros in parenthesis where appropriate) on Census Income data set.

ADAGRAD with ℓ_1-regularization, and we sweep the regularization multiplier λ from 10^{-8} to 10^{-1}. These values result in predictors ranging from a completely dense predictor to an all-zeros predictor, respectively.

We summarize our results in Figure 7, which shows the test set performance of ADAGRAD for each of the four categories ECAT, CCAT, GCAT, and MCAT. Within each plot, the horizontal black line labeled AROW designates the baseline performance of AROW on the text classification task, though we would like to note that AROW generates fully dense predictors. The plots all portray a similar story. With high regularization values, ADAGRAD exhibits, as expected, poor performance as it retains no predictive information from the learning task. Put another way, when the regularization value is high ADAGRAD is confined to an overly sparse predictor which exhibits poor generalization. However, as the regularization multiplier λ decreases, the learned predictor becomes less sparse and eventually the accuracy of ADAGRAD exceeds AROW's accuracy. It is interesting to note that for these experiments, as soon as the predictor resulting from a *single* pass

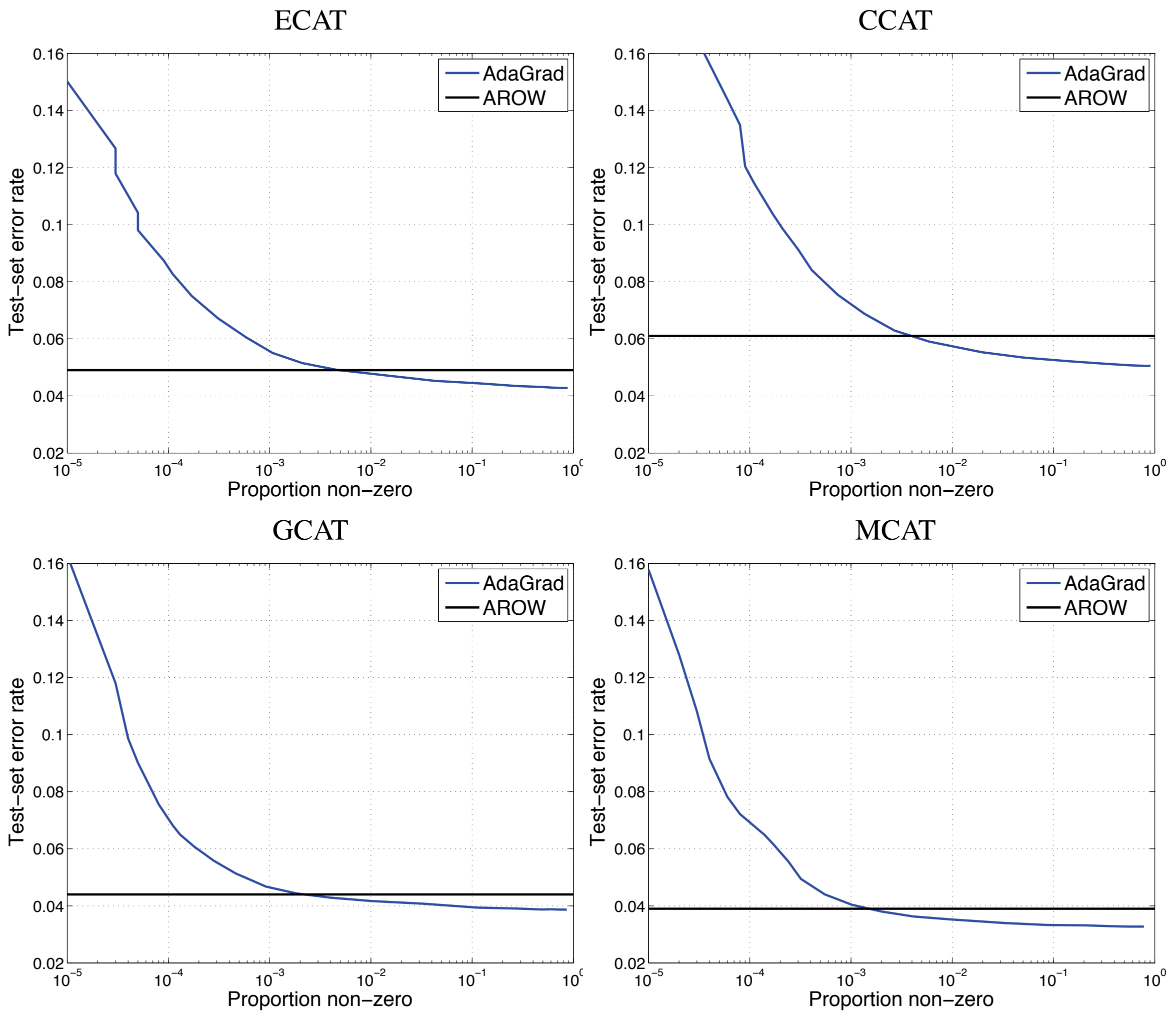


Figure 7: Test set error rates as a function of proportion of non-zeros in predictor x output by ADAGRAD (AROW plotted for reference).

through the data has more than 1% non-zero coefficients, ADAGRAD's performance matches that of AROW. We also would like to note that the variance in the test-set error rates for these experiments is on the order of 10^{-6}, and we thus do not draw error bars in the graphs. The performance of ADAGRAD as a function of regularization for other sparse data sets, especially in relation to that of AROW, was qualitatively similar to this experiment.

7. Conclusions

We presented a paradigm that adapts subgradient methods to the geometry of the problem at hand. The adaptation allows us to derive strong regret guarantees, which for some natural data distributions achieve better performance guarantees than previous algorithms. Our online regret bounds can be naturally converted into rate of convergence and generalization bounds (Cesa-Bianchi et al., 2004). Our experiments show that adaptive methods, specifically ADAGRAD-FOBOS, ADAGRAD-RDA, and AROW clearly outperform their non-adaptive counterparts. Furthermore, the ADAGRAD family of algorithms naturally incorporates regularization and gives very sparse solutions with similar performance to dense solutions. Our experiments with adaptive methods use a diagonal approximation to the matrix obtained by taking outer products of subgradients computed along the run of the algorithm. It remains to be tested whether using the full outer product matrix can further improve performance.

To conclude we would like to underscore a possible elegant generalization that interpolates between full-matrix proximal functions and diagonal approximations using block diagonal matrices. Specifically, for $v \in \mathbb{R}^d$ let $v = [v_{[1]}^\top \ \cdots \ v_{[k]}^\top]^\top$ where $v_{[i]} \in \mathbb{R}^{d_i}$ are subvectors of v with $\sum_{i=1}^k d_i = d$. We can define the associated block-diagonal approximation to the outer product matrix $\sum_{\tau=1}^t g_\tau g_\tau^\top$ by

$$G_t = \sum_{\tau=1}^t \begin{bmatrix} g_{\tau,[1]} g_{\tau,[1]}^\top & 0 & \cdots & 0 \\ 0 & g_{\tau,[2]} g_{\tau,[2]}^\top & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & g_{\tau,[k]} g_{\tau,[k]}^\top \end{bmatrix}.$$

In this case, a combination of Theorems 5 and 7 gives the next corollary.

Corollary 12 *Let G_t be the block-diagonal outer product matrix defined above and the sequence $\{x_t\}$ be defined by the RDA update of (3) with $\psi_t(x) = \left\langle x, G_t^{1/2} x \right\rangle$. Then, for any $x^* \in \mathcal{X}$,*

$$R_\phi(T) \le \frac{1}{\eta} \max_i \left\| x_{[i]}^* \right\|_2^2 \operatorname{tr}(G_T^{1/2}) + \eta \operatorname{tr}(G_T^{1/2}).$$

A similar bound holds for composite mirror-descent updates, and it is straightforward to get infimal equalities similar to those in Corollary 11 with the infimum taken over block-diagonal matrices. Such an algorithm can interpolate between the computational simplicity of the diagonal proximal functions and the ability of full matrices to capture correlation in the gradient vectors.

A few open questions stem from this line of research. The first is whether we can *efficiently* use full matrices in the proximal functions, as in Section 4. A second open issue is whether non-Euclidean proximal functions, such as the relative entropy, can be used. We also think that the strongly convex case—when f_t or φ is strongly convex—presents interesting challenges that we have not completely resolved. We hope to investigate both empirical and formal extensions of this work in the near future.

Acknowledgments

There are many people to whom we owe our sincere thanks for this research. Fernando Pereira helped push us in the direction of working on adaptive online methods and has been a constant source of discussion and helpful feedback. Samy Bengio provided us with a processed version of the ImageNet data set and was instrumental in helping to get our experiments running, and Adam Sadovsky gave many indispensable coding suggestions. The anonymous reviewers also gave several suggestions that improved the quality of the paper. Lastly, Sam Roweis was a sounding board for some of our earlier ideas on the subject, and we will miss him dearly.

Appendix A. Full Matrix Motivating Example

As in the diagonal case, as the adversary we choose $\varepsilon > 0$ and on rounds $t = 1, \ldots, \eta^2/\varepsilon^2$ play the vector $\pm v_1$. After the first η^2/ε^2 rounds, the adversary simply cycles through the vectors $v_2, \ldots, v_d$. Thus, for Zinkevich's projected gradient, we have $x_t = \alpha_{t,1} v_1$ for some multiplier $\alpha_{t,1} > 0$ when $t \le \eta^2/\varepsilon^2$. After the first η^2/ε^2 rounds, we perform the updates

$$x_{t+1} = \Pi_{\|x\|_2 \le \sqrt{d}} \left(x_t + \frac{\eta}{\sqrt{t}} v_i \right)$$

for some index i, but as in the diagonal case, $\eta/\sqrt{t} \le \varepsilon$, and by orthogonality of v_i, v_j, we have $x_t = V\alpha_t$ for some $\alpha_t \succeq 0$, and the projection step can only shrink the multiplier $\alpha_{t,i}$ for index i. Thus, each coordinate incurs loss at least $1/(2\varepsilon)$, and projected gradient descent suffers losses $\Omega(d/\varepsilon)$.

On the other hand, ADAGRAD suffers loss at most d. Indeed, since $g_1 = v_1$ and $\|v_1\|_2 = 1$, we have $G_1^2 = v_1 v_1^\top v_1 v_1^\top = v_1 v_1^\top = G_1$, so $G_1 = G_1^\dagger = G_1^{\frac{1}{2}}$, and

$$x_2 = x_1 + G_1^\dagger = x_1 + v_1 v_1^\top v_1 = x_1 + v_1.$$

Since $\langle x_2, v_1 \rangle = 1$, we see that ADAGRAD suffers no loss (and $G_t = G_1$) until a vector $z_t = \pm v_i$ for $i \ne 1$ is played by the adversary. However, an identical argument shows that G_t is simply updated to $v_1 v_1^\top + v_i v_i^\top$, in which case $x_t = v_1 + v_i$. Indeed, an inductive argument shows that until all the vectors v_i are seen, we have $\|x_t\|_2 < \sqrt{d}$ by orthogonality, and eventually we have

$$x_t = \sum_{i=1}^d v_i \quad \text{and} \quad \|x_t\|_2 = \sqrt{\sum_{i=1}^d \|v_i\|_2^2} = \sqrt{d}$$

so that $x_t \in \mathcal{X} = \{x : \|x\|_2 \le \sqrt{d}\}$ for ADAGRAD for all t. All future predictions thus achieve margin 1 and suffer no loss.

Appendix B. Technical Lemmas

Lemma 13 *Let $A \succeq B \succeq 0$ be symmetric $d \times d$ PSD matrices. Then $A^{1/2} \succeq B^{1/2}$.*

Proof This is Example 3 of Davis (1963). We include a proof for convenience of the reader. Let λ be any eigenvalue (with corresponding eigenvector x) of $A^{1/2} - B^{1/2}$; we show that $\lambda \ge 0$. Clearly $A^{1/2}x - \lambda x = B^{1/2}x$. Taking the inner product of both sides with $A^{1/2}x$, we have $\left\|A^{1/2}x\right\|_2^2 - \lambda \left\langle A^{1/2}x, x\right\rangle = \left\langle A^{1/2}x, B^{1/2}x\right\rangle$. We use the Cauchy-Schwarz inequality:

$$\left| \left\|A^{1/2}x\right\|_2^2 - \lambda \left\langle A^{1/2}x, x \right\rangle \right| \le \left\|A^{1/2}x\right\|_2 \left\|B^{1/2}x\right\|_2 = \sqrt{\langle Ax, x\rangle \langle Bx, x\rangle} \le \langle Ax, x\rangle = \left\|A^{1/2}x\right\|_2^2$$

where the last inequality follows from the assumption that $A \succeq B$. Thus we must have $\lambda \left\langle A^{1/2}x, x\right\rangle \ge 0$, which implies $\lambda \ge 0$. ■

The gradient of the function $\operatorname{tr}(X^p)$ is easy to compute for integer values of p. However, when p is real we need the following lemma. The lemma tacitly uses the fact that there is a unique positive semidefinite X^p when $X \succeq 0$ (Horn and Johnson, 1985, Theorem 7.2.6).

Lemma 14 *Let $p \in \mathbb{R}$ and $X \succ 0$. Then $\nabla_X \mathrm{tr}(X^p) = pX^{p-1}$.*

Proof We do a first order expansion of $(X+A)^p$ when $X \succ 0$ and A is symmetric. Let $X = U\Lambda U^\top$ be the symmetric eigen-decomposition of X and $VDV^\top$ be the decomposition of $\Lambda^{-1/2}U^\top AU\Lambda^{-1/2}$. Then

$$\begin{aligned}(X+A)^p &= (U\Lambda U^\top + A)^p = U(\Lambda + U^\top AU)^p U^\top = U\Lambda^{p/2}(I + \Lambda^{-1/2}U^\top AU\Lambda^{-1/2})^p \Lambda^{p/2}U^\top \\ &= U\Lambda^{p/2}V^\top (I+D)^p V\Lambda^{p/2}U^\top = U\Lambda^{p/2}V^\top(I + pD + o(D))V\Lambda^{p/2}U^\top \\ &= U\Lambda^p U^\top + pU\Lambda^{p/2}V^\top DV\Lambda^{p/2}U^\top + o(U\Lambda^{-/2}V^\top DV\Lambda^{p/2}U^\top) \\ &= X^p + U\Lambda^{(p-1)/2}U^\top AU\Lambda^{(p-1)/2}U^\top + o(A) = X^p + pX^{(p-1)/2}AX^{(p-1)/2} + o(A).\end{aligned}$$

In the above, $o(A)$ is a matrix that goes to zero faster than $A \to 0$, and the second line follows via a first-order Taylor expansion of $(1+d_i)^p$. From the above, we immediately have

$$\mathrm{tr}((X+A)^p) = \mathrm{tr}X^p + p\,\mathrm{tr}(X^{p-1}A) + o(\mathrm{tr}A),$$

which completes the proof. ■

Appendix C. Proof of Lemma 4

We prove the lemma by considering an arbitrary real-valued sequence $\{a_i\}$ and its vector representation $a_{1:i} = [a_1 \ \cdots \ a_i]$. We are next going to show that

$$\sum_{t=1}^{T} \frac{a_t^2}{\|a_{1:t}\|_2} \le 2\|a_{1:T}\|_2 \ , \tag{24}$$

where we define $\frac{0}{0} = 0$. We use induction on T to prove inequality (24). For $T = 1$, the inequality trivially holds. Assume the bound (24) holds true for $T-1$, in which case

$$\sum_{t=1}^{T} \frac{a_t^2}{\|a_{1:t}\|_2} = \sum_{t=1}^{T-1} \frac{a_t^2}{\|a_{1:t}\|_2} + \frac{a_T^2}{\|a_{1:T}\|_2} \le 2\|a_{1:T-1}\|_2 + \frac{a_T^2}{\|a_{1:T}\|_2} \ ,$$

where the inequality follows from the inductive hypothesis. We define $b_T = \sum_{t=1}^T a_t^2$ and use concavity to obtain that $\sqrt{b_T - a_T^2} \le \sqrt{b_T} - a_T^2 \frac{1}{2\sqrt{b_T}}$ so long as $b_T - a_T^2 \ge 0$.[2] Thus,

$$2\|a_{1:T-1}\|_2 + \frac{a_T^2}{\|a_{1:T}\|_2} = 2\sqrt{b_T - a_T^2} + \frac{a_T^2}{\sqrt{b_T}} \le 2\sqrt{b_T} = 2\|a_{1:T}\|_2 \ .$$

Having proved the bound (24), we note that by construction that $s_{t,i} = \|g_{1:t,i}\|_2$, so

$$\sum_{t=1}^{T} \left\langle g_t, \mathrm{diag}(s_t)^{-1} g_t \right\rangle = \sum_{t=1}^{T}\sum_{i=1}^{d} \frac{g_{t,i}^2}{\|g_{1:t,i}\|_2} \le 2\sum_{i=1}^{d} \|g_{1:T,i}\|_2 \ .$$

2. We note that we use an identical technique in the full-matrix case. See Lemma 8.

Appendix D. Proof of Lemmas 8 and 9

We begin with the more difficult proof of Lemma 8.

Proof of Lemma 8 The core of the proof is based on the concavity of the function $\mathrm{tr}(A^{1/2})$. However, careful analysis is required as A might not be strictly positive definite. We also use the previous lemma which implies that the gradient of $\mathrm{tr}(A^{1/2})$ is $\frac{1}{2}A^{-1/2}$ when $A \succ 0$.

First, A^p is matrix-concave for $A \succ 0$ and $0 \le p \le 1$ (see, for example, Corollary 4.1 in Ando, 1979 or Theorem 16.1 in Bondar, 1994). That is, for $A, B \succ 0$ and $\alpha \in [0,1]$ we have

$$(\alpha A + (1-\alpha)B)^p \succeq \alpha A^p + (1-\alpha)B^p \ . \tag{25}$$

Now suppose simply $A, B \succeq 0$ (but neither is necessarily strict). Then for any $\delta > 0$, we have $A + \delta I \succ 0$ and $B + \delta I \succ 0$ and therefore

$$(\alpha(A+\delta I) + (1-\alpha)(B+\delta I))^p \succeq \alpha(A+\delta I)^p + (1-\alpha)(B+\delta I)^p \succeq \alpha A^p + (1-\alpha)B^p \ ,$$

where we used Lemma 13 for the second matrix inequality. Moreover, $\alpha A + (1-\alpha)B + \delta I \to \alpha A + (1-\alpha)B$ as $\delta \to 0$. Since A^p is continuous (when we use the unique PSD root), this line of reasoning proves that (25) holds for $A, B \succeq 0$. Thus, we proved that

$$\mathrm{tr}((\alpha A + (1-\alpha)B)^p) \ge \alpha\, \mathrm{tr}(A^p) + (1-\alpha)\, \mathrm{tr}(B^p) \ \text{ for } 0 \le p \le 1 \ .$$

Recall now that Lemma 14 implies that the gradient of $\mathrm{tr}(A^{1/2})$ is $\frac{1}{2}A^{-1/2}$ when $A \succ 0$. Therefore, from the concavity of $A^{1/2}$ and the form of its gradient, we can use the standard first-order inequality for concave functions so that for any $A, B \succ 0$,

$$\mathrm{tr}(A^{1/2}) \le \mathrm{tr}(B^{1/2}) + \frac{1}{2}\mathrm{tr}(B^{-1/2}(A-B)) \ . \tag{26}$$

Let $A = B - \nu g g^\top \succeq 0$ and suppose only that $B \succeq 0$. We must take some care since $B^{-1/2}$ may not necessarily exist, and the above inequality does not hold true in the pseudo-inverse sense when $B \not\succ 0$. However, for any $\delta > 0$ we know that $2\nabla_B \mathrm{tr}((B+\delta I)^{1/2}) = (B+\delta I)^{-1/2}$, and $A - B = -\nu g g^\top$. From (26) and Lemma 13, we have

$$\begin{aligned} 2\,\mathrm{tr}(B - tgg^\top)^{1/2} \ &= \ 2\,\mathrm{tr}(A^{1/2}) \ \le \ 2\,\mathrm{tr}((A+\delta I)^{1/2}) \\ &\le \ 2\,\mathrm{tr}(B+\delta I)^{1/2} - \nu\, \mathrm{tr}((B+\delta I)^{-1/2} g g^\top) \ . \end{aligned} \tag{27}$$

Note that $g \in \mathrm{Range}(B)$, because if it were not, we could choose some u with $Bu = 0$ and $\langle g, u \rangle \ne 0$, which would give $\langle u, (B - cgg^\top)u \rangle = -c\langle g, u\rangle^2 < 0$, a contradiction. Now let $B = V \mathrm{diag}(\lambda) V^\top$ be the eigen-decomposition of B. Since $g \in \mathrm{Range}(B)$,

$$\begin{aligned} g^\top (B+\delta I)^{-1/2} g \ &= \ g^\top V \mathrm{diag}\Big(1/\sqrt{\lambda_i + \delta}\Big) V^\top g \\ &= \ \sum_{i:\lambda_i > 0} \frac{1}{\sqrt{\lambda_i + \delta}} (g^\top v_i)^2 \ \xrightarrow[\delta \downarrow 0]{} \ \sum_{i:\lambda_i > 0} \lambda_i^{-1/2} (g^\top v_i)^2 \ = \ g^\top (B^\dagger)^{1/2} g \ . \end{aligned}$$

Thus, by taking $\delta \downarrow 0$ in (27), and since both $\mathrm{tr}(B+\delta I)^{1/2}$ and $\mathrm{tr}((B+\delta I)^{-1/2} g g^\top)$ are evidently continuous in δ, we complete the proof. ■

Proof of Lemma 9 We begin by noting that $\delta^2 I \succeq gg^\top$, so from Lemma 13 we get $(A+gg^\top)^{1/2} \preceq (A+\delta^2 I)^{1/2}$. Since A and I are simultaneously diagonalizable, we can generalize the inequality $\sqrt{a+b} \le \sqrt{a}+\sqrt{b}$, which holds for $a,b \ge 0$, to positive semi-definite matrices, thus,

$$(A+\delta^2 I)^{1/2} \preceq A^{1/2} + \delta I \ .$$

Therefore, if $A+gg^\top$ is of full rank, we have $(A+gg^\top)^{-1/2} \succeq (A^{1/2}+\delta I)^{-1}$ (Horn and Johnson, 1985, Corollary 7.7.4(a)). Since $g \in \text{Range}((A+gg^\top)^{1/2})$, we can apply an analogous limiting argument to the one used in the proof of Lemma 8 and discard all zero eigenvalues of $A+gg^\top$, which completes the lemma. ■

Appendix E. Solution to Problem (15)

We prove here a technical lemma that is useful in characterizing the solution of the optimization problem below. Note that the second part of the lemma implies that we can treat the inverse of the solution matrix S^{-1} as $S^\dagger$. We consider solving

$$\min_S \ \text{tr}(S^{-1}A) \ \text{ subject to } \ S \succeq 0, \ \text{tr}(S) \le c \ \text{ where } \ A \succeq 0 \ . \tag{28}$$

Lemma 15 *If A is of full rank, then the minimizer of (28) is $S = cA^{\frac{1}{2}}/\text{tr}(A^{\frac{1}{2}})$. If A is not of full rank, then setting $S = cA^{\frac{1}{2}}/\text{tr}(A^{\frac{1}{2}})$ gives*

$$\text{tr}(S^\dagger A) = \inf_S \left\{ \text{tr}(S^{-1}A) : S \succeq 0, \ \text{tr}(S) \le c \right\} \ .$$

In either case, $\text{tr}(S^\dagger A) = \text{tr}(A^{\frac{1}{2}})^2/c$.

Proof Both proofs rely on constructing the Lagrangian for (28). We introduce $\theta \in \mathbb{R}_+$ for the trace constraint and $Z \succeq 0$ for the positive semidefinite constraint on S. In this case, the Lagrangian is

$$\mathcal{L}(S,\theta,Z) = \text{tr}(S^{-1}A) + \theta(\text{tr}(S) - c) - \text{tr}(SZ).$$

The derivative of $\mathcal{L}$ with respect to S is

$$-S^{-1}AS^{-1} + \theta I - Z. \tag{29}$$

If S is full rank, then to satisfy the generalized complementarity conditions for the problem (Boyd and Vandenberghe, 2004), we must have $Z = 0$. Therefore, we get $S^{-1}AS^{-1} = \theta I$. We now can multiply by S on the right and the left to get that $A = \theta S^2$, which implies that $S \propto A^{\frac{1}{2}}$. If A is of full rank, the optimal solution for $S \succ 0$ forces θ to be positive so that $\text{tr}(S) = c$. This yields the solution $S = cA^{\frac{1}{2}}/\text{tr}(A^{\frac{1}{2}})$. In order to verify optimality of this solution, we set $Z = 0$ and $\theta = c^{-2}\,\text{tr}(A^{1/2})^2$ which gives $\nabla_S \mathcal{L}(S,\theta,Z) = 0$, as is indeed required.

Suppose now that A is not full rank and that

$$A = Q \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} Q^\top$$

is the eigen-decomposition of A. Let n be the dimension of the null-space of A (so the rank of A is $d-n$). Define the variables

$$Z(\theta) = \begin{bmatrix} 0 & 0 \\ 0 & \theta I \end{bmatrix}, \quad S(\theta,\delta) = \frac{1}{\sqrt{\theta}} Q \begin{bmatrix} \Lambda^{\frac{1}{2}} & 0 \\ 0 & \delta I \end{bmatrix} Q^\top, \quad S(\delta) = \frac{c}{\operatorname{tr}(A^{\frac{1}{2}}) + \delta n} Q \begin{bmatrix} \Lambda^{\frac{1}{2}} & 0 \\ 0 & \delta I \end{bmatrix} Q^\top.$$

It is easy to see that $\operatorname{tr} S(\delta) = c$, and

$$\lim_{\delta \to 0} \operatorname{tr}(S(\delta)^{-1}A) = \operatorname{tr}(S(0)^\dagger A) = \operatorname{tr}(A^{\frac{1}{2}}) \operatorname{tr}(\Lambda^{\frac{1}{2}})/c = \operatorname{tr}(A^{\frac{1}{2}})^2/c.$$

Further, let $g(\theta) = \inf_S \mathcal{L}(S,\theta,Z(\theta))$ be the dual of (28). From the above analysis and (29), it is evident that

$$-S(\theta,\delta)^{-1} A S(\theta,\delta)^{-1} + \theta I - Z(\theta) = -\theta Q \begin{bmatrix} \Lambda^{-\frac{1}{2}} \Lambda \Lambda^{-\frac{1}{2}} & 0 \\ 0 & \delta^{-2} I \cdot 0 \end{bmatrix} Q^\top + \theta I - \begin{bmatrix} 0 & 0 \\ 0 & \theta I \end{bmatrix} = 0.$$

So $S(\theta,\delta)$ achieves the infimum in the dual for *any* $\delta > 0$, $\operatorname{tr}(S(0)Z(\theta)) = 0$, and

$$g(\theta) = \sqrt{\theta} \operatorname{tr}(\Lambda^{\frac{1}{2}}) + \sqrt{\theta} \operatorname{tr}(\Lambda^{\frac{1}{2}}) + \sqrt{\theta}\delta n - \theta c.$$

Setting $\theta = \operatorname{tr}(\Lambda^{\frac{1}{2}})^2/c^2$ gives $g(\theta) = \operatorname{tr}(\Lambda^{\frac{1}{2}})^2/c - \delta n \operatorname{tr}(\Lambda^{\frac{1}{2}})/c$. Taking $\delta \to 0$ gives $g(\theta) = \operatorname{tr}(A^{\frac{1}{2}})^2/c$, which means that $\lim_{\delta \to 0} \operatorname{tr}(S(\delta)^{-1}A) = \operatorname{tr}(A^{\frac{1}{2}})^2/c = g(\theta)$. Thus the duality gap for the original problem is 0 so $S(0)$ is the limiting solution.

The last statement of the lemma is simply plugging $S^\dagger = (A^\dagger)^{\frac{1}{2}} \operatorname{tr}(A^{\frac{1}{2}})/c$ in to the objective being minimized. ■

Appendix F. Proofs of Propositions 2 and 3

We begin with the proof of Proposition 2. The proof essentially builds upon Xiao (2010) and Nesterov (2009), with some modification to deal with the indexing of ψ_t. We include the proof for completeness.

Proof of Proposition 2 Define ψ_t^* to be the conjugate dual of $t\varphi(x) + \psi_t(x)/\eta$:

$$\psi_t^*(g) = \sup_{x \in \mathcal{X}} \left\{ \langle g, x \rangle - t\varphi(x) - \frac{1}{\eta}\psi_t(x) \right\}.$$

Since ψ_t/η is $1/\eta$-strongly convex with respect to the norm $\|\cdot\|_{\psi_t}$, the function ψ_t^* has η-Lipschitz continuous gradients with respect to $\|\cdot\|_{\psi_t^*}$:

$$\|\nabla\psi_t^*(g_1) - \nabla\psi_t^*(g_2)\|_{\psi_t} \le \eta \|g_1 - g_2\|_{\psi_t^*} \tag{30}$$

for any g_1, g_2 (see, e.g., Nesterov, 2005, Theorem 1 or Hiriart-Urruty and Lemaréchal, 1996, Chapter X). Further, a simple argument with the fundamental theorem of calculus gives that if f has L-Lipschitz gradients, $f(y) \le f(x) + \langle \nabla f(x), y - x \rangle + (L/2)\|y - x\|^2$, and

$$\nabla\psi_t^*(g) = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ -\langle g, x \rangle + t\varphi(x) + \frac{1}{\eta}\psi_t(x) \right\}. \tag{31}$$

Using the bound (30) and identity (31), we can give the proof of the corollary. Indeed, letting $g_t \in \partial f_t(x_t)$ and defining $z_t = \sum_{\tau=1}^t g_\tau$, we have

$$\begin{aligned}
&\sum_{t=1}^T f_t(x_t) + \varphi(x_t) - f_t(x^*) - \varphi(x^*) \\
&\le \sum_{t=1}^T \langle g_t, x_t - x^* \rangle - \varphi(x^*) + \varphi(x_t) \\
&\le \sum_{t=1}^T \langle g_t, x_t \rangle + \varphi(x_t) + \sup_{x \in \mathcal{X}} \left\{ -\sum_{t=1}^T \langle g_t, x \rangle - T\varphi(x) - \frac{1}{\eta}\psi_T(x) \right\} + \psi_T(x^*) \\
&= \frac{1}{\eta}\psi_T(x^*) + \sum_{t=1}^T \langle g_t, x_t \rangle + \varphi(x_t) + \psi_T^*(-z_T).
\end{aligned}$$

Since $\psi_{t+1} \ge \psi_t$, it is clear that

$$\begin{aligned}
\psi_T^*(-z_T) &= -\sum_{t=1}^T \langle g_t, x_{T+1} \rangle - T\varphi(x_{T+1}) - \frac{1}{\eta}\psi_T(x_{T+1}) \\
&\le -\sum_{t=1}^T \langle g_t, x_{T+1} \rangle - (T-1)\varphi(x_{T+1}) - \varphi(x_{T+1}) - \frac{1}{\eta}\psi_{T-1}(x_{T+1}) \\
&\le \sup_{x \in \mathcal{X}} \left(-\langle z_T, x \rangle - (T-1)\varphi(x) - \frac{1}{\eta}\psi_{T-1}(x) \right) - \varphi(x_{T+1}) = \psi_{T-1}^*(-z_T) - \varphi(x_{T+1}).
\end{aligned}$$

The Lipschitz continuity of $\nabla\psi_t^*$, the identity (31), and the fact that $z_T - z_{T-1} = -g_T$ give

$$\begin{aligned}
&\sum_{t=1}^T f_t(x_t) + \varphi(x_{t+1}) - f_t(x^*) - \varphi(x^*) \\
&\le \frac{1}{\eta}\psi_T(x^*) + \sum_{t=1}^T \langle g_t, x_t \rangle + \varphi(x_{t+1}) + \psi_{T-1}^*(-z_T) - \varphi(x_{T+1}) \\
&\le \frac{1}{\eta}\psi_T(x^*) + \sum_{t=1}^T \langle g_t, x_t \rangle + \varphi(x_{t+1}) - \varphi(x_{T+1}) \\
&\qquad + \psi_{T-1}^*(-z_{T-1}) - \left\langle \nabla\psi_{T-1}^*(z_{T-1}), g_T \right\rangle + \frac{\eta}{2}\|g_T\|_{\psi_{T-1}^*}^2 \\
&= \frac{1}{\eta}\psi_T(x^*) + \sum_{t=1}^{T-1} \langle g_t, x_t \rangle + \varphi(x_{t+1}) + \psi_{T-1}^*(-z_{T-1}) + \frac{\eta}{2}\|g_T\|_{\psi_{T-1}^*}^2.
\end{aligned}$$

We can repeat the same sequence of steps that gave the last equality to see that

$$\sum_{t=1}^T f_t(x_t) + \varphi(x_{t+1}) - f_t(x^*) - \varphi(x^*) \le \frac{1}{\eta}\psi_T(x^*) + \frac{\eta}{2}\sum_{t=1}^T \|g_t\|_{\psi_{t-1}^*}^2 + \psi_0^*(-z_0).$$

Recalling that $x_1 = \operatorname{argmin}_{x \in \mathcal{X}}\{\varphi(x)\}$ and that $\psi_0^*(0) = 0$ completes the proof. ■

We now turn to the proof of Proposition 3. We begin by stating and fully proving an (essentially) immediate corollary to Lemma 2.3 of Duchi et al. (2010).

Lemma 16 *Let $\{x_t\}$ be the sequence defined by the update (4) and assume that $B_{\psi_t}(\cdot,\cdot)$ is strongly convex with respect to a norm $\|\cdot\|_{\psi_t}$. Let $\|\cdot\|_{\psi_t^*}$ be the associated dual norm. Then for any x^*,*

$$\eta\left(f_t(x_t)-f_t(x^*)\right)+\eta\left(\varphi(x_{t+1})-\varphi(x^*)\right)\le B_{\psi_t}(x^*,x_t)-B_{\psi_t}(x^*,x_{t+1})+\frac{\eta^2}{2}\left\|f_t'(x_t)\right\|_{\psi_t^*}^2$$

Proof The optimality of x_{t+1} for (4) implies for all $x\in\mathcal{X}$ and $\varphi'(x_{t+1})\in\partial\varphi(x_{t+1})$

$$\left\langle x-x_{t+1},\eta f'(x_t)+\nabla\psi_t(x_{t+1})-\nabla\psi_t(x_t)+\eta\varphi'(x_{t+1})\right\rangle\ge 0. \tag{32}$$

In particular, this obtains for $x=x^*$. From the subgradient inequality for convex functions, we have $f_t(x^*)\ge f_t(x_t)+\langle f_t'(x_t),x^*-x_t\rangle$, or $f_t(x_t)-f_t(x^*)\le\langle f_t'(x_t),x_t-x^*\rangle$, and likewise for $\varphi(x_{t+1})$. We thus have

$$\begin{aligned}
&\eta\left[f_t(x_t)+\varphi(x_{t+1})-f_t(x^*)-\varphi(x^*)\right]\\
&\quad\le\ \eta\left\langle x_t-x^*,f_t'(x_t)\right\rangle+\eta\left\langle x_{t+1}-x^*,\varphi'(x_{t+1})\right\rangle\\
&\quad=\ \eta\left\langle x_{t+1}-x^*,f_t'(x_t)\right\rangle+\eta\left\langle x_{t+1}-x^*,\varphi'(x_{t+1})\right\rangle+\eta\left\langle x_t-x_{t+1},f_t'(x_t)\right\rangle\\
&\quad=\ \left\langle x^*-x_{t+1},\nabla\psi_t(x_t)-\nabla\psi_t(x_{t+1})-\eta f_t'(x_t)-\eta\varphi'(x_{t+1})\right\rangle\\
&\qquad\quad+\left\langle x^*-x_{t+1},\nabla\psi_t(x_{t+1})-\nabla\psi_t(x_t)\right\rangle+\eta\left\langle x_t-x_{t+1},f_t'(x_t)\right\rangle.
\end{aligned}$$

Now, by (32), the first term in the last equation is non-positive. Thus we have that

$$\begin{aligned}
&\eta\left[f_t(x_t)+\varphi(x_{t+1})-f_t(x^*)-\varphi(x^*)\right]\\
&\quad\le\ \left\langle x^*-x_{t+1},\nabla\psi_t(x_{t+1})-\nabla\psi_t(x_t)\right\rangle+\eta\left\langle x_t-x_{t+1},f_t'(x_t)\right\rangle\\
&\quad=\ B_{\psi_t}(x^*,x_t)-B_{\psi_t}(x_{t+1},x_t)-B_{\psi_t}(x^*,x_{t+1})+\eta\left\langle x_t-x_{t+1},f_t'(x_t)\right\rangle\\
&\quad=\ B_{\psi_t}(x^*,x_t)-B_{\psi_t}(x_{t+1},x_t)-B_{\psi_t}(x^*,x_{t+1})+\eta\left\langle\eta^{-\frac{1}{2}}(x_t-x_{t+1}),\sqrt{\eta}f_t'(x_t)\right\rangle\\
&\quad\le\ B_{\psi_t}(x^*,x_t)-B_{\psi_t}(x_{t+1},x_t)-B_{\psi_t}(x^*,x_{t+1})+\frac{1}{2}\left\|x_t-x_{t+1}\right\|_{\psi_t}^2+\frac{\eta^2}{2}\left\|f_t'(x_t)\right\|_{\psi_t^*}^2\\
&\quad\le\ B_{\psi_t}(x^*,x_t)-B_{\psi_t}(x^*,x_{t+1})+\frac{\eta^2}{2}\left\|f_t'(x_t)\right\|_{\psi_t^*}^2.
\end{aligned}$$

In the above, the first equality follows from simple algebra with Bregman divergences, the second to last inequality follows from Fenchel's inequality applied to the conjugate functions $\frac{1}{2}\|\cdot\|_{\psi_t}^2$ and $\frac{1}{2}\|\cdot\|_{\psi_t^*}^2$ (Boyd and Vandenberghe, 2004, Example 3.27), and the last inequality follows from the assumed strong convexity of B_{ψ_t} with respect to the norm $\|\cdot\|_{\psi_t}$. ■

Proof of Proposition 3 Sum the equation in the conclusion of Lemma 16. ■

Appendix G. Derivations of Algorithms

In this appendix, we give the formal derivations of the solution to the ADAGRAD update for ℓ_1-regularization and projection to an ℓ_1-ball, as described originally in Section 5.

G.1 ℓ_1-regularization

We give the derivation for the primal-dual subgradient update, as composite mirror-descent is entirely similar. We need to solve update (3), which amounts to

$$\min_x \ \eta \langle g_t, x\rangle + \frac{1}{2t}\delta \|x\|_2^2 + \frac{1}{2t}\langle x, \operatorname{diag}(s_t)x\rangle + \eta\lambda \|x\|_1 \ .$$

Let $\hat{x}$ denote the optimal solution of the above optimization problem. Standard subgradient calculus implies that when $|g_{t,i}| \le \lambda$ the solution is $\hat{x}_i = 0$. Similarly, when $g_{t,i} < -\lambda$, then $\hat{x}_i > 0$, the objective is differentiable, and the solution is obtained by setting the gradient to zero:

$$\eta g_{t,i} + \frac{H_{t,ii}}{t}\hat{x}_i + \eta\lambda = 0 \ , \quad \text{so that} \quad \hat{x}_i = \frac{\eta t}{H_{t,ii}}\left(-g_{t,i} - \lambda\right) \ .$$

Likewise, when $g_{t,i} > \lambda$ then $\hat{x}_i < 0$, and the solution is $\hat{x}_i = \frac{\eta t}{H_{t,ii}}(-g_{t,i} + \lambda)$. Combining the three cases, we obtain the simple update (19) for $x_{t+1,i}$.

G.2 ℓ_1-ball projections

The derivation we give is somewhat terse, and we refer the interested reader to Brucker (1984) or Pardalos and Rosen (1990) for more depth. Recall that our original problem (20) is symmetric in its objective and constraints, so we assume without loss of generality that $v \succeq 0$ (otherwise, we reverse the sign of each negative component in v, then flip the sign of the corresponding component in the solution vector). This gives

$$\min_z \ \frac{1}{2}\|z - v\|_2^2 \ \text{ s.t. } \langle a, z\rangle \le c, \ z \succeq 0 \ .$$

Clearly, if $\langle a, v\rangle \le c$ the optimal $z^* = v$, hence we assume that $\langle a, v\rangle > c$. We also assume without loss of generality that $v_i/a_i \ge v_{i+1}/a_{i+1}$ for simplicity of our derivation. (We revisit this assumption at the end of the derivation.) Introducing Lagrange multipliers $\theta \in \mathbb{R}_+$ for the constraint that $\langle a, z\rangle \le c$ and $\alpha \in \mathbb{R}_+^d$ for the positivity constraint on z, we get

$$\mathcal{L}(z, \alpha, \theta) = \frac{1}{2}\|z - v\|_2^2 + \theta(\langle a, z\rangle - c) - \langle \alpha, z\rangle \ .$$

Computing the gradient of $\mathcal{L}$, we have $\nabla_z \mathcal{L}(z, \alpha, \theta) = z - v + \theta a - \alpha$. Suppose that we knew the optimal $\theta^* \ge 0$. Using the complementarity conditions on z and α for optimality of z (Boyd and Vandenberghe, 2004), we see that the solution z_i^* satisfies

$$z_i^* = \begin{cases} v_i - \theta^* a_i & \text{if } v_i \ge \theta^* a_i \\ 0 & \text{otherwise .} \end{cases}$$

Analogously, the complimentary conditions on $\langle a, z\rangle \le c$ show that given θ^*, we have

$$\sum_{i=1}^d a_i \left[v_i - \theta^* a_i\right]_+ = c \quad \text{or} \quad \sum_{i=1}^d a_i^2 \left[\frac{v_i}{a_i} - \theta^*\right]_+ = c \ .$$

Conversely, had we obtained a value $\theta \ge 0$ satisfying the above equation, then θ would evidently induce the optimal z^* through the equation $z_i = [v_i - \theta a_i]_+$.

Now, let ρ be the largest index in $\{1,\ldots,d\}$ such that $v_i - \theta^* a_i > 0$ for $i \le \rho$ and $v_i - \theta^* a_i \le 0$ for $i > \rho$. From the assumption that $v_i/a_i \le v_{i+1}/a_{i+1}$, we have $v_{\rho+1}/a_{\rho+1} \le \theta^* < v_\rho/a_\rho$. Thus, had we known the last non-zero index ρ, we would have obtained

$$\sum_{i=1}^{\rho} a_i v_i - \frac{v_\rho}{a_\rho}\sum_{i=1}^{\rho} a_i^2 = \sum_{i=1}^{\rho} a_i^2\left(\frac{v_i}{a_i} - \frac{v_\rho}{a_\rho}\right) < c \ ,$$
$$\sum_{i=1}^{\rho} a_i v_i - \frac{v_{\rho+1}}{a_{\rho+1}}\sum_{i=1}^{\rho} a_i^2 = \sum_{i=1}^{\rho+1} a_i^2\left(\frac{v_i}{a_i} - \frac{v_{\rho+1}}{a_{\rho+1}}\right) \ge c \ .$$

Given ρ satisfying the above inequalities, we can reconstruct the optimal θ^* by noting that the latter inequality should equal c exactly when we replace v_ρ/a_ρ with θ, that is,

$$\theta^* = \frac{\sum_{i=1}^{\rho} a_i v_i - c}{\sum_{i=1}^{\rho} a_i^2} \ . \tag{33}$$

The above derivation results in the following procedure (when $\langle a, v\rangle > c$). We sort v in descending order of v_i/a_i and find the largest index ρ such that $\sum_{i=1}^{\rho} a_i v_i - (v_\rho/a_\rho)\sum_{i=1}^{\rho-1} a_i^2 < c$. We then reconstruct θ^* using equality (33) and return the soft-thresholded values of v_i (see Algorithm 3). It is easy to verify that the algorithm can be implemented in $O(d\log d)$ time. A randomized search with bookkeeping (Pardalos and Rosen, 1990) can be straightforwardly used to derive a linear time algorithm.

References

J. Abernethy, P. L. Bartlett, A. Rakhlin, and A. Tewari. Optimal strategies and minimax lower bounds for online convex games. In *Proceedings of the Twenty First Annual Conference on Computational Learning Theory*, 2008.

T. Ando. Concavity of certain maps on positive definite matrices and applications to Hadamard products. *Linear Algebra and its Applications*, 26:203–241, 1979.

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

P. Auer and C. Gentile. Adaptive and self-confident online learning algorithms. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 2000.

P. L. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In *Advances in Neural Information Processing Systems 20*, 2007.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.

J. V. Bondar. Comments on and complements to *Inequalities: Theory of Majorization and Its Applications*. *Linear Algebra and its Applications*, 199:115–129, 1994.

A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

P. Brucker. An $O(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters*, 3 (3):163–166, 1984.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, September 2004.

N. Cesa-Bianchi, A. Conconi, , and C. Gentile. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005.

N. Cesa-Bianchi, Y. Mansour, and G. Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66:321–352, 2007.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

K. Crammer, M. Dredze, and F. Pereira. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems 22*, 2008.

K. Crammer, M. Dredze, and A. Kulesza. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems 23*, 2009.

C. Davis. Notions generalizing convexity for functions defined on spaces of matrices. In *Proceedings of the Symposia in Pure Mathematics*, volume 7, pages 187–201. American Mathematical Society, 1963.

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: a large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2873–2908, 2009.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In *Proceedings of the Twenty Third Annual Conference on Computational Learning Theory*, 2010.

R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970.

D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1371–1384, 2008.

E. Hazan and S. Kale. Extracting certainty from uncertainty: regret bounded by variation in costs. In *Proceedings of the Twenty First Annual Conference on Computational Learning Theory*, 2008.

E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006.

J. B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II*. Springer-Verlag, 1996.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

A. Juditsky, A. Nemirovski, and C. Tauvel. Solving variational inequalities with the stochastic mirror-prox algorithm. http://arxiv.org/abs/0809.0815, 2008.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2003.

G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming Series A*, 2010. Online first; to appear.

D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

H. B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. In *Proceedings of the Twenty Third Annual Conference on Computational Learning Theory*, 2010.

A. Nedić. *Subgradient Methods for Convex Minimization*. PhD thesis, Massachusetts Institute of Technology, 2002.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Efficiency in Optimization*. John Wiley and Sons, 1983.

Y. Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103: 127–152, 2005.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

G. Obozinski, B. Taskar, and M. Jordan. Joint covariate selection for grouped classification. Technical Report 743, Dept. of Statistics, University of California Berkeley, 2007.

P. M. Pardalos and J. B. Rosen. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46:321–328, 1990.

A. Rakhlin. Lecture notes on online learning. For the Statistical Machine Learning Course at University of California, Berkeley, 2009.

G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5), 1988.

N. Z. Shor. Utilization of the operation of space dilation in the minimization of convex functions. *Cybernetics and Systems Analysis*, 6(1):7–15, 1972. Translated from *Kibernetika*.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Technical report, Department of Mathematics, University of Washington, 2008.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. Technical Report MSR-TR-2010-23, Microsoft Research, 2010.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

Journal of Machine Learning Research 12 (2011) 2161-2180 Submitted 12/10; Revised 4/11; Published 7/11

On the Relation between Realizable and Nonrealizable Cases of the Sequence Prediction Problem

Daniil Ryabko DANIIL@RYABKO.NET
INRIA Lille-Nord Europe
40, avenue Halley
Parc Scientifique de la Haute Borne
59650 Villeneuve d'Ascq, France

Editor: Nicolò Cesa-Bianchi

Abstract

A sequence $x_1,\dots,x_n,\dots$ of discrete-valued observations is generated according to some unknown probabilistic law (measure) μ. After observing each outcome, one is required to give conditional probabilities of the next observation. The realizable case is when the measure μ belongs to an arbitrary but known class $\mathcal{C}$ of process measures. The non-realizable case is when μ is completely arbitrary, but the prediction performance is measured with respect to a given set $\mathcal{C}$ of process measures. We are interested in the relations between these problems and between their solutions, as well as in characterizing the cases when a solution exists and finding these solutions. We show that if the quality of prediction is measured using the total variation distance, then these problems coincide, while if it is measured using the expected average KL divergence, then they are different. For some of the formalizations we also show that when a solution exists it can be obtained as a Bayes mixture over a countable subset of $\mathcal{C}$. We also obtain several characterization of those sets $\mathcal{C}$ for which solutions to the considered problems exist. As an illustration to the general results obtained, we show that a solution to the non-realizable case of the sequence prediction problem exists for the set of all finite-memory processes, but does not exist for the set of all stationary processes. It should be emphasized that the framework is completely general: the processes measures considered are not required to be i.i.d., mixing, stationary, or to belong to any parametric family.

Keywords: sequence prediction, time series, online prediction, realizable sequence prediction, non-realizable sequence prediction

1. Introduction

A sequence $x_1,\dots,x_n,\dots$ of discrete-valued observations (where x_i belong to a finite set $\mathcal{X}$) is generated according to some unknown probabilistic law (measure). That is, μ is a probability measure on the space $\Omega = (\mathcal{X}^\infty,\mathcal{B})$ of one-way infinite sequences (here $\mathcal{B}$ is the usual Borel σ-algebra). After each new outcome x_n is revealed, one is required to predict conditional *probabilities* of the next observation $x_{n+1} = a$, $a \in \mathcal{X}$, given the past $x_1,\dots,x_n$. Since a predictor ρ is required to give conditional probabilities $\rho(x_{n+1} = a|x_1,\dots,x_n)$ for all possible histories $x_1,\dots,x_n$, it defines itself a probability measure on the space Ω of one-way infinite sequences. In other words, a probability measure on Ω can be considered both as a data-generating mechanism and as a predictor.

Therefore, given a set $\mathcal{C}$ of probability measures on Ω, one can ask two kinds of questions about $\mathcal{C}$. First, does there exist a predictor ρ whose forecast probabilities converge (in a certain sense) to the μ-conditional probabilities, if an arbitrary $\mu \in \mathcal{C}$ is chosen to generate the data? Here we assume that the "true" measure that generates the data belongs to the set $\mathcal{C}$ of interest, and would like to construct a predictor that predicts all measures in $\mathcal{C}$. The second type of questions is as follows: does there exist a predictor that predicts at least as well as any predictor $\rho \in \mathcal{C}$, if the measure that generates the data comes possibly from outside of $\mathcal{C}$? Thus, here we consider elements of $\mathcal{C}$ as predictors, and we would like to combine their predictive properties, if this is possible. Note that in this setting the two questions above concern the same object: a set $\mathcal{C}$ of probability measures on Ω.

Each of these two questions, the realizable and the non-realizable one, have enjoyed much attention in the literature; the setting for the non-realizable case is usually slightly different, which is probably why it has not (to the best of the author's knowledge) been studied as another facet of the realizable case. The realizable case traces back to Laplace, who has considered the problem of predicting outcomes of a series of independent tosses of a biased coin. That is, he has considered the case when the set $\mathcal{C}$ is that of all i.i.d. process measures. Other classical examples studied are the set of all computable (or semi-computable) measures (Solomonoff, 1978), the set of k-order Markov and finite-memory processes (e.g., Krichevsky, 1993) and the set of all stationary processes (Ryabko, 1988). The general question of finding predictors for an arbitrary given set $\mathcal{C}$ of process measures has been addressed in Ryabko and Hutter (2007, 2008); Ryabko (2010a); the latter work shows that when a solution exists it can be obtained as a Bayes mixture over a countable subset of $\mathcal{C}$.

The non-realizable case is usually studied in a slightly different, non-probabilistic, setting. We refer to Cesa-Bianchi and Lugosi (2006) for a comprehensive overview. It is usually assumed that the observed sequence of outcomes is an arbitrary (deterministic) sequence; it is required not to give conditional probabilities, but just deterministic guesses (although these guesses can be selected using randomisation). Predictions result in a certain loss, which is required to be small as compared to the loss of a given set of reference predictors (experts) $\mathcal{C}$. The losses of the experts and the predictor are observed after each round. In this approach, it is mostly assumed that the set $\mathcal{C}$ is finite or countable. The main difference with the formulation considered in this work is that we require a predictor to give probabilities, and thus the loss is with respect to something never observed (probabilities, not outcomes). The loss itself is not completely observable in our setting. In this sense our non-realizable version of the problem is more difficult. Assuming that the data generating mechanism is probabilistic, even if it is completely unknown, makes sense in such problems as, for example, game playing, or market analysis. In these cases one may wish to assign smaller loss to those models or experts who give probabilities closer to the correct ones (which are never observed), even though different probability forecasts can often result in the same action. Aiming at predicting probabilities of outcomes also allows us to abstract from the actual use of the predictions (for example, making bets) and thus from considering losses in a general form; instead, we can concentrate on those forms of loss that are more convenient for the analysis. In this latter respect, the problems we consider are easier than those considered in prediction with expert advice. (However, in principle, nothing restricts us to considering the simple losses that we chose; they are just a convenient choice.) Noteworthy, the probabilistic approach also makes the machinery of probability theory applicable, hopefully making the problem easier. A reviewer suggested the following summary explanation of the difference between the non-realizable problems of this work and prediction with expert advice: the latter is prequential (in the sense of Dawid, 1992), whereas the former is not.

In this work we consider two measures of the quality of prediction. The first one is the total variation distance, which measures the difference between the forecast and the "true" conditional probabilities of all future events (not just the probability of the next outcome). The second one is expected (over the data) average (over time) Kullback-Leibler divergence. Requiring that predicted and true probabilities converge in total variation is very strong; in particular, this is possible if (Blackwell and Dubins, 1962) and only if (Kalai and Lehrer, 1994) the process measure generating the data is absolutely continuous with respect to the predictor. The latter fact makes the sequence prediction problem relatively easy to analyse. Here we investigate what can be paralleled for the other measure of prediction quality (average KL divergence), which is much weaker, and thus allows for solutions for the cases of much larger sets $\mathcal{C}$ of process measures (considered either as predictors or as data generating mechanisms).

Having introduced our measures of prediction quality, we can further break the non-realizable case into two problems. The first one is as follows. Given a set $\mathcal{C}$ of predictors, we want to find a predictor whose prediction error converges to zero if there is at least one predictor in $\mathcal{C}$ whose prediction error converges to zero; we call this problem simply the "non-realizable" case, or Problem 2 (leaving the name "Problem 1" to the realizable case). The second non-realizable problem is the "fully agnostic" problem: it is to make the prediction error asymptotically as small as that of the best (for the given process measure generating the data) predictor in $\mathcal{C}$ (we call this Problem 3). Thus, we now have three problems about a set of process measures $\mathcal{C}$ to address.

We show that if the quality of prediction is measured in total variation then all the three problems coincide: any solution to any one of them is a solution to the other two. For the case of expected average KL divergence, all the three problems are different: the realizable case is strictly easier than non-realizable (Problem 2), which is, in turn, strictly easier than the fully agnostic case (Problem 3). We then analyse which results concerning prediction in total variation can be transferred to which of the problems concerning prediction in average KL divergence. It was shown in Ryabko (2010a) that, for the realizable case, if there is a solution for a given set of process measures $\mathcal{C}$, then a solution can also be obtained as a Bayesian mixture over a countable subset of $\mathcal{C}$; this holds both for prediction in total variation and in expected average KL divergence. Here we show that this result also holds true for the (non-realizable) case of Problem 2, for prediction in expected average KL divergence. We do not have an analogous result for Problem 3 (and, in fact, conjecture that the opposite statement holds true). However, for the fully agnostic case of Problem 3, we show that separability with respect to a certain topology given by KL divergence is a sufficient (though not a necessary) condition for the existence of a predictor. This is used to demonstrate that there is a solution to this problem for the set of all finite-memory process measures, complementing similar results obtained earlier in different settings. On the other hand, we show that there is no solution to this problem for the set of all stationary process measures, in contrast to a result of B. Ryabko (1988) that gives a solution to the realizable case of this problem (that is, a predictor whose expected average KL error goes to zero if any stationary process is chosen to generate the data). Finally, we also consider a modified version of Problem 3, in which the performance of predictors is only compared on individual sequences. For this problem, we obtain, using a result from (Ryabko, 1986), a characterisation of those sets $\mathcal{C}$ for which a solution exists in terms of the Hausdorff dimension.

2. Notation and Definitions

Let $\mathcal{X}$ be a finite set. The notation $x_{1..n}$ is used for $x_1,\dots,x_n$. We consider stochastic processes (probability measures) on $\Omega := (\mathcal{X}^\infty, \mathcal{B})$ where $\mathcal{B}$ is the sigma-field generated by the cylinder sets $[x_{1..n}], x_i \in \mathcal{X}, n \in \mathbb{N}$ ($[x_{1..n}]$ is the set of all infinite sequences that start with $x_{1..n}$). For a finite set A denote $|A|$ its cardinality. We use $\mathbf{E}_\mu$ for expectation with respect to a measure μ.

Next we introduce the measures of the quality of prediction used in this paper. For two measures μ and ρ we are interested in how different the μ- and ρ-conditional probabilities are, given a data sample $x_{1..n}$. Introduce the *(conditional) total variation* distance

$$v(\mu,\rho,x_{1..n}) := \sup_{A\in\mathcal{B}} |\mu(A|x_{1..n}) - \rho(A|x_{1..n})|,$$

if $\mu(x_{1..n}) \neq 0$ and $\rho(x_{1..n}) \neq 0$, and $v(\mu,\rho,x_{1..n}) = 1$ otherwise.

Definition 1 *We say that* ρ *predicts* μ *in total variation if*

$$v(\mu,\rho,x_{1..n}) \to 0 \ \mu\text{-}a.s.$$

This convergence is rather strong. In particular, it means that ρ-conditional probabilities of arbitrary far-off events converge to μ-conditional probabilities. Moreover, ρ predicts μ in total variation if (Blackwell and Dubins, 1962) and only if (Kalai and Lehrer, 1994) μ is absolutely continuous with respect to ρ. Denote $\geq_{tv}$ the relation of absolute continuity (that is, $\rho \geq_{tv} \mu$ if μ is absolutely continuous with respect to ρ).

Thus, for a class $\mathcal{C}$ of measures there is a predictor ρ that predicts every $\mu \in \mathcal{C}$ in total variation if and only if every $\mu \in \mathcal{C}$ has a density with respect to ρ. Although such sets of processes are rather large, they do not include even such basic examples as the set of all Bernoulli i.i.d. processes. That is, there is no ρ that would predict in total variation every Bernoulli i.i.d. process measure δ_p, $p \in [0,1]$, where p is the probability of 0. Indeed, all these processes δ_p, $p \in [0,1]$, are singular with respect to one another; in particular, each of the non-overlapping sets T_p of all sequences which have limiting fraction p of 0s has probability 1 with respect to one of the measures and 0 with respect to all others; since there are uncountably many of these measures, there is no measure ρ with respect to which they all would have a density (since such a measure should have $\rho(T_p) > 0$ for all p).

Therefore, perhaps for many (if not most) practical applications this measure of the quality of prediction is too strong, and one is interested in weaker measures of performance.

For two measures μ and ρ introduce the *expected cumulative Kullback-Leibler divergence (KL divergence)* as

$$d_n(\mu,\rho) := \mathbf{E}_\mu \sum_{t=1}^{n} \sum_{a\in\mathcal{X}} \mu(x_t = a|x_{1..t-1}) \log \frac{\mu(x_t = a|x_{1..t-1})}{\rho(x_t = a|x_{1..t-1})},$$

In words, we take the expected (over data) cumulative (over time) KL divergence between μ- and ρ-conditional (on the past data) probability distributions of the next outcome.

Definition 2 *We say that* ρ *predicts* μ *in expected average KL divergence if*

$$\frac{1}{n} d_n(\mu,\rho) \to 0.$$

This measure of performance is much weaker, in the sense that it requires good predictions only one step ahead, and not on every step but only on average; also the convergence is not with probability 1 but in expectation. With prediction quality so measured, predictors exist for relatively large classes of measures; most notably, Ryabko (1988) provides a predictor which predicts every stationary process in expected average KL divergence.

We will use the following well-known identity (introduced, in the context of sequence prediction, by Ryabko, 1988)

$$d_n(\mu,\rho) = -\sum_{x_{1..n}\in\mathcal{X}^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})},$$

where on the right-hand side we have simply the KL divergence between measures μ and ρ restricted to the first n observations.

Thus, the results of this work will be established with respect to two very different measures of prediction quality, one of which is very strong and the other rather weak. This suggests that the facts established reflect some fundamental properties of the problem of prediction, rather than those pertinent to particular measures of performance. On the other hand, it remains open to extend the results below to different measures of performance.

Definition 3 *Consider the following classes of process measures: $\mathcal{P}$ is the set of all process measures, $\mathcal{D}$ is the set of all degenerate discrete process measures, $\mathcal{S}$ is the set of all stationary processes and $\mathcal{M}_k$ is the set of all stationary measures with memory not greater than k (k-order Markov processes, with $\mathcal{M}_0$ being the set of all i.i.d. processes):*

$$\mathcal{D} := \{\mu \in \mathcal{P} : \exists x \in \mathcal{X}^\infty \;\; \mu(x) = 1\},$$

$$\mathcal{S} := \{\mu \in \mathcal{P} : \forall n,k \geq 1\, \forall a_{1..n} \in \mathcal{X}^n \mu(x_{1..n} = a_{1..n}) = \mu(x_{1+k..n+k} = a_{1..n})\}.$$

$$\mathcal{M}_k := \{\mu \in \mathcal{S} : \forall n \geq k \forall a \in \mathcal{X} \forall a_{1..n} \in \mathcal{X}^n$$
$$\mu(x_{n+1} = a | x_{1..n} = a_{1..n}) = \mu(x_{k+1} = a | x_{1..k} = a_{n-k+1..n})\}.$$

Abusing the notation, we will sometimes use elements of $\mathcal{D}$ and $\mathcal{X}^\infty$ interchangeably. The following (simple and well-known) statement will be used repeatedly in the examples.

Lemma 4 *For every $\rho \in \mathcal{P}$ there exists $\mu \in \mathcal{D}$ such that $d_n(\mu,\rho) \geq n \log|\mathcal{X}|$ for all $n \in \mathbb{N}$.*

Proof Indeed, for each n we can select $\mu(x_n = a) = 1$ for $a \in \mathcal{X}$ that minimizes $\rho(x_n = a | x_{1..n-1})$, so that $\rho(x_{1..n}) \leq |\mathcal{X}|^{-n}$. ■

3. Sequence Prediction Problems

For the two notions of predictive quality introduced, we can now state formally the sequence prediction problems.

Problem 1(realizable case). Given a set of probability measures $\mathcal{C}$, find a measure ρ such that ρ predicts in total variation (expected average KL divergence) every $\mu \in \mathcal{C}$, if such a ρ exists.

Thus, Problem 1 is about finding a predictor for the case when the process generating the data is known to belong to a given class $\mathcal{C}$. That is, the set $\mathcal{C}$ here is a set of measures that generate the data. Next let us formulate the questions about $\mathcal{C}$ as a set of predictors.

Problem 2 (non-realizable case). Given a set of process measures (predictors) $\mathcal{C}$, find a process measure ρ such that ρ predicts in total variation (in expected average KL divergence) every measure $\nu \in \mathcal{P}$ such that there is $\mu \in \mathcal{C}$ which predicts (in the same sense) ν.

While Problem 2 is already quite general, it does not yet address what can be called the fully agnostic case: if nothing at all is known about the process ν generating the data, it means that there may be no $\mu \in \mathcal{C}$ such that μ predicts ν, and then, even if we have a solution ρ to the Problem 2, we still do not know what the performance of ρ is going to be on the data generated by ν, compared to the performance of the predictors from $\mathcal{C}$. To address this fully agnostic case we have to introduce the notion of loss.

Definition 5 *Introduce the almost sure total variation loss of* ρ *with respect to* μ

$$l_{tv}(\mu,\rho) := \inf\{\alpha \in [0,1] : \limsup_{n\to\infty} v(\mu,\rho,x_{1..n}) \leq \alpha\ \mu\text{–}a.s.\},$$

and the asymptotic KL loss

$$l_{KL}(\nu,\rho) := \limsup_{n\to\infty} \frac{1}{n} d_n(\nu,\rho).$$

We can now formulate the fully agnostic version of the sequence prediction problem.

Problem 3. Given a set of process measures (predictors) $\mathcal{C}$, find a process measure ρ such that ρ predicts at least as well as any μ in $\mathcal{C}$, if any process measure $\nu \in \mathcal{P}$ is chosen to generate the data:

$$l(\nu,\rho) - l(\nu,\mu) \leq 0$$

for every $\nu \in \mathcal{P}$ and every $\mu \in \mathcal{C}$, where $l(\cdot,\cdot)$ is either $l_{tv}(\cdot,\cdot)$ or $l_{KL}(\cdot,\cdot)$.

The three problems just formulated represent different conceptual approaches to the sequence prediction problem. Let us illustrate the difference by the following **informal example**. Suppose that the set $\mathcal{C}$ is that of all (ergodic, finite-state) Markov chains. Markov chains being a familiar object in probability and statistics, we can easily construct a predictor ρ that predicts every $\mu \in \mathcal{C}$ (for example, in expected average KL divergence, see Krichevsky, 1993). That is, if we know that the process μ generating the data is Markovian, we know that our predictor is going to perform well. This is the realizable case of Problem 1. In reality, rarely can we be sure that the Markov assumption holds true for the data at hand. We may believe, however, that it is still a reasonable assumption, in the sense that there is a Markovian model which, for our purposes (for the purposes of prediction), is a good model of the data. Thus we may assume that there is a Markov model (a predictor) that predicts well the process that we observe, and we would like to combine the predictive qualities of all these Markov models. This is the "non-realizable" case of Problem 2. Note that this problem is more difficult than the first one; in particular, a process ν generating the data may be singular with respect to any Markov process, and still be predicted well (in the sense of expected average KL divergence, for example) by some of them. Still, here we are making some assumptions about the process generating the data, and, if these assumptions are wrong, then we do not know anything about the performance of our predictor. Thus, we may ultimately wish to acknowledge that we do not know anything at all about the data; we still know a lot about Markov processes, and we would like to use this knowledge on our data. If there is anything at all Markovian in it (that is, anything that can be captured by a Markov model), then we would like our predictor to use it. In other words, we want to have a predictor that predicts any process measure whatsoever (at least) as well as any Markov predictor. This is the "fully agnostic" case of Problem 3.

Of course, Markov processes were just mentioned as an example, while in this work we are only concerned with the most general case of arbitrary (uncountable) sets $\mathcal{C}$ of process measures.

The following statement is rather obvious.

Proposition 6 *Any solution to Problem 3 is a solution to Problem 2, and any solution to Problem 2 is a solution to Problem 1.*

Despite the conceptual differences in formulations, it may be somewhat unclear whether the three problems are indeed different. It appears that this depends on the measure of predictive quality chosen: for the case of prediction in total variation distance all the three problems coincide, while for the case of prediction in expected average KL divergence they are different.

4. Prediction in Total Variation

As it was mentioned, a measure μ is absolutely continuous with respect to a measure ρ if and only if ρ predicts μ in total variation distance. This reduces studying at least Problem 1 for total variation distance to studying the relation of absolute continuity. Introduce the notation $\rho \geq_{tv} \mu$ for this relation.

Let us briefly recall some facts we know about $\geq_{tv}$; details can be found, for example, in Plesner and Rokhlin (1946). Let $[\mathcal{P}]_{tv}$ denote the set of equivalence classes of $\mathcal{P}$ with respect to $\geq_{tv}$, and for $\mu \in \mathcal{P}_{tv}$ denote $[\mu]$ the equivalence class that contains μ. Two elements $\sigma_1, \sigma_2 \in [\mathcal{P}]_{tv}$ (or $\sigma_1, \sigma_2 \in \mathcal{P}$) are called disjoint (or singular) if there is no $\nu \in [\mathcal{P}]_{tv}$ such that $\sigma_1 \geq_{tv} \nu$ and $\sigma_2 \geq_{tv} \nu$; in this case we write $\sigma_1 \perp_{tv} \sigma_2$. We write $[\mu_1] + [\mu_2]$ for $[\frac{1}{2}(\mu_1 + \mu_2)]$. Every pair $\sigma_1, \sigma_2 \in [\mathcal{P}]_{tv}$ has a supremum $\sup(\sigma_1, \sigma_2) = \sigma_1 + \sigma_2$. Introducing into $[\mathcal{P}]_{tv}$ an extra element 0 such that $\sigma \geq_{tv} 0$ for all $\sigma \in [\mathcal{P}]_{tv}$, we can state that for every $\rho, \mu \in [\mathcal{P}]_{tv}$ there exists a unique pair of elements μ_s and μ_a such that $\mu = \mu_a + \mu_s$, $\rho \geq \mu_a$ and $\rho \perp_{tv} \mu_s$. (This is a form of Lebesgue decomposition.) Moreover, $\mu_a = \inf(\rho, \mu)$. Thus, every pair of elements has a supremum and an infimum. Moreover, every bounded set of disjoint elements of $[\mathcal{P}]_{tv}$ is at most countable.

Furthermore, we introduce the (unconditional) total variation distance between process measures.

Definition 7 (unconditional total variation distance) *The (unconditional) total variation distance is defined as*

$$v(\mu, \rho) := \sup_{A \in \mathcal{B}} |\mu(A) - \rho(A)|.$$

Known characterizations of those sets C that are bounded with respect to $\geq_{tv}$ can now be related to our prediction problems 1-3 as follows.

Theorem 8 *Let $C \subset \mathcal{P}$. The following statements about C are equivalent.*

(i) There exists a solution to Problem 1 in total variation.

(ii) There exists a solution to Problem 2 in total variation.

(iii) There exists a solution to Problem 3 in total variation.

(iv) C is upper-bounded with respect to $\geq_{tv}$.

(v) There exists a sequence $\mu_k \in C$, $k \in \mathbb{N}$ such that for some (equivalently, for every) sequence of weights $w_k \in (0,1]$, $k \in \mathbb{N}$ such that $\sum_{k\in\mathbb{N}} w_k = 1$, the measure $\nu = \sum_{k\in\mathbb{N}} w_k \mu_k$ satisfies $\nu \geq_{tv} \mu$ for every $\mu \in C$.

(vi) C is separable with respect to the total variation distance.

(vii) Let $\mathcal{C}^+ := \{\mu \in \mathcal{P} : \exists \rho \in \mathcal{C}\, \rho \geq_{tv} \mu\}$. Every disjoint (with respect to $\geq_{tv}$) subset of $\mathcal{C}^+$ is at most countable.

Moreover, every solution to any of the Problems 1-3 is a solution to the other two, as is any upper bound for $\mathcal{C}$. The sequence μ_k in the statement (v) can be taken to be any dense (in the total variation distance) countable subset of $\mathcal{C}$ (cf. (vi)), or any maximal disjoint (with respect to $\geq_{tv}$) subset of $\mathcal{C}^+$ of statement (vii), in which every measure that is not in $\mathcal{C}$ is replaced by any measure from $\mathcal{C}$ that dominates it.

Proof The implications $(i) \Leftarrow (ii) \Leftarrow (iii)$ are obvious (cf. Proposition 6). The implication $(iv) \Rightarrow (i)$ is a reformulation of the result of Blackwell and Dubins (1962). The converse (and hence $(v) \Rightarrow (iv)$) was established in Kalai and Lehrer (1994). $(i) \Rightarrow (ii)$ follows from the equivalence $(i) \Leftrightarrow (iv)$ and the transitivity of $\geq_{tv}$; $(i) \Rightarrow (iii)$ follows from the transitivity of $\geq_{tv}$ and from Lemma 9 below: indeed, from Lemma 9 we have $l_{tv}(\nu,\mu) = 0$ if $\mu \geq_{tv} \nu$ and $l_{tv}(\nu,\mu) = 1$ otherwise. From this and the transitivity of $\geq_{tv}$ it follows that if $\rho \geq_{tv} \mu$ then also $l_{tv}(\nu,\rho) \leq l_{tv}(\nu,\mu)$ for all $\nu \in \mathcal{P}$. The equivalence of (v), (vi), and (i) was established in Ryabko (2010a). The equivalence of (iv) and (vii) was proven in Plesner and Rokhlin (1946). The concluding statements of the theorem are easy to demonstrate from the results cited above. ■

The following lemma is an easy consequence of Blackwell and Dubins (1962).

Lemma 9 *Let μ, ρ be two process measures. Then $v(\mu,\rho,x_{1..n})$ converges to either 0 or 1 with μ-probability 1.*

Proof Assume that μ is not absolutely continuous with respect to ρ (the other case is covered by Blackwell and Dubins, 1962). By Lebesgue decomposition theorem, the measure μ admits a representation $\mu = \alpha\mu_a + (1-\alpha)\mu_s$ where $\alpha \in [0,1]$ and the measures μ_a and μ_s are such that μ_a is absolutely continuous with respect to ρ and μ_s is singular with respect to ρ. Let W be such a set that $\mu_a(W) = \rho(W) = 1$ and $\mu_s(W) = 0$. Note that we can take $\mu_a = \mu|_W$ and $\mu_s = \mu|_{\mathcal{X}^\infty\backslash W}$. From Blackwell and Dubins (1962) we have $v(\mu_a,\rho,x_{1..n}) \to 0$ μ_a-a.s., as well as $v(\mu_a,\mu,x_{1..n}) \to 0$ μ_a-a.s. and $v(\mu_s,\mu,x_{1..n}) \to 0$ μ_s-a.s. Moreover, $v(\mu_s,\rho,x_{1..n}) \geq |\mu_s(W|x_{1..n}) - \rho(W|x_{1..n})| = 1$ so that $v(\mu_s,\rho,x_{1..n}) \to 1$ μ_s-a.s. Furthermore,

$$v(\mu,\rho,x_{1..n}) \leq v(\mu,\mu_a,x_{1..n}) + v(\mu_a,\rho,x_{1..n}) = I$$

and

$$v(\mu,\rho,x_{1..n}) \geq -v(\mu,\mu_s,x_{1..n}) + v(\mu_s,\rho,x_{1..n}) = II.$$

We have $I \to 0$ μ_a-a.s. and hence $\mu|_W$-a.s., as well as $II \to 1$ μ_s-a.s. and hence $\mu|_{\mathcal{X}^\infty\backslash W}$-a.s. Thus, $\mu(v(\mu,\rho,x_{1..n}) \to 0 \text{ or } 1) \leq \mu(W)\mu|_W(I \to 0) + \mu(\mathcal{X}^\infty\backslash W)\mu|_{\mathcal{X}^\infty\backslash W}(II \to 1) = \mu(W) + \mu(\mathcal{X}^\infty\backslash W) = 1$, which concludes the proof. ■

Remark. Using Lemma 9 we can also define *expected* (rather than almost sure) total variation loss of ρ with respect to μ, as the μ-probability that $v(\mu,\rho)$ converges to 1:

$$l'_{tv}(\mu,\rho) := \mu\{x_1,x_2,\dots \in \mathcal{X}^\infty : v(\mu,\rho,x_{1..n}) \to 1\}.$$

Then Problem 3 can be reformulated for this notion of loss. However, it is easy to see that for this reformulation Theorem 8 holds true as well.

Thus, we can see that, for the case of prediction in total variation, all the sequence prediction problems formulated reduce to studying the relation of absolute continuity for process measures and those families of measures that are absolutely continuous (have a density) with respect to some measure (a predictor). On the one hand, from a statistical point of view such families are rather large: the assumption that the probabilistic law in question has a density with respect to some (nice) measure is a standard one in statistics. It should also be mentioned that such families can easily be uncountable. (In particular, this means that they are large from a computational point of view.) On the other hand, even such basic examples as the set of all Bernoulli i.i.d. measures does not allow for a predictor that predicts every measure in total variation (as explained in Section 2).

That is why we have to consider weaker notions of predictions; from these, prediction in expected average KL divergence is perhaps one of the weakest. The goal of the next sections is to see which of the properties that we have for total variation can be transferred (and in which sense) to the case of expected average KL divergence.

5. Prediction in Expected Average KL Divergence

First of all, we have to observe that for prediction in KL divergence Problems 1, 2, and 3 are different, as the following theorem shows. While the examples provided in the proof are artificial, there is a very important example illustrating the difference between Problem 1 and Problem 3 for expected average KL divergence: the set $\mathcal{S}$ of all stationary processes, given in Theorem 16 in the end of this section.

Theorem 10 *For the case of prediction in expected average KL divergence, Problems 1, 2 and 3 are different: there exists a set $\mathcal{C}_1 \subset \mathcal{P}$ for which there is a solution to Problem 1 but there is no solution to Problem 2, and there is a set $\mathcal{C}_2 \subset \mathcal{P}$ for which there is a solution to Problem 2 but there is no solution to Problem 3.*

Proof We have to provide two examples. Fix the binary alphabet $\mathcal{X} = \{0,1\}$. For each deterministic sequence $t = t_1, t_2, \dots \in \mathcal{X}^\infty$ construct the process measure γ_t as follows: $\gamma_t(x_n = t_n | t_{1..n-1}) := 1 - \frac{1}{n+1}$ and for $x_{1..n-1} \neq t_{1..n-1}$ let $\gamma_t(x_n = 0|x_{1..n-1}) = 1/2$, for all $n \in \mathbb{N}$. That is, γ_t is Bernoulli i.i.d. 1/2 process measure strongly biased towards a specific deterministic sequence, t. Let also $\gamma(x_{1..n}) = 2^{-n}$ for all $x_{1..n} \in \mathcal{X}^n$, $n \in \mathbb{N}$ (the Bernoulli i.i.d. 1/2). For the set $\mathcal{C}_1 := \{\gamma_t : t \in \mathcal{X}^\infty\}$ we have a solution to Problem 1: indeed, $d_n(\gamma_t, \gamma) \leq 1 = o(n)$. However, there is no solution to Problem 2. Indeed, for each $t \in \mathcal{D}$ we have $d_n(t, \gamma_t) = \log n = o(n)$ (that is, for every deterministic measure there is an element of $\mathcal{C}_1$ which predicts it), while by Lemma 4 for every $\rho \in \mathcal{P}$ there exists $t \in \mathcal{D}$ such that $d_n(t, \rho) \geq n$ for all $n \in \mathbb{N}$ (that is, there is no predictor which predicts every measure that is predicted by at least one element of $\mathcal{C}_1$).

The second example is similar. For each deterministic sequence $t = t_1, t_2, \dots \in \mathcal{D}$ construct the process measure γ_t as follows: $\gamma'_t(x_n = t_n | t_{1..n-1}) := 2/3$ and for $x_{1..n-1} \neq t_{1..n-1}$ let $\gamma'_t(x_n = 0|x_{1..n-1}) = 1/2$, for all $n \in \mathbb{N}$. It is easy to see that γ is a solution to Problem 2 for the set $\mathcal{C}_2 := \{\gamma'_t : t \in \mathcal{X}^\infty\}$. Indeed, if $\nu \in \mathcal{P}$ is such that $d_n(\nu, \gamma') = o(n)$ then we must have $\nu(t_{1..n}) = o(1)$. From this and the fact that γ and γ' coincide (up to $O(1)$) on all other sequences we conclude $d_n(\nu, \gamma) = o(n)$. However, there is no solution to Problem 3 for $\mathcal{C}_2$. Indeed, for every $t \in \mathcal{D}$ we have $d_n(t, \gamma'_t) = n \log 3/2 + o(n)$. Therefore, if ρ is a solution to Problem 3 then $\limsup \frac{1}{n} d_n(t, \rho) \leq$

$\log 3/2 < 1$ which contradicts Lemma 4. ■

Thus, prediction in expected average KL divergence turns out to be a more complicated matter than prediction in total variation. The next idea is to try and see which of the facts about prediction in total variation can be generalized to some of the problems concerning prediction in expected average KL divergence.

First, observe that, for the case of prediction in total variation, the equivalence of Problems 1 and 2 was derived from the transitivity of the relation $\geq_{tv}$ of absolute continuity. For the case of expected average KL divergence, the relation "ρ predicts μ in expected average KL divergence" is not transitive (and Problems 1 and 2 are not equivalent). However, for Problem 2 we are interested in the following relation: ρ "dominates" μ if ρ predicts every ν such that μ predicts ν. Denote this relation by $\geq_{KL}$:

Definition 11 ($\geq_{KL}$) *We write $\rho \geq_{KL} \mu$ if for every $\nu \in \mathcal{P}$ the equality $\limsup \frac{1}{n} d_n(\nu,\mu) = 0$ implies $\limsup \frac{1}{n} d_n(\nu,\rho) = 0$.*

The relation $\geq_{KL}$ has some similarities with $\geq_{tv}$. First of all, $\geq_{KL}$ is also transitive (as can be easily seen from the definition). Moreover, similarly to $\geq_{tv}$, one can show that for any μ,ρ any strictly convex combination $\alpha\mu + (1-\alpha)\rho$ is a supremum of $\{\rho,\mu\}$ with respect to $\geq_{KL}$. Next we will obtain a characterization of predictability with respect to $\geq_{KL}$ similar to one of those obtained for $\geq_{tv}$.

The key observation is the following. If there is a solution to Problem 2 for a set $\mathcal{C}$ then a solution can be obtained as a Bayesian mixture over a countable subset of $\mathcal{C}$. For total variation this is the statement (v) of Theorem 8.

Theorem 12 *Let $\mathcal{C}$ be a set of probability measures on Ω. If there is a measure ρ such that $\rho \geq_{KL} \mu$ for every $\mu \in \mathcal{C}$ (ρ is a solution to Problem 2) then there is a sequence $\mu_k \in \mathcal{C}$, $k \in \mathbb{N}$, such that $\sum_{k\in\mathbb{N}} w_k\mu_k \geq_{KL} \mu$ for every $\mu \in \mathcal{C}$, where w_k are some positive weights.*

The proof is deferred to Section 7. An analogous result for Problem 1 was established in Ryabko (2009). (The proof of Theorem 12 is based on similar ideas, but is more involved.)

For the case of Problem 3, we do not have results similar to Theorem 12 (or statement (v) of Theorem 8); in fact, we conjecture that the opposite is true: there exists a (measurable) set $\mathcal{C}$ of measures such that there is a solution to Problem 3 for $\mathcal{C}$, but there is no Bayesian solution to Problem 3, meaning that there is no probability distribution on $\mathcal{C}$ (discrete or not) such that the mixture over $\mathcal{C}$ with respect to this distribution is a solution to Problem 3 for $\mathcal{C}$.

However, we can take a different route and extend another part of Theorem 8 to obtain a characterization of sets $\mathcal{C}$ for which a solution to Problem 3 exists.

We have seen that, in the case of prediction in total variation, separability with respect to the topology of this distance is a necessary and sufficient condition for the existence of a solution to Problems 1-3. In the case of expected average KL divergence the situation is somewhat different, since, first of all, (asymptotic average) KL divergence is not a metric. While one can introduce a topology based on it, separability with respect to this topology turns out to be a sufficient but not a necessary condition for the existence of a predictor, as is shown in the next theorem.

Definition 13 *Define the distance $d_\infty(\mu_1,\mu_2)$ on process measures as follows*

$$d_\infty(\mu_1,\mu_2)=\limsup_{n\to\infty}\sup_{x_{1..n}\in\mathcal{X}^n}\frac{1}{n}\left|\log\frac{\mu_1(x_{1..n})}{\mu_2(x_{1..n})}\right|,$$

where we assume $\log 0/0:=0$.

Clearly, d_∞ is symmetric and satisfies the triangle inequality, but it is not exact. Moreover, for every μ_1,μ_2 we have

$$\limsup_{n\to\infty}\frac{1}{n}d_n(\mu_1,\mu_2)\le d_\infty(\mu_1,\mu_2).$$

The distance $d_\infty(\mu_1,\mu_2)$ measures the difference in behaviour of μ_1 and μ_2 on all individual sequences. Thus, using this distance to analyse Problem 3 is most close to the traditional approach to the non-realizable case, which is formulated in terms of predicting individual deterministic sequences.

Theorem 14 *(i) Let $\mathcal{C}$ be a set of process measures. If $\mathcal{C}$ is separable with respect to d_∞ then there is a solution to Problem 3 for $\mathcal{C}$, for the case of prediction in expected average KL divergence.*

(ii) There exists a set of process measures $\mathcal{C}$ such that $\mathcal{C}$ is not separable with respect to d_∞, but there is a solution to Problem 3 for this set, for the case of prediction in expected average KL divergence.

Proof For the first statement, let $\mathcal{C}$ be separable and let $(\mu_k)_{k\in\mathbb{N}}$ be a dense countable subset of $\mathcal{C}$. Define $\nu:=\sum_{k\in\mathbb{N}}w_k\mu_k$, where w_k are any positive summable weights. Fix any measure τ and any $\mu\in\mathcal{C}$. We will show that $\limsup_{n\to\infty}\frac{1}{n}d_n(\tau,\nu)\le\limsup_{n\to\infty}\frac{1}{n}d_n(\tau,\mu)$. For every ε, find such a $k\in\mathbb{N}$ that $d_\infty(\mu,\mu_k)\le\varepsilon$. We have

$$\begin{aligned}d_n(\tau,\nu)\le d_n(\tau,w_k\mu_k)&=\mathbf{E}_\tau\log\frac{\tau(x_{1..n})}{\mu_k(x_{1..n})}-\log w_k\\&=\mathbf{E}_\tau\log\frac{\tau(x_{1..n})}{\mu(x_{1..n})}+\mathbf{E}_\tau\log\frac{\mu(x_{1..n})}{\mu_k(x_{1..n})}-\log w_k\\&\le d_n(\tau,\mu)+\sup_{x_{1..n}\in\mathcal{X}^n}\log\left|\frac{\mu(x_{1..n})}{\mu_k(x_{1..n})}\right|-\log w_k.\end{aligned}$$

From this, dividing by n taking $\limsup_{n\to\infty}$ on both sides, we conclude

$$\limsup_{n\to\infty}\frac{1}{n}d_n(\tau,\nu)\le\limsup_{n\to\infty}\frac{1}{n}d_n(\tau,\mu)+\varepsilon.$$

Since this holds for every $\varepsilon>0$ the first statement is proven.

The second statement is proven by the following example. Let $\mathcal{C}$ be the set of all deterministic sequences (measures concentrated on just one sequence) such that the number of 0s in the first n symbols is less than $\sqrt{n}$, for all $n\in\mathbb{N}$. Clearly, this set is uncountable. It is easy to check that $\mu_1\ne\mu_2$ implies $d_\infty(\mu_1,\mu_2)=\infty$ for every $\mu_1,\mu_2\in\mathcal{C}$, but the predictor ν, given by $\nu(x_n=0)=1/n$ independently for different n, predicts every $\mu\in\mathcal{C}$ in expected average KL divergence. Since all elements of $\mathcal{C}$ are deterministic, ν is also a solution to Problem 3 for $\mathcal{C}$. ■

Although simple, Theorem 14 can be used to establish the existence of a solution to Problem 3 for an important class of process measures: that of all processes with finite memory, as the next theorem shows. Results similar to Theorem 15 are known in different settings, for example, Ziv and Lempel (1978), Ryabko (1984), Cesa-Bianchi and Lugosi (1999) and others.

Theorem 15 *There exists a solution to Problem 3 for prediction in expected average KL divergence for the set of all finite-memory process measures* $\mathcal{M} := \cup_{k\in\mathbb{N}}\mathcal{M}_k$.

Proof We will show that the set $\mathcal{M}$ is separable with respect to d_∞. Then the statement will follow from Theorem 14. It is enough to show that each set $\mathcal{M}_k$ is separable with respect to d_∞.

For simplicity, assume that the alphabet is binary ($|\mathcal{X}| = 2$; the general case is analogous). Observe that the family $\mathcal{M}_k$ of k-order stationary binary-valued Markov processes is parametrized by $|\mathcal{X}|^k$ $[0,1]$-valued parameters: probability of observing 0 after observing $x_{1..k}$, for each $x_{1..k} \in \mathcal{X}^k$. Note that this parametrization is continuous (as a mapping from the parameter space with the Euclidean topology to $\mathcal{M}_k$ with the topology of d_∞). Indeed, for any $\mu_1,\mu_2 \in \mathcal{M}_k$ and every $x_{1..n} \in \mathcal{X}^n$ such that $\mu_i(x_{1..n}) \neq 0, i = 1,2$, it is easy to see that

$$\frac{1}{n}\left|\log\frac{\mu_1(x_{1..n})}{\mu_2(x_{1..n})}\right| \leq \sup_{x_{1..k+1}} \frac{1}{k+1}\left|\log\frac{\mu_1(x_{1..k+1})}{\mu_2(x_{1..k+1})}\right|, \tag{1}$$

so that the right-hand side of (1) also upper-bounds $d_\infty(\mu_1,\mu_2)$, implying continuity of the parametrization.

It follows that the set μ^k_q, $q \in Q^{|\mathcal{X}|^k}$ of all stationary k-order Markov processes with rational values of all the parameters ($Q := \mathbb{Q} \cap [0,1]$) is dense in $\mathcal{M}_k$, proving the separability of the latter set. ■

Another important example is the set of all stationary process measures $\mathcal{S}$. This example also illustrates the difference between the prediction problems that we consider. For this set a solution to Problem 1 was given in Ryabko (1988). In contrast, here we show that there is no solution to Problem 3 for $\mathcal{S}$.

Theorem 16 *There is no solution to Problem 3 for the set of all stationary processes* $\mathcal{S}$.

Proof This proof is based on the construction similar to the one used in Ryabko (1988) to demonstrate impossibility of consistent prediction of stationary processes without Cesaro averaging.

Let m be a Markov chain with states $0,1,2,\dots$ and state transitions defined as follows. From each sate $k \in \mathbb{N}\cup\{0\}$ the chain passes to the state $k+1$ with probability 2/3 and to the state 0 with probability 1/3. It is easy to see that this chain possesses a unique stationary distribution on the set of states (see, e.g., Shiryaev, 1996); taken as the initial distribution it defines a stationary ergodic process with values in $\mathbb{N}\cup\{0\}$. Fix the ternary alphabet $\mathcal{X} = \{a,0,1\}$. For each sequence $t = t_1,t_2,\dots \in \{0,1\}^\infty$ define the process μ_t as follows. It is a deterministic function of the chain m. If the chain is in the state 0 then the process μ_t outputs a; if the chain m is in the state $k > 0$ then the process outputs t_k. That is, we have defined a hidden Markov process which in the state 0 of the underlying Markov chain always outputs a, while in other states it outputs either 0 or 1 according to the sequence t.

To show that there is no solution to Problem 3 for $\mathcal{S}$, we will show that there is no solution to Problem 3 for the smaller set $\mathcal{C} := \{\mu_t : t \in \{0,1\}^\infty\}$. Indeed, for any $t \in \{0,1\}^\infty$ we have $d_n(t,\mu_t) = n\log 3/2 + o(n)$. Then if ρ is a solution to Problem 3 for $\mathcal{C}$ we should have $\limsup_{n\to\infty} \frac{1}{n} d_n(t,\rho) \le \log 3/2 < 1$ for every $t \in \mathcal{D}$, which contradicts Lemma 4. ■

From the proof of Theorem 16 one can see that, in fact, the statement that is proven is stronger: there is no solution to Problem 3 for the set of all functions of stationary ergodic countable-state Markov chains. We conjecture that a solution to Problem 2 exists for the latter set, but not for the set of all stationary processes.

As we have seen in the statements above, the set of all deterministic measures $\mathcal{D}$ plays an important role in the analysis of the predictors in the sense of Problem 3. Therefore, an interesting question is to characterize those sets $\mathcal{C}$ of measures for which there is a predictor ρ that predicts *every individual sequence* at least as well as any measure from $\mathcal{C}$. Such a characterization can be obtained in terms of Hausdorff dimension, using a result of Ryabko (1986), that shows that Hausdorff dimension of a set characterizes the optimal prediction error that can be attained by any predictor.

For a set $A \subset \mathcal{X}^\infty$ denote $H(A)$ its Hausdorff dimension (see, for example, Billingsley, 1965 for its definition).

Theorem 17 *Let $\mathcal{C} \subset \mathcal{P}$. The following statements are equivalent.*

(i) There is a measure $\rho \in \mathcal{P}$ that predicts every individual sequence at least as well as the best measure from $\mathcal{C}$: for every $\mu \in \mathcal{C}$ and every sequence $x_1, x_2, \dots \in \mathcal{X}^\infty$ we have

$$\liminf_{n\to\infty} -\frac{1}{n}\log\rho(x_{1..n}) \le \liminf_{n\to\infty} -\frac{1}{n}\log\mu(x_{1..n}).$$

(ii) For every $\alpha \in [0,1]$ the Hausdorff dimension of the set of sequences on which the average prediction error of the best measure in $\mathcal{C}$ is not greater than α is bounded by $\alpha/\log|\mathcal{X}|$:

$$H(\{x_1, x_2, \dots \in \mathcal{X}^\infty : \inf_{\mu\in\mathcal{C}} \liminf_{n\to\infty} -\frac{1}{n}\log\mu(x_{1..n}) \le \alpha\}) \le \alpha/\log|\mathcal{X}|.$$

Proof The implication $(i) \Rightarrow (ii)$ follows directly from Ryabko (1986) where it is shown that for every measure ρ one must have $H(\{x_1,x_2,\dots \in \mathcal{X}^\infty : \liminf_{n\to\infty} -\frac{1}{n}\log\rho(x_{1..n}) \le \alpha\}) \le \alpha/\log|\mathcal{X}|$.

To show the opposite implication, we again refer to Ryabko (1986): for every set $A \subset \mathcal{X}^\infty$ there is a measure ρ_A such that

$$\liminf_{n\to\infty} -\frac{1}{n}\log\rho_A(x_{1..n}) \le H(A)\log|\mathcal{X}|. \tag{2}$$

For each $\alpha \in [0,1]$ define $A_\alpha := \{x_1,x_2,\dots \in \mathcal{X}^\infty : \inf_{\mu\in\mathcal{C}} \liminf_{n\to\infty} -\frac{1}{n}\log\mu(x_{1..n}) \le \alpha\})$. By assumption, $H(A_\alpha) \le \alpha/\log|\mathcal{X}|$, so that from (2) for all $x_1,x_2,\dots \in A_\alpha$ we obtain

$$\liminf_{n\to\infty} -\frac{1}{n}\log\rho_A(x_{1..n}) \le \alpha. \tag{3}$$

Furthermore, define $\rho := \sum_{q\in Q} w_q \rho_{A_q}$, where $Q = [0,1]\cap\mathbb{Q}$ is the set of rationals in $[0,1]$ and $(w_q)_{q\in Q}$ is any sequence of positive reals satisfying $\sum_{q\in Q} w_q = 1$. For every $\alpha\in[0,1]$ let $q_k\in Q$, $k\in\mathbb{N}$ be such a sequence that $0\le q_k-\alpha\le 1/k$. Then, for every $n\in\mathbb{N}$ and every $x_1,x_2,\dots\in A_{q_k}$ we have

$$-\frac{1}{n}\log\rho(x_{1..n}) \le -\frac{1}{n}\log\rho_q(x_{1..n}) - \frac{\log w_{q_k}}{n}.$$

From this and (3) we get

$$\liminf_{n\to\infty} -\frac{1}{n}\log\rho(x_{1..n}) \le \liminf_{n\to\infty}\rho_{q_k}(x_{1..n}) + 1/k \le q_k + 1/k.$$

Since this holds for every $k\in\mathbb{N}$, it follows that for all $x_1,x_2,\dots\in\cap_{k\in\mathbb{N}}A_{q_k} = A_\alpha$ we have

$$\liminf_{n\to\infty} -\frac{1}{n}\log\rho(x_{1..n}) \le \inf_{k\in\mathbb{N}}(q_k+1/k) = \alpha,$$

which completes the proof of the implication $(ii)\Rightarrow(i)$. ■

6. Discussion

It has been long realized that the so-called probabilistic and agnostic (adversarial, non-stochastic, deterministic) settings of the problem of sequential prediction are strongly related. This has been most evident from looking at the solutions to these problems, which are usually based on the same ideas. Here we have proposed a formulation of the agnostic problem as a non-realizable case of the probabilistic problem. While being very close to the traditional one, this setting allows us to directly compare the two problems. As a somewhat surprising result, we can see that whether the two problems are different depends on the measure of performance chosen: in the case of prediction in total variation distance they coincide, while in the case of prediction in expected average KL divergence they are different. In the latter case, the distinction becomes particularly apparent on the example of stationary processes: while a solution to the realizable problem has long been known, here we have shown that there is no solution to the agnostic version of this problem. This formalization also allowed us to introduce another problem that lies in between the realizable and the fully agnostic problems: given a class of process measures $\mathcal{C}$, find a predictor whose predictions are asymptotically correct for every measure for which at least one of the measures in $\mathcal{C}$ gives asymptotically correct predictions (Problem 2). This problem is less restrictive then the fully agnostic one (in particular, it is not concerned with the behaviour of a predictor on every deterministic sequence) but at the same time the solutions to this problem have performance guarantees far outside the model class considered.

In essence, the formulation of Problem 2 suggests to assume that we have a set of models one of which is good enough to make predictions, with the goal of combining the predictive powers of these models. This is perhaps a good compromise between making modelling assumptions on the data (the data is generated by one of the models we have) and the fully agnostic, worst-case, setting.

Since the problem formulations presented here are mostly new (at least, in such a general form), it is not surprising that there are many questions left open. A promising route to obtain new results seems to be to first analyse the case of prediction in total variation, which amounts to studying the relation of absolute continuity and singularity of probability measures, and then to try and find analogues in less restrictive (and thus more interesting and difficult) cases of predicting only the next observation, possibly with Cesaro averaging. This is the approach that we took in this work. Here it is interesting to find properties common to all or most of the prediction problems (in total variation as well as with respect to other measures of the performance), if it is at all possible. For example, the "countable Bayes" property of Theorem 12, that states that if there is a solution to a given sequence prediction problem for a set $\mathcal{C}$ then a solution can be obtained as a mixture over a suitable countable subset of $\mathcal{C}$, holds for Problems 1–3 in total variation, and for Problems 1 and 2 in KL divergence; however we conjecture that it does not hold for the Problem 3 in KL divergence.

It may also be interesting to study algebraic properties of the relation $\geq_{KL}$ that arises when studying Problem 2. We have show that it shares some properties with the relation $\geq_{tv}$ of absolute continuity. Since the latter characterizes prediction in total variation and the former characterizes prediction in KL divergence (in the sense of Problem 2), which is much weaker, it would be interesting to see exactly what properties the two relations share.

Another direction for future research concerns finite-time performance analysis. In this work we have adopted the asymptotic approach to the prediction problem, ignoring the behaviour of predictors before asymptotic. While for prediction in total variation it is a natural choice, for other measures of performance, including average KL divergence, it is clear that Problems 1-3 admit non-asymptotic formulations. It is also interesting what are the relations between performance guarantees that can be obtained in non-asymptotic formulations of Problems 1–3.

7. Proof of Theorem 12

Proof Define the sets C_μ as the set of all measures $\tau \in \mathcal{P}$ such that μ predicts τ in expected average KL divergence. Let $\mathcal{C}^+ := \cup_{\mu\in\mathcal{C}} C_\mu$. For each $\tau \in \mathcal{C}^+$ let $p(\tau)$ be any (fixed) $\mu \in \mathcal{C}$ such that $\tau \in C_\mu$. In other words, $\mathcal{C}^+$ is the set of all measures that are predicted by some of the measures in $\mathcal{C}$, and for each measure τ in $\mathcal{C}^+$ we designate one "parent" measure $p(\tau)$ from $\mathcal{C}$ such that $p(\tau)$ predicts τ.

Define the weights $w_k := 1/k(k+1)$, for all $k \in \mathbb{N}$.

Step 1. For each $\mu \in \mathcal{C}^+$ let δ_n be any monotonically increasing function such that $\delta_n(\mu) = o(n)$ and $d_n(\mu, p(\mu)) = o(\delta_n(\mu))$. Define the sets

$$U_\mu^n := \left\{ x_{1..n} \in \mathcal{X}^n : \mu(x_{1..n}) \geq \frac{1}{n}\rho(x_{1..n}) \right\}, \tag{4}$$

$$V_\mu^n := \left\{ x_{1..n} \in \mathcal{X}^n : p(\mu)(x_{1..n}) \geq 2^{-\delta_n(\mu)}\mu(x_{1..n}) \right\}, \tag{5}$$

and

$$T_\mu^n := U_\mu^n \cap V_\mu^n. \tag{6}$$

We will upper-bound $\mu(T_\mu^n)$. First, using Markov's inequality, we derive

$$\mu(\mathcal{X}^n \backslash U_\mu^n) = \mu\left(\frac{\rho(x_{1..n})}{\mu(x_{1..n})} > n\right) \leq \frac{1}{n} E_\mu \frac{\rho(x_{1..n})}{\mu(x_{1..n})} = \frac{1}{n}. \tag{7}$$

Next, observe that for every $n \in \mathbb{N}$ and every set $A \subset \mathcal{X}^n$, using Jensen's inequality we can obtain

$$\begin{aligned} -\sum_{x_{1..n}\in A} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} &= -\mu(A) \sum_{x_{1..n}\in A} \frac{1}{\mu(A)} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} \\ &\geq -\mu(A) \log \frac{\rho(A)}{\mu(A)} \geq -\mu(A) \log \rho(A) - \frac{1}{2}. \end{aligned} \tag{8}$$

Moreover,

$$d_n(\mu, p(\mu)) = -\sum_{x_{1..n}\in\mathcal{X}^n\setminus V_\mu^n} \mu(x_{1..n})\log\frac{p(\mu)(x_{1..n})}{\mu(x_{1..n})}$$
$$-\sum_{x_{1..n}\in V_\mu^n} \mu(x_{1..n})\log\frac{p(\mu)(x_{1..n})}{\mu(x_{1..n})} \geq \delta_n(\mu_n)\mu(\mathcal{X}^n\setminus V_\mu^n) - 1/2,$$

where in the inequality we have used (5) for the first summand and (8) for the second. Thus,

$$\mu(\mathcal{X}^n\setminus V_\mu^n) \leq \frac{d_n(\mu,p(\mu))+1/2}{\delta_n(\mu)} = o(1). \tag{9}$$

From (6), (7) and (9) we conclude

$$\mu(\mathcal{X}^n\setminus T_\mu^n) \leq \mu(\mathcal{X}^n\setminus V_\mu^n) + \mu(\mathcal{X}^n\setminus U_\mu^n) = o(1). \tag{10}$$

Step 2n: a countable cover, time n. Fix an $n\in\mathbb{N}$. Define $m_1^n := \max_{\mu\in\mathcal{C}}\rho(T_\mu^n)$ (since $\mathcal{X}^n$ are finite all suprema are reached). Find any μ_1^n such that $\rho_1^n(T_{\mu_1^n}^n) = m_1^n$ and let $T_1^n := T_{\mu_1^n}^n$. For $k>1$, let $m_k^n := \max_{\mu\in\mathcal{C}}\rho(T_\mu^n\setminus T_{k-1}^n)$. If $m_k^n>0$, let μ_k^n be any $\mu\in\mathcal{C}$ such that $\rho(T_{\mu_k^n}^n\setminus T_{k-1}^n) = m_k^n$, and let $T_k^n := T_{k-1}^n\cup T_{\mu_k^n}^n$; otherwise let $T_k^n := T_{k-1}^n$. Observe that (for each n) there is only a finite number of positive m_k^n, since the set $\mathcal{X}^n$ is finite; let K_n be the largest index k such that $m_k^n>0$. Let

$$\nu_n := \sum_{k=1}^{K_n} w_k p(\mu_k^n).$$

As a result of this construction, for every $n\in\mathbb{N}$ every $k\leq K_n$ and every $x_{1..n}\in T_k^n$ using the definitions (6), (4) and (5) we obtain

$$\nu_n(x_{1..n}) \geq w_k\frac{1}{n}2^{-\delta_n(\mu)}\rho(x_{1..n}). \tag{11}$$

Step 2: the resulting predictor. Finally, define

$$\nu := \frac{1}{2}\gamma + \frac{1}{2}\sum_{n\in\mathbb{N}} w_n\nu_n, \tag{12}$$

where γ is the i.i.d. measure with equal probabilities of all $x\in\mathcal{X}$ (that is, $\gamma(x_{1..n}) = |\mathcal{X}|^{-n}$ for every $n\in\mathbb{N}$ and every $x_{1..n}\in\mathcal{X}^n$). We will show that ν predicts every $\mu\in\mathcal{C}^+$, and then in the end of the proof (Step r) we will show how to replace γ by a combination of a countable set of elements of $\mathcal{C}$ (in fact, γ is just a regularizer which ensures that ν-probability of any word is never too close to 0).

Step 3: ν predicts every $\mu\in\mathcal{C}^+$. Fix any $\mu\in\mathcal{C}^+$. Introduce the parameters $\varepsilon_\mu^n\in(0,1)$, $n\in\mathbb{N}$, to be defined later, and let $j_\mu^n := 1/\varepsilon_\mu^n$. Observe that $\rho(T_k^n\setminus T_{k-1}^n)\geq\rho(T_{k+1}^n\setminus T_k^n)$, for any $k>1$ and any $n\in\mathbb{N}$, by definition of these sets. Since the sets $T_k^n\setminus T_{k-1}^n$, $k\in\mathbb{N}$ are disjoint, we obtain $\rho(T_k^n\setminus T_{k-1}^n)\leq 1/k$. Hence, $\rho(T_\mu^n\setminus T_j^n)\leq\varepsilon_\mu^n$ for some $j\leq j_\mu^n$, since otherwise $m_j^n = \max_{\mu\in\mathcal{C}}\rho(T_\mu^n\setminus T_{j_\mu^n}^n) > \varepsilon_\mu^n$ so that $\rho(T_{j_\mu^n+1}^n\setminus T_{j_\mu^n}^n) > \varepsilon_\mu^n = 1/j_\mu^n$, which is a contradiction. Thus,

$$\rho(T_\mu^n\setminus T_{j_\mu^n}^n) \leq \varepsilon_\mu^n. \tag{13}$$

We can upper-bound $\mu(T_\mu^n \backslash T_{j_\mu^n}^n)$ as follows. First, observe that

$$d_n(\mu,\rho) = -\sum_{x_{1..n}\in T_\mu^n \cap T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} - \sum_{x_{1..n}\in T_\mu^n \backslash T_{j_\mu^n}^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} - \sum_{x_{1..n}\in \mathcal{X}^n \backslash T_\mu^n} \mu(x_{1..n}) \log \frac{\rho(x_{1..n})}{\mu(x_{1..n})} = I + II + III. \quad (14)$$

Then, from (6) and (4) we get

$$I \geq -\log n. \quad (15)$$

From (8) and (13) we get

$$II \geq -\mu(T_\mu^n \backslash T_{j_\mu^n}^n) \log \rho(T_\mu^n \backslash T_{j_\mu^n}^n) - 1/2 \geq -\mu(T_\mu^n \backslash T_{j_\mu^n}^n) \log \varepsilon_\mu^n - 1/2. \quad (16)$$

Furthermore,

$$III \geq \sum_{x_{1..n}\in \mathcal{X}^n \backslash T_\mu^n} \mu(x_{1..n}) \log \mu(x_{1..n}) \geq \mu(\mathcal{X}^n \backslash T_\mu^n) \log \frac{\mu(\mathcal{X}^n \backslash T_\mu^n)}{|\mathcal{X}^n \backslash T_\mu^n|} \geq -\frac{1}{2} - \mu(\mathcal{X}^n \backslash T_\mu^n) n \log |\mathcal{X}|, \quad (17)$$

where the first inequality is obvious, in the second inequality we have used the fact that entropy is maximized when all events are equiprobable and in the third one we used $|\mathcal{X}^n \backslash T_\mu^n| \leq |\mathcal{X}|^n$. Combining (14) with the bounds (15), (16) and (17) we obtain

$$d_n(\mu,\rho) \geq -\log n - \mu(T_\mu^n \backslash T_{j_\mu^n}^n) \log \varepsilon_\mu^n - 1 - \mu(\mathcal{X}^n \backslash T_\mu^n) n \log |\mathcal{X}|,$$

so that

$$\mu(T_\mu^n \backslash T_{j_\mu^n}^n) \leq \frac{1}{-\log \varepsilon_\mu^n} \Big(d_n(\mu,\rho) + \log n + 1 + \mu(\mathcal{X}^n \backslash T_\mu^n) n \log |\mathcal{X}| \Big). \quad (18)$$

From the fact that $d_n(\mu,\rho) = o(n)$ and (10) it follows that the term in brackets is $o(n)$, so that we can define the parameters ε_μ^n in such a way that $-\log \varepsilon_\mu^n = o(n)$ while at the same time the bound (18) gives $\mu(T_\mu^n \backslash T_{j_\mu^n}^n) = o(1)$. Fix such a choice of ε_μ^n. Then, using (10), we conclude

$$\mu(\mathcal{X}^n \backslash T_{j_\mu^n}^n) \leq \mu(\mathcal{X}^n \backslash T_\mu^n) + \mu(T_\mu^n \backslash T_{j_\mu^n}^n) = o(1). \quad (19)$$

We proceed with the proof of $d_n(\mu,\nu) = o(n)$. For any $x_{1..n} \in T_{j_\mu^n}^n$ we have

$$\nu(x_{1..n}) \geq \frac{1}{2} w_n \nu_n(x_{1..n}) \geq \frac{1}{2} w_n w_{j_\mu^n} \frac{1}{n} 2^{-\delta_n(\mu)} \rho(x_{1..n}) \geq \frac{w_n}{4n} (\varepsilon_\mu^n)^2 2^{-\delta_n(\mu)} \rho(x_{1..n}), \quad (20)$$

where the first inequality follows from (12), the second from (11), and in the third we have used $w_{j^n_\mu} = 1/(j^n_\mu)(j^n_\mu+1)$ and $j^n_\mu = 1/\varepsilon^\mu_n$. Next we use the decomposition

$$d_n(\mu,\nu) = -\sum_{x_{1..n}\in T^n_{j^n_\mu}} \mu(x_{1..n}) \log\frac{\nu(x_{1..n})}{\mu(x_{1..n})} - \sum_{x_{1..n}\in \mathcal{X}^n\backslash T^n_{j^n_\mu}} \mu(x_{1..n}) \log\frac{\nu(x_{1..n})}{\mu(x_{1..n})} = I + II. \qquad (21)$$

From (20) we find

$$\begin{aligned} I &\le -\log\left(\frac{w_n}{4n}(\varepsilon^n_\mu)^2 2^{-\delta_n(\mu)}\right) - \sum_{x_{1..n}\in T^n_{j^n_\mu}} \mu(x_{1..n}) \log\frac{\rho(x_{1..n})}{\mu(x_{1..n})} \\ &= (o(n) - 2\log\varepsilon^n_\mu + \delta_n(\mu)) + \left(d_n(\mu,\rho) + \sum_{x_{1..n}\in \mathcal{X}^n\backslash T^n_{j^n_\mu}} \mu(x_{1..n}) \log\frac{\rho(x_{1..n})}{\mu(x_{1..n})}\right) \\ &\le o(n) - \sum_{x_{1..n}\in \mathcal{X}^n\backslash T^n_{j^n_\mu}} \mu(x_{1..n}) \log \mu(x_{1..n}) \\ &\le o(n) + \mu(\mathcal{X}^n\backslash T^n_{j^n_\mu}) n \log|\mathcal{X}| = o(n), \qquad (22) \end{aligned}$$

where in the second inequality we have used $-\log\varepsilon^n_\mu = o(n)$, $d_n(\mu,\rho) = o(n)$ and $\delta_n(\mu) = o(n)$, in the last inequality we have again used the fact that the entropy is maximized when all events are equiprobable, while the last equality follows from (19). Moreover, from (12) we find

$$II \le \log 2 - \sum_{x_{1..n}\in \mathcal{X}^n\backslash T^n_{j^n_\mu}} \mu(x_{1..n}) \log\frac{\gamma(x_{1..n})}{\mu(x_{1..n})} \le 1 + n\mu(\mathcal{X}^n\backslash T^n_{j^n_\mu}) \log|\mathcal{X}| = o(n), \qquad (23)$$

where in the last inequality we have used $\gamma(x_{1..n}) = |\mathcal{X}|^{-n}$ and $\mu(x_{1..n}) \le 1$, and the last equality follows from (19).

From (21), (22) and (23) we conclude $\frac{1}{n}d_n(\nu,\mu) \to 0$.

Step r: the regularizer γ. It remains to show that the i.i.d. regularizer γ in the definition of ν (12), can be replaced by a convex combination of a countably many elements from $\mathcal{C}$. Indeed, for each $n\in\mathbb{N}$, denote

$$A_n := \{x_{1..n}\in\mathcal{X}^n : \exists\mu\in\mathcal{C}\; \mu(x_{1..n}) \ne 0\},$$

and let for each $x_{1..n}\in\mathcal{X}^n$ the measure $\mu_{x_{1..n}}$ be any measure from $\mathcal{C}$ such that $\mu_{x_{1..n}}(x_{1..n}) \ge \frac{1}{2}\sup_{\mu\in\mathcal{C}}\mu(x_{1..n})$. Define

$$\gamma'_n(x'_{1..n}) := \frac{1}{|A_n|}\sum_{x_{1..n}\in A_n} \mu_{x_{1..n}}(x'_{1..n}),$$

for each $x'_{1..n}\in A^n$, $n\in\mathbb{N}$, and let $\gamma' := \sum_{k\in\mathbb{N}} w_k\gamma'_k$. For every $\mu\in\mathcal{C}$ we have

$$\gamma'(x_{1..n}) \ge w_n|A_n|^{-1}\mu_{x_{1..n}}(x_{1..n}) \ge \frac{1}{2}w_n|\mathcal{X}|^{-n}\mu(x_{1..n})$$

for every $n\in\mathbb{N}$ and every $x_{1..n}\in A_n$, which clearly suffices to establish the bound $II = o(n)$ as in (23). ■

Acknowledgments

Some of the results have appeared (Ryabko, 2010b) in the proceedings of COLT'10. The author is grateful to the anonymous reviewers for their constructive comments on the paper. This research was partially supported by the French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council and FEDER through CPER 2007-2013, ANR projects EXPLO-RA (ANR-08-COSI-004) and Lampada (ANR-09-EMER-007), by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 231495 (project CompLACS), and by Pascal-2.

References

P. Billingsley. *Ergodic Theory and Information*. Wiley, New York, 1965.

D. Blackwell and L. Dubins. Merging of opinions with increasing information. *Annals of Mathematical Statistics*, 33:882–887, 1962.

N. Cesa-Bianchi and G. Lugosi. On prediction of individual sequences. *Annals of Statistics*, 27: 1865–1895, 1999.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006. ISBN 0521841089.

A. P. Dawid. Prequential data analysis. *Lecture Notes-Monograph Series*, 17:113–126, 1992.

E. Kalai and E. Lehrer. Weak and strong merging of opinions. *Journal of Mathematical Economics*, 23:73–86, 1994.

R. Krichevsky. *Universal Compression and Retrival*. Kluwer Academic Publishers, 1993.

A.I. Plesner and V.A. Rokhlin. Spectral theory of linear operators, II. *Uspekhi Matematicheskikh Nauk*, 1:71–191, 1946.

B. Ryabko. Twice-universal coding. *Problems of Information Transmission*, 3:173–177, 1984.

B. Ryabko. Noiseless coding of combinatorial sources, Hausdorff dimension, and Kolmogorov complexity. *Problems of Information Transmission*, 22:16–26, 1986.

B. Ryabko. Prediction of random sequences and universal coding. *Problems of Information Transmission*, 24:87–96, 1988.

D. Ryabko. Characterizing predictable classes of processes. In A. Ng J. Bilmes, editor, *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09)*, Montreal, Canada, 2009.

D. Ryabko. On finding predictors for arbitrary families of processes. *Journal of Machine Learning Research*, 11:581–602, 2010a.

D. Ryabko. Sequence prediction in realizable and non-realizable cases. In *Proc. the 23rd Conference on Learning Theory (COLT 2010)*, pages 119–131, Haifa, Israel, 2010b.

D. Ryabko and M. Hutter. On sequence prediction for arbitrary measures. In *Proc. 2007 IEEE International Symposium on Information Theory*, pages 2346–2350, Nice, France, 2007. IEEE.

D. Ryabko and M. Hutter. Predicting non-stationary processes. *Applied Mathematics Letters*, 21 (5):477–482, 2008.

A. N. Shiryaev. *Probability*. Springer, 1996.

R. J. Solomonoff. Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Information Theory*, IT-24:422–432, 1978.

J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.

Journal of Machine Learning Research 12 (2011) 2181-2210 Submitted 4/11; Published 7/11

Discriminative Learning of Bayesian Networks via Factorized Conditional Log-Likelihood

Alexandra M. Carvalho ASMC@INESC-ID.PT
Department of Electrical Engineering
Instituto Superior Técnico, Technical University of Lisbon
INESC-ID, R. Alves Redol 9
1000-029 Lisboa, Portugal

Teemu Roos TEEMU.ROOS@CS.HELSINKI.FI
Department of Computer Science
Helsinki Institute for Information Technology
P.O. Box 68
FI-00014 University of Helsinki, Finland

Arlindo L. Oliveira AML@INESC-ID.PT
Department of Computer Science and Engineering
Instituto Superior Técnico, Technical University of Lisbon
INESC-ID, R. Alves Redol 9
1000-029 Lisboa, Portugal

Petri Myllymäki PETRI.MYLLYMAKI@CS.HELSINKI.FI
Department of Computer Science
Helsinki Institute for Information Technology
P.O. Box 68
FI-00014 University of Helsinki, Finland

Editor: Russell Greiner

Abstract

We propose an efficient and parameter-free scoring criterion, the factorized conditional log-likelihood ($\hat{\text{f}}$CLL), for learning Bayesian network classifiers. The proposed score is an approximation of the conditional log-likelihood criterion. The approximation is devised in order to guarantee decomposability over the network structure, as well as efficient estimation of the optimal parameters, achieving the same time and space complexity as the traditional log-likelihood scoring criterion. The resulting criterion has an information-theoretic interpretation based on interaction information, which exhibits its discriminative nature. To evaluate the performance of the proposed criterion, we present an empirical comparison with state-of-the-art classifiers. Results on a large suite of benchmark data sets from the UCI repository show that $\hat{\text{f}}$CLL-trained classifiers achieve at least as good accuracy as the best compared classifiers, using significantly less computational resources.

Keywords: Bayesian networks, discriminative learning, conditional log-likelihood, scoring criterion, classification, approximation

1. Introduction

Bayesian networks have been widely used for classification, see Friedman et al. (1997), Grossman and Domingos (2004), Su and Zhang (2006) and references therein. However, they are often outperformed by much simpler methods (Domingos and Pazzani, 1997; Friedman et al., 1997). One of the likely causes for this appears to be the use of so called *generative learning* methods in choosing the Bayesian network structure as well as its parameters. In contrast to generative learning, where the goal is to be able to describe (or generate) the entire data, *discriminative learning* focuses on the capacity of a model to discriminate between different classes of instances. Unfortunately, discriminative learning of Bayesian network classifiers has turned out to be computationally much more challenging than generative learning. This led Friedman et al. (1997) to bring up the question: are there heuristic approaches that allow efficient discriminative learning of Bayesian network classifiers?

During the past years different discriminative approaches have been proposed, which tend to decompose the problem into two tasks: (i) discriminative structure learning, and (ii) discriminative parameter learning. Greiner and Zhou (2002) were among the first to work along these lines. They introduced a discriminative parameter learning algorithm, called the *Extended Logistic Regression* (ELR) algorithm, that uses gradient descent to maximize the *conditional log-likelihood* (CLL) of the class variable given the other variables. Their algorithm can be applied to an arbitrary Bayesian network structure. However, they only considered *generative* structure learning methods. Greiner and Zhou (2002) demonstrated that their parameter learning method, although computationally more expensive than the usual generative approach that only involves counting relative frequencies, leads to improved parameter estimates. More recently, Su et al. (2008) have managed to significantly reduce the computational cost by proposing an alternative discriminative parameter learning method, called the *Discriminative Frequency Estimate* (DFE) algorithm, that exhibits nearly the same accuracy as the ELR algorithm but is considerably more efficient.

Full structure and parameter learning based on the ELR algorithm is a burdensome task. Employing the procedure of Greiner and Zhou (2002) would require a new gradient descent for each candidate network at each search step, turning the method computationally infeasible. Moreover, even in parameter learning, ELR is not guaranteed to find globally optimal CLL parameters. Roos et al. (2005) have showed that globally optimal solutions can be guaranteed *only* for network structures that satisfy a certain graph-theoretic property, including for example, the naive Bayes and tree-augmented naive Bayes (TAN) structures (see Friedman et al., 1997) as special cases. The work by Greiner and Zhou (2002) supports this result empirically by demonstrating that their ELR algorithm is successful when combined with (generatively learned) TAN classifiers.

For discriminative structure learning, Kontkanen et al. (1998) and Grossman and Domingos (2004) propose to choose network structures by maximizing CLL while choosing parameters by maximizing the parameter posterior or the (joint) *log-likelihood* (LL). The *BNC algorithm* of Grossman and Domingos (2004) is actually very similar to the hill-climbing algorithm of Heckerman et al. (1995), except that it uses CLL as the primary objective function. Grossman and Domingos (2004) also experiment with full structure and parameter optimization for CLL. However, they found that full optimization does not produce better results than those obtained by the much simpler approach where parameters are chosen by maximizing LL.

The contribution of this paper is to present two criteria similar to CLL, but with much better computational properties. The criteria can be used for efficient learning of augmented naive Bayes classifiers. We mostly focus on structure learning. Compared to the work of Grossman and Domingos (2004), our structure learning criteria have the advantage of being *decomposable*, a property that enables the use of simple and very efficient search heuristics. For the sake of simplicity, we assume a binary valued class variable when deriving our results. However, the methods are directly applicable to multi-class classification, as demonstrated in the experimental part (Section 5).

Our first criterion is the *approximated conditional log-likelihood* (aCLL). The proposed score is the minimum variance unbiased (MVU) approximation to CLL under a class of uniform priors on certain parameters of the joint distribution of the class variable and the other attributes. We show that for most parameter values, the approximation error is very small. However, the aCLL criterion still has two unfavorable properties. First, the parameters that maximize aCLL are hard to obtain, which poses problems at the parameter learning phase, similar to those posed by using CLL directly. Second, the criterion is not well-behaved in the sense that it sometimes diverges when the parameters approach the usual relative frequency estimates (maximizing LL).

In order to solve these two shortcomings, we devise a second approximation, the *factorized conditional log-likelihood* ($\hat{\text{f}}$CLL). The $\hat{\text{f}}$CLL approximation is uniformly bounded, and moreover, it is maximized by the easily obtainable relative frequency parameter estimates. The $\hat{\text{f}}$CLL criterion allows a neat interpretation as a sum of LL and another term involving the *interaction information* between a node, its parents, and the class variable; see Pearl (1988), Cover and Thomas (2006), Bilmes (2000) and Pernkopf and Bilmes (2005).

To gauge the performance of the proposed criteria in classification tasks, we compare them with several popular classifiers, namely, *tree augmented naive Bayes* (TAN), *greedy hill-climbing* (GHC), C4.5, *k-nearest neighbor* (*k*-NN), *support vector machine* (SVM) and *logistic regression* (LogR). On a large suite of benchmark data sets from the UCI repository, $\hat{\text{f}}$CLL-trained classifiers outperform, with a statistically significant margin, their generatively-trained counterparts, as well as C4.5, *k*-NN and LogR classifiers. Moreover, $\hat{\text{f}}$CLL-optimal classifiers are comparable with ELR induced ones, as well as SVMs (with linear, polynomial, and radial basis function kernels). The advantage of $\hat{\text{f}}$CLL with respect to these latter classifiers is that it is computationally as efficient as the LL scoring criterion, and considerably more efficient than ELR and SVMs.

The paper is organized as follows. In Section 2 we review some basic concepts of Bayesian networks and introduce our notation. In Section 3 we discuss generative and discriminative learning of Bayesian network classifiers. In Section 4 we present our scoring criteria, followed by experimental results in Section 5. Finally, we draw some conclusions and discuss future work in Section 6. The proofs of the results stated throughout this paper are given in the Appendix.

2. Bayesian Networks

In this section we introduce some notation, while recalling relevant concepts and results concerning discrete Bayesian networks.

Let X be a *discrete random variable* taking values in a countable set $\mathcal{X} \subset \mathbb{R}$. In all what follows, the domain $\mathcal{X}$ is finite. We denote an n-dimensional *random vector* by $\mathbf{X} = (X_1, \ldots, X_n)$ where each component X_i is a random variable over $\mathcal{X}_i$. For each variable X_i, we denote the elements of $\mathcal{X}_i$ by $x_{i1}, \ldots, x_{ir_i}$ where r_i is the number of values X_i can take. The probability that $\mathbf{X}$ takes value $\mathbf{x}$ is denoted by $P(\mathbf{x})$, conditional probabilities $P(\mathbf{x} \mid \mathbf{z})$ being defined correspondingly.

A *Bayesian network* (BN) is defined by a pair $B = (G, \Theta)$, where G refers to the graph structure, and Θ are the parameters. The structure $G = (V, E)$ is a *directed acyclic graph* (DAG) with vertices (nodes) V, each corresponding to one of the random variables X_i, and edges E representing direct dependencies between the variables. The (possibly empty) set of nodes from which there is an edge to node X_i is called the set of the *parents* of X_i, and denoted by Π_{X_i}. For each node X_i, we denote the number of possible *parent configurations* (vectors of the parents' values) by q_i, the actual parent configurations being ordered (arbitrarily) and denoted by $w_{i1}, \ldots, w_{iq_i}$. The *parameters*, $\Theta = \{\theta_{ijk}\}_{i\in\{1,\ldots,n\},\, j\in\{1,\ldots,q_i\},k\in\{1,\ldots,r_i\}}$, determine the *local distributions* in the network via

$$P_B(X_i = x_{ik} \mid \Pi_{X_i} = w_{ij}) = \theta_{ijk}.$$

The local distributions define a unique joint probability distribution over $\mathbf{X}$ given by

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid \Pi_{X_i}).$$

The conditional independence properties pertaining to the joint distribution are essentially determined by the network structure. Specifically, X_i is conditionally independent of its non-descendants given its parents Π_{X_i} in G (Pearl, 1988).

Learning unrestricted Bayesian networks from data under typical scoring criteria is NP-hard (Chickering et al., 2004). This result has led the Bayesian network community to search for the largest subclass of network structures for which there is an efficient learning algorithm. First attempts confined the network to tree structures and used Edmonds (1967) and Chow and Liu (1968) optimal branching algorithms. More general classes of Bayesian networks have eluded efforts to develop efficient learning algorithms. Indeed, Chickering (1996) showed that learning the structure of a Bayesian network is NP-hard even for networks constrained to have in-degree at most two. Later, Dasgupta (1999) showed that even learning an optimal *polytree* (a DAG in which there are not two different paths from one node to another) with maximum in-degree two is NP-hard. Moreover, Meek (2001) showed that identifying the best *path structure*, that is, a total order over the nodes, is hard. Due to these hardness results exact polynomial-time algorithms for learning Bayesian networks have been restricted to tree structures.

Consequently, the standard methodology for addressing the problem of learning Bayesian networks has become heuristic score-based learning where a *scoring criterion* ϕ is considered in order to quantify the capability of a Bayesian network to explain the observed data. Given data $D = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ and a scoring criterion ϕ, the task is to find a Bayesian network B that maximizes the score $\phi(B,D)$. Many search algorithms have been proposed, varying both in terms of the formulation of the search space (network structures, equivalence classes of network structures and orderings over the network variables), and in the algorithm to search the space (greedy hill-climbing, simulated annealing, genetic algorithms, tabu search, etc). The most common scoring criteria are reviewed in Carvalho (2009) and Yang and Chang (2002). We refer the interested reader to newly developed scoring criteria to the works of de Campos (2006) and Silander et al. (2010).

Score-based learning algorithms can be extremely efficient if the employed scoring criterion is decomposable. A scoring criterion ϕ is said to be *decomposable* if the score can be expressed as a sum of local scores that depends only on each node and its parents, that is, in the form

$$\phi(B,D) = \sum_{i=1}^{n} \phi_i(\Pi_{X_i}, D).$$

One of the most common criteria is the *log-likelihood* (LL), see Heckerman et al. (1995):

$$\mathrm{LL}(B \mid D) \quad = \quad \sum_{t=1}^{N} \log P_B(y_t^1,\ldots,y_t^n) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk} \log \theta_{ijk},$$

which is clearly decomposable.

The *maximum likelihood* (ML) parameters that maximize LL are easily obtained as the *observed frequency estimates* (OFE) given by

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}, \tag{1}$$

where N_{ijk} denotes the number of instances in D where $X_i = x_{ik}$ and $\Pi_{X_i} = w_{ij}$, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Plugging these estimates back into the LL criterion yields

$$\widehat{\mathrm{LL}}(G \mid D) \quad = \quad \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk} \log\left(\frac{N_{ijk}}{N_{ij}}\right).$$

The notation with G as the argument instead of $B = (G, \Theta)$ emphasizes that once the use of the OFE parameters is decided upon, the criterion is a function of the network structure, G, only.

The $\widehat{\mathrm{LL}}$ scoring criterion tends to favor complex network structures with many edges since adding an edge never decreases the likelihood. This phenomenon leads to *overfitting* which is usually avoided by adding a complexity penalty to the log-likelihood or by restricting the network structure.

3. Bayesian Network Classifiers

A *Bayesian network classifier* is a Bayesian network over $\mathbf{X} = (X_1,\ldots,X_n,C)$, where C is a class variable, and the goal is to classify instances $(X_1,\ldots,X_n)$ to different classes. The variables $X_1,\ldots,X_n$ are called *attributes*, or *features*. For the sake of computational efficiency, it is common to restrict the network structure. We focus on *augmented naive Bayes classifiers*, that is, Bayesian network classifiers where the class variable has no parents, $\Pi_C = \emptyset$, and all attributes have at least the class variable as a parent, $C \in \Pi_{X_i}$ for all X_i.

For convenience, we introduce a few additional notations that apply to augmented naive Bayes models. Let the class variable C range over s distinct values, and denote them by $z_1,\ldots,z_s$. Recall that the parents of X_i are denoted by Π_{X_i}. The parents of X_i without the class variable are denoted by $\Pi^*_{X_i} = \Pi_{X_i} \setminus \{C\}$. We denote the number of possible configurations of the parent set $\Pi^*_{X_i}$ by q^*_i; hence, $q^*_i = \prod_{X_j \in \Pi^*_{X_i}} r_j$. The j'th configuration of $\Pi^*_{X_i}$ is represented by w^*_{ij}, with $1 \le j \le q^*_i$. Similarly to the general case, local distributions are determined by the corresponding parameters

$$P(C = z_c) = \theta_c,$$
$$P(X_i = x_{ik} \mid \Pi^*_{X_i} = w^*_{ij}, C = z_c) = \theta_{ijck}.$$

We denote by N_{ijck} the number of instances in the data D where $X_i = x_{ik}$, $\Pi^*_{X_i} = w^*_{ij}$ and $C = z_c$. Moreover, the following short-hand notations will become useful:

$$N_{ij*k} = \sum_{c=1}^{s} N_{ijck}, \qquad N_{ij*} = \sum_{k=1}^{r_i}\sum_{c=1}^{s} N_{ijck},$$

$$N_{ijc} = \sum_{k=1}^{r_i} N_{ijck}, \qquad N_c = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q^*_i}\sum_{k=1}^{r_i} N_{ijck}.$$

Finally, we recall that the total number of instances in the data D is N.

The ML estimates (1) become now

$$\hat{\theta}_c = \frac{N_c}{N}, \qquad \text{and} \qquad \hat{\theta}_{ijck} = \frac{N_{ijck}}{N_{ijc}}, \tag{2}$$

which can again be plugged into the LL criterion:

$$\begin{aligned} \widehat{\mathrm{LL}}(G \mid D) &= \sum_{t=1}^{N} \log P_B(y_t^1, \ldots, y_t^n, c_t) \\ &= \sum_{c=1}^{s} \left(N_c \log\left(\frac{N_c}{N}\right) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijck} \log\left(\frac{N_{ijck}}{N_{ijc}}\right) \right). \end{aligned} \tag{3}$$

As mentioned in the introduction, if the goal is to discriminate between instances belonging to different classes, it is more natural to consider the *conditional log-likelihood* (CLL), that is, the probability of the class variable given the attributes, as a score:

$$\mathrm{CLL}(B \mid D) = \sum_{t=1}^{N} \log P_B(c_t \mid y_t^1, \ldots, y_t^n).$$

Friedman et al. (1997) noticed that the log-likelihood can be rewritten as

$$\mathrm{LL}(B \mid D) = \mathrm{CLL}(B \mid D) + \sum_{t=1}^{N} \log P_B(y_t^1, \ldots, y_t^n). \tag{4}$$

Interestingly, the objective of generative learning is precisely to maximize the whole sum, whereas the goal of discriminative learning consists on maximizing only the first term in (4). Friedman et al. (1997) attributed the underperformance of learning methods based on LL to the term $\mathrm{CLL}(B \mid D)$ being potentially much smaller than the second term in Equation (4). Unfortunately, CLL does not decompose over the network structure, which seriously hinders structure learning, see Bilmes (2000); Grossman and Domingos (2004). Furthermore, there is no closed-form formula for optimal parameter estimates maximizing CLL, and computationally more expensive techniques such as ELR are required (Greiner and Zhou, 2002; Su et al., 2008).

4. Factorized Conditional Log-Likelihood Scoring Criterion

The above shortcomings of earlier discriminative approaches to learning Bayesian network classifiers, and the CLL criterion in particular, make it natural to explore good approximations to the CLL that are more amenable to efficient optimization. More specifically, we now set out to construct approximations that are *decomposable*, as discussed in Section 2.

4.1 Developing a New Scoring Criterion

For simplicity, assume that the class variable is binary, $C = \{0, 1\}$. For the binary case the conditional probability of the class variable can then be written as

$$P_B(c_t \mid y_t^1, \ldots, y_t^n) = \frac{P_B(y_t^1, \ldots, y_t^n, c_t)}{P_B(y_t^1, \ldots, y_t^n, c_t) + P_B(y_t^1, \ldots, y_t^n, 1 - c_t)}. \tag{5}$$

For convenience, we denote the two terms in the denominator as

$$\begin{aligned} U_t &= P_B(y_t^1,\ldots,y_t^n,c_t) \qquad \text{and} \\ V_t &= P_B(y_t^1,\ldots,y_t^n,1-c_t), \end{aligned} \tag{6}$$

so that Equation (5) becomes simply

$$P_B(c_t \mid y_t^1,\ldots,y_t^n) = \frac{U_t}{U_t+V_t}.$$

We stress that both U_t and V_t depend on B, but for the sake of readability we omit B in the notation. Observe that while $(y_t^1,\ldots,y_t^n,c_t)$ is the t'th sample in the data set D, the vector $(y_t^1,\ldots,y_t^n,1-c_t)$, which we call the *dual sample* of $(y_t^1,\ldots,y_t^n,c_t)$, may or may not occur in D.

The log-likelihood (LL), and the conditional log-likelihood (CLL) now take the form

$$\begin{aligned} \mathrm{LL}(B \mid D) &= \sum_{t=1}^{N} \log U_t, \qquad \text{and} \\ \mathrm{CLL}(B \mid D) &= \sum_{t=1}^{N} \log U_t - \log(U_t+V_t). \end{aligned}$$

Recall that our goal is to derive a decomposable scoring criterion. Unfortunately, even though $\log U_t$ decomposes, $\log(U_t+V_t)$ does not.

Now, let us consider approximating the log-ratio

$$f(U_t,V_t) = \log\left(\frac{U_t}{U_t+V_t}\right),$$

by functions of the form

$$\hat{f}(U_t,V_t) = \alpha \log U_t + \beta \log V_t + \gamma,$$

where α, β, and γ are real numbers to be chosen so as to minimize the approximation error. Since the accuracy of the approximation obviously depends on the values of U_t and V_t as well as the constants α, β, and γ, we need to make some assumptions about U_t and V_t in order to determine suitable values of α, β and γ. We explicate one possible set of assumptions, which will be seen to lead to a good approximation for a very wide range of U_t and V_t. We emphasize that the role of the assumptions is to aid in arriving at good choices of the constants α, β, and γ, after which we can dispense with the assumptions—they need not, in particular, hold true exactly.

Start by noticing that $R_t = 1-(U_t+V_t)$ is the probability of observing neither the t'th sample nor its dual, and hence, the triplet (U_t,V_t,R_t) are the parameters of a trinomial distribution. We assume, for the time being, that no knowledge about the values of the parameters (U_t,V_t,R_t) is available. Therefore, it is natural to assume that (U_t,V_t,R_t) follows the uniform Dirichlet distribution, $\mathrm{Dirichlet}(1,1,1)$, which implies that

$$(U_t,V_t) \sim \mathrm{Uniform}(\Delta^2), \tag{7}$$

where $\Delta^2 = \{(x,y) : x+y \leq 1 \text{ and } x,y \geq 0\}$ is the 2-simplex set. However, with a brief reflection on the matter, we can see that such an assumption is actually rather unrealistic. Firstly, by inspecting the total number of possible observed samples, we expect, R_t to be relatively large (close to 1).

In fact, U_t and V_t are expected to become exponentially small as the number of attributes grows. Therefore, it is reasonable to assume that

$$U_t, V_t \leq p < \frac{1}{2} < R_t$$

for some $0 < p < \frac{1}{2}$. Combining this constraint with the uniformity assumption, Equation (7), yields the following assumption, which we will use as a basis for our further analysis.

Assumption 1 There exists a small positive $p < \frac{1}{2}$ such that

$$(U_t, V_t) \sim \text{Uniform}(\Delta^2)|_{U_t, V_t \leq p} = \text{Uniform}([0,p] \times [0,p]).$$

Assumption 1 implies that U_t and V_t are uniform i.i.d. random variables over $[0, p]$ for some (possibly unknown) positive real number $p < \frac{1}{2}$. (See Appendix B for an alternative justification for Assumption 1.) As we show below, we do not need to know the actual value of p. Consequently, the envisaged approximation will be robust to the choice of p.

We obtain the best fitting approximation $\hat{f}$ by the least squares method.

Theorem 1 Under Assumption 1, the values of α, β and γ that minimize the *mean square error* (MSE) of $\hat{f}$ w.r.t. f are given by

$$\alpha = \frac{\pi^2 + 6}{24}, \tag{8}$$

$$\beta = \frac{\pi^2 - 18}{24}, \text{ and} \tag{9}$$

$$\gamma = \frac{\pi^2}{12 \ln 2} - \left(2 + \frac{(\pi^2 - 6)\log p}{12}\right), \tag{10}$$

where log is the binary logarithm and ln is the natural logarithm.

We show that the mean difference between $\hat{f}$ and f is zero for all values of p, that is, $\hat{f}$ is *unbiased*.[1] Moreover, we show that $\hat{f}$ is the approximation with the lowest variance among unbiased ones.

Theorem 2 The approximation $\hat{f}$ with α, β, γ defined as in Theorem 1 is the *minimum variance unbiased* (MVU) approximation of f.

Next, we derive the standard error of the approximation $\hat{f}$ in the square $[0, p] \times [0, p]$, which, curiously, does not depend on p. To this end, consider

$$\mu = E[f(U_t, V_t)] = \frac{1}{2\ln(2)} - 2$$

which is a negative value; as it should since f ranges over $(-\infty, 0]$.

1. Herein we apply the nomenclature of estimation theory in the context of approximation. Thus, an approximation is *unbiased* if $E[\hat{f}(U_t, V_t) - f(U_t, V_t)] = 0$ for all p. Moreover, an approximation is *(uniformly) minimum variance unbiased* if the value $E[(\hat{f}(U_t, V_t) - f(U_t, V_t))^2]$ is the lowest for all unbiased approximations and values of p.

Theorem 3 The approximation $\hat{f}$ with α, β, and γ defined as in Theorem 1 has *standard error* given by

$$\sigma = \sqrt{\frac{36 + 36\pi^2 - \pi^4}{288 \ln^2(2)} - 2} \approx 0.352$$

and *relative standard error* $\sigma/|\mu| \approx 0.275$.

Figure 1 illustrates the function f as well as its approximation $\hat{f}$ for $(U_t, V_t) \in [0, p] \times [0, p]$ with $p = 0.05$. The approximation error, $f - \hat{f}$ is shown in Figure 2. While the properties established in Theorems 1–3 are useful, we find it even more important that, as seen in Figure 2, the error is close to zero except when either U_t or V_t approaches zero. Moreover, we point out that the choice of $p = 0.05$ used in the figure is not important: having chosen another value would have produced identical graphs except in the scale of the U_t and V_t. In particular, the scale and numerical values on the vertical axis (i.e., in Figure 2, the error) would have been precisely the same.

Using the approximation in Theorem 1 to approximate CLL yields

$$\begin{aligned}
\mathrm{CLL}(B \mid D) &\approx \sum_{t=1}^{N} \alpha \log U_t + \beta \log V_t + \gamma \\
&= \sum_{t=1}^{N} (\alpha + \beta) \log U_t - \beta \log\left(\frac{U_t}{V_t}\right) + \gamma \\
&= (\alpha + \beta)\mathrm{LL}(B \mid D) - \beta \sum_{t=1}^{N} \log\left(\frac{U_t}{V_t}\right) + N\gamma, \qquad (11)
\end{aligned}$$

where constants α, β and γ are given by Equations (8), (9) and (10), respectively. Since we want to maximize $\mathrm{CLL}(B \mid D)$, we can drop the constant $N\gamma$ in the approximation, as maxima are invariant under monotone transformations, and so we can just maximize the following formula, which we call the *approximate conditional log-likelihood* (aCLL):

$$\begin{aligned}
\mathrm{aCLL}(B \mid D) &= (\alpha + \beta)\mathrm{LL}(B \mid D) - \beta \sum_{t=1}^{N} \log\left(\frac{U_t}{V_t}\right) \\
&= (\alpha + \beta)\mathrm{LL}(B \mid D) - \beta \sum_{i=1}^{n} \sum_{j=1}^{q_i^*} \sum_{k=1}^{r_i} \sum_{c=0}^{1} N_{ijck} \log\left(\frac{\theta_{ijck}}{\theta_{ij(1-c)k}}\right) \\
&\qquad - \beta \sum_{c=0}^{1} N_c \log\left(\frac{\theta_c}{\theta_{(1-c)}}\right). \qquad (12)
\end{aligned}$$

The fact that $N\gamma$ can be removed from the maximization in (11) is most fortunate, as we eliminate the dependency on p. An immediate consequence of this fact is that we do not need to know the actual value of p in order to employ the criterion.

We are now in the position of having constructed a *decomposable* approximation of the conditional log-likelihood score that was shown to be very accurate for a wide range of parameters U_t and V_t. Due to the dependency of these parameters on Θ, that is, the parameters of the Bayesian network B (recall Equation (6)), the score still requires that a suitable set of parameters is chosen. Finding the parameters maximizing the approximation is, however, difficult; apparently as difficult as finding parameters maximizing the CLL directly. Therefore, whatever computational advantage

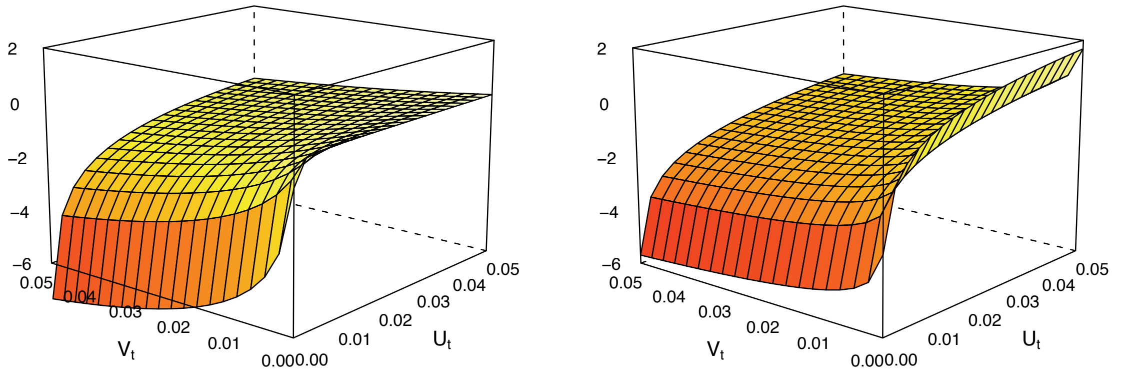


Figure 1: Comparison between $f(U_t,V_t) = \log\left(\frac{U_t}{U_t+V_t}\right)$ (left), and $\hat{f}(U_t,V_t) = \alpha \log U_t + \beta \log V_t + \gamma$ (right). Both functions diverge (to $-\infty$) as $U_t \to 0$. The latter diverges (to $+\infty$) also when $V_t \to 0$. For the interpretation of different colors, see Figure 2 below.

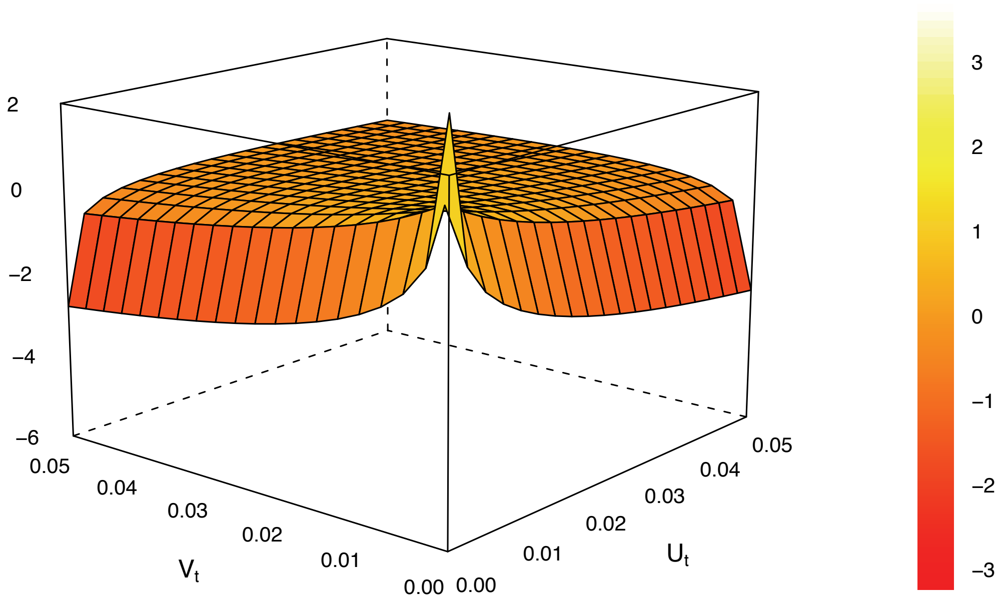


Figure 2: Approximation error: the difference between the exact value and the approximation given in Theorem 1. Notice that the error is symmetric in the two arguments, and diverges as $U_t \to 0$ or $V_t \to 0$. For points where neither argument is close to zero, the error is small (close to zero).

is gained by decomposability, it would seem to be dwarfed by the expensive parameter optimization phase.

Furthermore, trying to use the OFE parameters in aCLL may lead to problems since the approximation is undefined at points where either U_t or V_t is zero. To better see why this is the case, substitute the OFE parameters, Equation (2), into the aCLL criterion, Equation (12), to obtain

$$\hat{\mathrm{a}}\mathrm{CLL}(G \mid D) = (\alpha+\beta)\,\widehat{\mathrm{LL}}(G \mid D) - \beta \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\log\left(\frac{N_{ijck}N_{ij(1-c)}}{N_{ijc}N_{ij(1-c)k}}\right)$$
$$-\beta\sum_{c=0}^{1} N_c \log\left(\frac{N_c}{N_{1-c}}\right). \qquad (13)$$

The problems are associated with the denominator in the second term. In LL and CLL criteria, similar expressions where the denominator may be zero are always eliminated by the OFE parameters since they are always multiplied by zero, see, for example, Equation (3), where $N_{ijc} = 0$ implies $N_{ijck} = 0$. However, there is no guarantee that $N_{ij(1-c)k}$ is non-zero even if the factors in the numerator are non-zero, and hence the division by zero may lead to actual indeterminacies.

Next, we set out to resolve these issues by presenting a well-behaved approximation that enables easy optimization of both structure (via decomposability), as well as parameters.

4.2 Achieving a Well-Behaved Approximation

In this section, we address the singularities of aCLL under OFE by constructing an approximation that is well-behaved.

Recall aCLL in Equation (12). Given a fixed network structure, the parameters that maximize the first term, $(\alpha+\beta)\mathrm{LL}(B \mid D)$, are given by OFE. However, as observed above, the second term may actually be unbounded due to the appearance of $\theta_{ij(1-c)k}$ in the denominator. In order to obtain a well-behaved score, we must therefore make a further modification to the second term. Our strategy is to ensure that the resulting score is *uniformly bounded* and *maximized by OFE parameters*. The intuition behind this is that we can thus guarantee not only that the score is well-behaved, but also that parameter learning is achieved in a simple and efficient way by using the OFE parameters—solving both of the aforementioned issues with the aCLL score. As it turns out, we can satisfy our goal while still retaining the discriminative nature of the score.

The following result is of importance in what follows.

Theorem 4 Consider a Bayesian network B whose structure is given by a fixed directed acyclic graph, G. Let $f(B \mid D)$ be a score defined by

$$f(B \mid D) = \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\left(\lambda\log\left(\frac{\theta_{ijck}}{\frac{N_{ijc}}{N_{ij*}}\theta_{ijck}+\frac{N_{ij(1-c)}}{N_{ij*}}\theta_{ij(1-c)k}}\right)\right), \qquad (14)$$

where λ is an arbitrary positive real value. Then, the parameters Θ that maximize $f(B \mid D)$ are given by the observed frequency estimates (OFE) obtained from G.

The theorem implies that by replacing the second term in (12) by (a non-negative multiple of) $f(B \mid D)$ in Equation (14), we get a criterion where both the first and the second term are maximized by the OFE parameters. We will now proceed to determine a suitable value for the parameter λ appearing in Equation (14).

To clarify the analysis, we introduce the following short-hand notations:

$$\begin{aligned} A_1 &= N_{ijc}\theta_{ijck}, & A_2 &= N_{ijc}, \\ B_1 &= N_{ij(1-c)}\theta_{ij(1-c)k}, & B_2 &= N_{ij(1-c)}. \end{aligned} \tag{15}$$

With simple algebra, we can rewrite the logarithm in the second term of Equation (12) using the above notations as

$$\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ij(1-c)}\theta_{ij(1-c)k}}\right) - \log\left(\frac{N_{ijc}}{N_{ij(1-c)}}\right) = \log\left(\frac{A_1}{B_1}\right) - \log\left(\frac{A_2}{B_2}\right). \tag{16}$$

Similarly, the logarithm in (14) becomes

$$\begin{aligned} &\lambda\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right) + \rho - \lambda\log\left(\frac{N_{ijc}}{N_{ijc}+N_{ij(1-c)}}\right) - \rho \\ &= \lambda\log\left(\frac{A_1}{A_1+B_1}\right) + \rho - \lambda\log\left(\frac{A_2}{A_2+B_2}\right) - \rho, \end{aligned} \tag{17}$$

where we used $N_{ij*} = N_{ijc} + N_{ij(1-c)}$; we have introduced the constant ρ that was added and subtracted without changing the value of the expression for a reason that will become clear shortly. By comparing Equations (16) and (17), it can be seen that the latter is obtained from the former by replacing expressions of the form $\log(\frac{A}{B})$ by expressions of the form $\lambda\log(\frac{A}{A+B})+\rho$.

We can simplify the two-variable approximation to a single variable one by taking $W = \frac{A}{A+B}$. In this case we have that $\frac{A}{B} = \frac{W}{1-W}$, and so we propose to apply once again the least squares method to approximate the function

$$g(W) = \log\left(\frac{W}{1-W}\right)$$

by

$$\hat{g}(W) = \lambda\log W + \rho.$$

The role of the constant ρ is simply to translate the approximate function to better match the target $g(W)$.

As in the previous approximation, here too it is necessary to make assumptions about the values of A and B (and thus W), in order to find suitable values for the parameters λ and ρ. Again, we stress that the sole purpose of the assumption is to guide in the choice of the parameters.

As both A_1, A_2, B_1, and B_2 in Equation (15) are all non-negative, the ratio $W_i = \frac{A_i}{A_i+B_i}$ is always between zero and one, for both $i \in \{1,2\}$, and hence it is natural to make the straightforward assumption that W_1 and W_2 are uniformly distributed along the unit interval. This gives us the following assumption.

Assumption 2 We assume that

$$\begin{aligned} \frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}} &\sim \text{Uniform}(0,1), \quad \text{and} \\ \frac{N_{ijc}}{N_{ijc}+N_{ij(1-c)}} &\sim \text{Uniform}(0,1). \end{aligned}$$

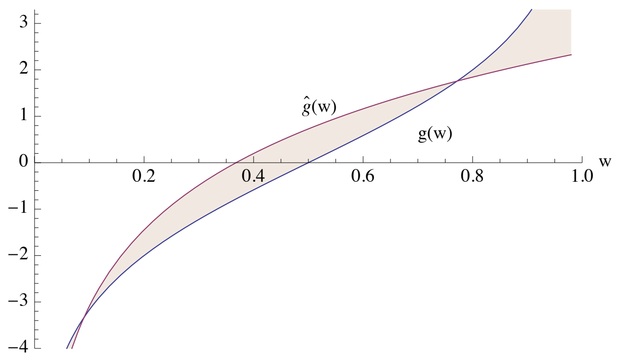


Figure 3: Plot of $g(w) = \log\left(\frac{w}{1-w}\right)$ and $\hat{g}(w) = \lambda \log w + \rho$.

Herein, it is worthwhile noticing that although the previous assumption was meant to hold for general parameters, in practice, we know in this case that OFE will be used. Hence, Assumption 2 reduces to

$$\frac{N_{ijck}}{N_{ij*k}} \sim \text{Uniform}(0,1), \text{ and } \frac{N_{ijc}}{N_{ij*}} \sim \text{Uniform}(0,1).$$

Under this assumption, the mean squared error of the approximation can be minimized analytically, yielding the following solution.

Theorem 5 Under Assumption 2, the values of λ and ρ that minimize the mean square error (MSE) of $\hat{g}$ w.r.t. g are given by

$$\lambda = \frac{\pi^2}{6}, \text{ and} \tag{18}$$

$$\rho = \frac{\pi^2}{6 \ln 2}. \tag{19}$$

Theorem 6 The approximation $\hat{g}$ with λ and ρ defined as in Theorem 5 is the minimum variance unbiased (MVU) approximation of g.

In order to get an idea of the accuracy of the approximation $\hat{g}$, consider the graph of $\log\left(\frac{w}{1-w}\right)$ and $\lambda \log w + \rho$ in Figure 3. It may appear problematic that the approximation gets worse as w tends to one. However this is actually unavoidable since that is precisely where âCLL diverges, and our goal is to obtain a criterion that is uniformly bounded.

To wrap up, we first rewrite the logarithm of the second term in Equation (12) using formula (16), and then apply the above approximation to both terms to get

$$\log\left(\frac{\theta_{ijck}}{\theta_{ij(1-c)k}}\right) \approx \lambda \log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck} + N_{ij(1-c)}\theta_{ij(1-c)k}}\right) + \rho - \lambda \log\left(\frac{N_{ijc}}{N_{ij*}}\right) - \rho, \tag{20}$$

where ρ cancels out. A similar analysis can be applied to rewrite the logarithm of the third term in Equation (12) leading to

$$\log\left(\frac{\theta_c}{\theta_{(1-c)}}\right) = \log\left(\frac{\theta_c}{1-\theta_c}\right) \approx \lambda \log \theta_c + \rho. \tag{21}$$

Plugging the approximations of Equations (20) and (21) into Equation (12) gives us finally the *factorized conditional log-likelihood* (fCLL) score:

$$\begin{aligned}
\mathrm{fCLL}(B \mid D) &= (\alpha+\beta)\,\mathrm{LL}(B \mid D) \\
&- \beta\lambda \sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\left(\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}-N_{ij(1-c)}\theta_{ij(1-c)k}}\right)-\log\left(\frac{N_{ijc}}{N_{ij*}}\right)\right) \\
&- \beta\lambda\sum_{c=0}^{1} N_c \log\theta_c - \beta N\rho.
\end{aligned} \tag{22}$$

Observe that the third term of Equation (22) is such that

$$-\beta\lambda\sum_{c=0}^{1} N_c\log\theta_c = -\beta\lambda N\sum_{c=0}^{1}\frac{N_c}{N}\log\theta_c, \tag{23}$$

and, since $\beta < 0$, by Gibbs inequality (see Lemma 8 in the Appendix at page 2204) the parameters that maximize Equation (23) are given by the OFE, that is, $\hat{\theta}_c = \frac{N_c}{N}$. Therefore, by Theorem 4, given a fixed structure, the maximizing parameters of fCLL are easily obtained as OFE. Moreover, the fCLL score is clearly decomposable.

As a final step, we plug in the OFE parameters, Equation (2), into the fCLL criterion, Equation (22), to obtain

$$\begin{aligned}
\widehat{\mathrm{f}}\mathrm{CLL}(G \mid D) &= (\alpha+\beta)\,\widehat{\mathrm{LL}}(B \mid D) - \beta\lambda\sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1} N_{ijck}\left(\log\left(\frac{N_{ijck}}{N_{ij*k}}\right)-\log\left(\frac{N_{ijc}}{N_{ij*}}\right)\right) \\
&\quad -\beta\lambda\sum_{c=0}^{1} N_c\log\left(\frac{N_c}{N}\right) - \beta N\rho,
\end{aligned} \tag{24}$$

where we also use the OFE parameters in the log-likelihood $\widehat{\mathrm{LL}}$. Observe that we can drop the last two terms in Equation (24) as they become constants for a given data set.

4.3 Information-Theoretical Interpretation

Before we present empirical results illustrating the behavior of the proposed scoring criteria, we point out that the $\widehat{\mathrm{f}}\mathrm{CLL}$ criterion has an interesting information-theoretic interpretation based on *interaction information*. We will first rewrite LL in terms of conditional mutual information, and then, similarly, rewrite the second term of $\widehat{\mathrm{f}}\mathrm{CLL}$ in Equation (24) in terms of interaction information.

As Friedman et al. (1997) point out, the local contribution of the i'th variable to $\mathrm{LL}(B \mid D)$ (recall Equation (3)) is given by

$$\begin{aligned}
N\sum_{j=1}^{q_i^*}\sum_{c=0}^{1}\sum_{k=1}^{r_i}\frac{N_{ijck}}{N}\log\left(\frac{N_{ijck}}{N_{ijc}}\right) &= -NH_{\hat{P}_D}(X_i \mid \Pi^*_{X_i}, C) \\
&= -NH_{\hat{P}_D}(X_i \mid C) + NI_{\hat{P}_D}(X_i\,;\Pi^*_{X_i} \mid C),
\end{aligned} \tag{25}$$

where $H_{\hat{P}_D}(X_i \mid \ldots)$ denotes the *conditional entropy*, and $I_{\hat{P}_D}(X_i\,;\Pi^*_{X_i} \mid C)$ denotes the *conditional mutual information*, see Cover and Thomas (2006). The subscript $\hat{P}_D$ indicates that the information theoretic quantities are evaluated under the joint distribution $\hat{P}_D$ of $(\vec{X}, C)$ induced by the OFE parameters.

Since the first term on the right-hand side of (25) does not depend on $\Pi^*_{X_i}$, finding the parents of X_i that maximize $\mathrm{LL}(B \mid D)$ is equivalent to choosing the parents that maximize the second term, $NI_{\hat{P}_D}(X_i\,;\Pi^*_{X_i} \mid C)$, which measures the information that $\Pi^*_{X_i}$ provides about X_i when the value of C is known.

Let us now turn to the second term of the $\hat{\mathrm{f}}$CLL score in Equation (24). The contribution of the i'th variable in it can also be expressed in information theoretic terms as follows:

$$\begin{aligned} -\beta\lambda N \left(H_{\hat{P}_D}(C \mid X_i, \Pi^*_{X_i}) - H_{\hat{P}_D}(C \mid \Pi^*_{X_i})\right) &= \beta\lambda N I_{\hat{P}_D}(C\,; X_i \mid \Pi^*_{X_i}) \\ &= \beta\lambda N \left(I_{\hat{P}_D}(C\,; X_i\,; \Pi^*_{X_i}) + I_{\hat{P}_D}(C\,; X_i))\right), \end{aligned} \tag{26}$$

where $I_{\hat{P}_D}(C\,; X_I\,; \Pi^*_{X_i})$ denotes the *interaction information* (McGill, 1954), or the *"co-information"* (Bell, 2003); for a review on the history and use of interaction information in machine learning and statistics, see Jakulin (2005).

Since $I_{\hat{P}_D}(X_i\,; C)$ on the last line of Equation (26) does not depend on $\Pi^*_{X_i}$, finding the parents of X_i that maximize the sum amounts to maximizing the interaction information. This is intuitive, since the interaction information measures the increase—or the decrease, as it can also be negative—of the mutual information between X_i and C when the parent set $\Pi^*_{X_i}$ is included in the model.

All said, the $\hat{\mathrm{f}}$CLL criterion can be written as

$$\hat{\mathrm{f}}\mathrm{CLL}(G \mid D) = \sum_{i=1}^{n} \left[(\alpha + \beta) N I_{\hat{P}_D}(X_i\,; \Pi^*_{X_i} \mid C) - \beta\lambda N I_{\hat{P}_D}(C\,; X_i\,; \Pi^*_{X_i})\right] + const, \tag{27}$$

where *const* is a constant independent of the network structure and can thus be omitted. To get a concrete idea of the trade-off between the first two terms, the numerical values of the constants can be evaluated to obtain

$$\hat{\mathrm{f}}\mathrm{CLL}(G \mid D) \approx \sum_{i=1}^{n} \left[0.322 N I_{\hat{P}_D}(X_i\,; \Pi^*_{X_i} \mid C) + 0.557 N I_{\hat{P}_D}(C\,; X_i\,; \Pi^*_{X_i})\right] + const. \tag{28}$$

Normalizing the weights shows that the first term that corresponds to the behavior of the LL criterion, Equation (25), has proportional weight of approximately 36.7 percent, while the second term that gives $\hat{\mathrm{f}}$CLL criterion its discriminative nature has the weight 63.3 percent.[2]

In addition to the insight provided by the information-theoretic interpretation of $\hat{\mathrm{f}}$CLL, it also provides a practically most useful corollary: the $\hat{\mathrm{f}}$CLL criterion is score equivalent. A scoring criterion is said to be *score equivalent* if it assigns the same score to all network structures encoding the same independence assumptions, see Verma and Pearl (1990), Chickering (2002), Yang and Chang (2002) and de Campos (2006).

Theorem 7 The $\hat{\mathrm{f}}$CLL criterion is score equivalent for augmented naive Bayes classifiers.

The practical utility of the above result is due to the fact that it enables the use of powerful algorithms, such as the tree-learning method by Chow and Liu (1968), in learning TAN classifiers.

2. The particular linear combination of the two terms in Equation (28) brings out the question what would happen in only one of the terms was retained, or equivalently, if one of the weights was set to zero. As mentioned above, the first term corresponds to the LL criterion, and hence, setting the weight of the second term to zero would reduce the criterion to LL. We also experimented with a criterion where only the second term is retained but this was observed to yield poor results; for details, see the additional material at `http://kdbio.inesc-id.pt/~asmc/software/fCLL.html`.

4.4 Beyond Binary Classification and TAN

Although âCLL and $\hat{\text{f}}$CLL scoring criteria were devised having in mind binary classification tasks, their application in multi-class problems is straightforward.[3] For the case of $\hat{\text{f}}$CLL, the expression (24) does not involve the dual samples. Hence, it can be trivially adapted for non-binary classification tasks. On the other hand, the score âCLL in Equation (13) does depend on the dual samples. To adapt it for multi-class problems, we considered $N_{ij(1-c)k} = N_{ij*k} - N_{ijck}$ and $N_{ij(1-c)} = N_{ij} - N_{ijc}$.

Finally, we point out that despite being derived under the augmented naive Bayes model, the $\hat{\text{f}}$CLL score can be readily applied to models where the class variable is *not* a parent of some of the attributes. Hence, we can use it as a criterion for learning more general structures. The empirical results below demonstrate that this indeed leads to good classifiers.

5. Experimental Results

We implemented the $\hat{\text{f}}$CLL scoring criterion on top of the Weka open-source software (Hall et al., 2009). Unfortunately, the Weka implementation of the TAN classifier assumes that the learning criterion is score equivalent, which rules out the use of our âCLL criterion. Non-score-equivalent criteria require the Edmonds' maximum branchings algorithm that builds a maximal *directed* spanning tree (see Edmonds 1967, or Lawler 1976) instead of an undirected one obtained by the Chow-Liu algorithm (Chow and Liu, 1968). Edmonds' algorithm had already been implemented by some of the authors (see Carvalho et al., 2007) using Mathematica 7.0 and the Combinatorica package (Pemmaraju and Skiena, 2003). Hence, the âCLL criterion was implemented in this environment. All source code and the data sets used in the experiments, can be found at fCLL web page.[4]

We evaluated the performance of âCLL and $\hat{\text{f}}$CLL scoring criteria in classification tasks comparing them with state-of-the-art classifiers. We performed our evaluation on the same 25 benchmark data sets used by Friedman et al. (1997). These include 23 data sets from the UCI repository of Newman et al. (1998) and two artificial data sets, *corral* and *mofn*, designed by Kohavi and John (1997) to evaluate methods for feature subset selection. A description of the data sets is presented in Table 1. All continuous-valued attributes were discretized using the supervised entropy-based method by Fayyad and Irani (1993). For this task we used the Weka package.[5] Instances with missing values were removed from the data sets.

The classifiers used in the experiments were:
GHC2: Greedy hill climber classifier with up to 2 parents.
TAN: Tree augmented naive Bayes classifier.
C4.5: C4.5 classifier.
k-NN: *k*-nearest neighbor classifier, with $k = 1,3,5$.
SVM: Support vector machine with linear kernel.
SVM2: Support vector machine with polynomial kernel of degree 2.

3. As suggested by an anonymous referee, the techniques used in Section 4.1 for deriving the âCLL criterion can be generalized to the multi-class case as well as to other distributions in addition to the uniform one in a straightforward manner by applying results from regression theory. We plan to explore such generalizations of both the âCLL and $\hat{\text{f}}$CLL criteria in future work.
4. fCLL web page is at `http://kdbio.inesc-id.pt/~asmc/software/fCLL.html`.
5. Discretization was done using `weka.filters.supervised.attribute.Discretize`, with default parameters. This discretization improved the accuracy of all classifiers used in the experiments, including those that do not necessarily require discretization, that is, C4.5 *k*-NN, SVM, and LogR.

	Data Set	Features	Classes	Train	Test
1	australian	15	2	690	CV-5
2	breast	10	2	683	CV-5
3	chess	37	2	2130	1066
4	cleve	14	2	296	CV-5
5	corral	7	2	128	CV-5
6	crx	16	2	653	CV-5
7	diabetes	9	2	768	CV-5
8	flare	11	2	1066	CV-5
9	german	21	2	1000	CV-5
10	glass	10	7	214	CV-5
11	glass2	10	2	163	CV-5
12	heart	14	2	270	CV-5
13	hepatitis	20	2	80	CV-5
14	iris	5	3	150	CV-5
15	letter	17	26	15000	5000
16	lymphography	19	4	148	CV-5
17	mofn-3-7-10	11	2	300	1024
18	pima	9	2	768	CV-5
19	satimage	37	6	4435	2000
20	segment	20	7	1540	770
21	shuttle-small	10	7	3866	1934
22	soybean-large	36	19	562	CV-5
23	vehicle	19	4	846	CV-5
24	vote	17	2	435	CV-5
25	waveform-21	22	3	300	4700

Table 1: Description of data sets used in the experiments.

SVMG: Support vector machine with Gaussian (RBF) kernel.
LogR: Logistic regression.

Bayesian network-based classifiers (GHC2 and TAN) were included in different flavors, differing in the scoring criterion used for structure learning (LL, $\hat{a}$CLL, $\hat{f}$CLL) and the parameter estimator (OFE, ELR). Each variant along with the implementation used in the experiments is described in Table 2. Default parameters were used in all cases unless explicitly stated. Excluding TAN classifiers obtained with the ELR method, we improved the performance of Bayesian network classifiers by smoothing parameter estimates according to a Dirichlet prior (see Heckerman et al., 1995). The smoothing parameter was set to 0.5, the default in Weka. The same strategy was used for TAN classifiers implemented in Mathematica. For discriminative parameter learning with ELR, parameters were initialized to the OFE values. The gradient descent parameter optimization was terminated using *cross tuning* as suggested in Greiner et al. (2005).

Three different kernels were applied in SVM classifiers: (i) a linear kernel of the form $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$; (ii) a polynomial kernel of the form $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$; and (iii) a Gaussian (radial basis

Classifier	**Struct.**	**Param.**	**Implementation**
GHC2	LL	OFE	`HillClimber` (P=2) implementation from Weka
GHC2	$\hat{f}$CLL	OFE	`HillClimber` (P=2) implementation from Weka
TAN	LL	OFE	`TAN` implementation from Weka
TAN	LL	ELR	`TAN` implementation from Greiner and Zhou (2002)
TAN	$\hat{a}$CLL	OFE	`TAN` implementation from Carvalho et al. (2007)
TAN	$\hat{f}$CLL	OFE	`TAN` implementation from Weka
C4.5			`J48` implementation from Weka
1-NN			`IBk` (K=1) implementation from Weka
3-NN			`IBk` (K=3) implementation from Weka
5-NN			`IBk` (K=5) implementation from Weka
SVM			`SMO` implementation from Weka
SVM2			`SMO` with `PolyKernel` (E=2) implementation from Weka
SVMG			`SMO` with `RBFKernel` implementation from Weka
LogR			`Logistic` implementation from Weka

Table 2: Classifiers used in the experiments.

function) kernel of the form $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$. Following established practice (see Hsu et al., 2003), we used a grid-search on the penalty parameter C and the RBF kernel parameter γ, using cross-validation. For linear and polynomial kernels we selected C from $[10^{-1}, 1, 10, 10^2]$ by using 5-fold cross-validation on the training set. For the RBF kernel we selected C and γ from $[10^{-1}, 1, 10, 10^2]$ and $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$, respectively, by using 5-fold cross-validation on the training set.

The accuracy of each classifier is defined as the percentage of successful predictions on the test sets in each data set. As suggested by Friedman et al. (1997), accuracy was measured via the holdout method for larger training sets, and via stratified five-fold cross-validation for smaller ones, using the methods described by Kohavi (1995). Throughout the experiments, we used the same cross-validation folds for every classifier. Scatter plots of the accuracies of the proposed methods against the others are depicted in Figure 4 and Figure 5. Points above the diagonal line represent cases where the method shown in the vertical axis performs better than the one on the horizontal axis. Crosses over the points depict the standard deviation. The standard deviation is computed according to the binomial formula $\sqrt{acc \times (1 - acc)/m}$, where acc is the classifier accuracy and, for the cross-validation tests, m is the size of the data set. For the case of holdout tests, m is the size of the test set. Tables with the accuracies and standard deviations can be found at the fCLL webpage.

We compare the performance of the classifiers using Wilcoxon signed-rank tests, using the same procedure as Grossman and Domingos (2004). This test is applicable when paired classification accuracy differences, along the data sets, are independent and non-normally distributed. Alternatively, a paired t-test could be used, but as the Wilcoxon signed-rank test is more conservative than the paired t-test, we apply the former. Results are depicted in Table 3 and Table 4. Each entry of Table 3 and Table 4 gives the Z-score and p-value of the significance test for the corresponding pairs

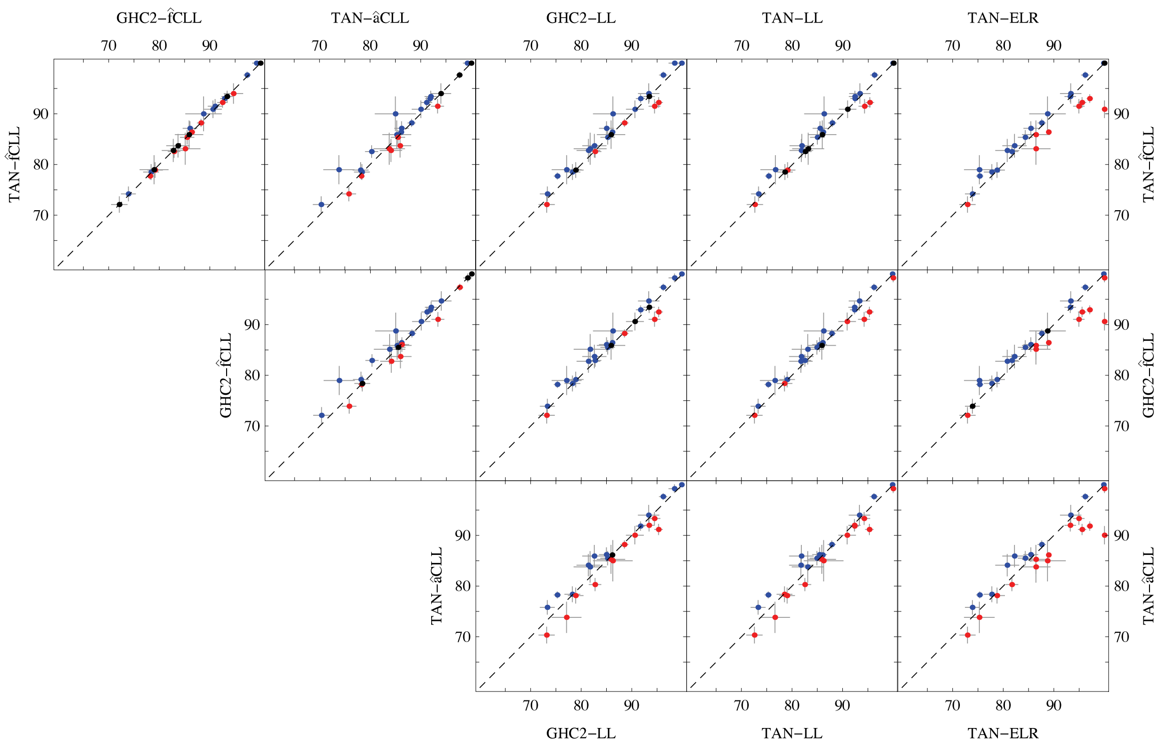


Figure 4: Scatter plots of the accuracy of Bayesian network-based classifiers.

of classifiers. The arrow points towards the learning algorithm that yields superior classification performance. A double arrow is used if the difference is significant with p-value smaller than 0.05.

Over all, TAN-f̂CLL-OFE and GHC-f̂CLL-OFE performed the best (Tables 3–4). They outperformed C4.5, the nearest neighbor classifiers, and logistic regression, as well as the generatively-trained Bayesian network classifiers, TAN-LL-OFE and GHC-LL-OFE, all differences being statistically significant at the $p < 0.05$ level. On the other hand, TAN-âCLL-OFE did not stand out compared to most of the other methods. Moreover, TAN-f̂CLL-OFE and GHC-f̂CLL-OFE classifiers fared sightly better than TAN-LL-ELR and the SVM classifiers, although the difference was not statistically significant. In these cases, the only practically relevant factor is computational efficiency.

To roughly characterize the computational complexity of learning the various classifiers, we measured the total time required by each classifier to process all the 25 data sets.[6] Most of the methods only took a few seconds ($\sim 1-3$ seconds), except for TAN-âCLL-OFE which took a few minutes ($\sim 2-3$ minutes), SVM with linear kernel which took some minutes ($\sim 17-18$ minutes), TAN-LL-ELR and SVM with polynomial kernel which took a few hours ($\sim 1-2$ hours) and, finally, logistic regression and SVM with RBF kernel which took several hours ($\sim 18-32$ hours). In the case of TAN-âCLL-OFE, the slightly increased computation time was likely caused by the Mathematica package, which is not intended for numerical computation. In theory, the computational complexity of TAN-âCLL-OFE is of the same order as TAN-LL-OFE or TAN-f̂CLL-

6. Reporting the total time instead of the individual times for each data set will emphasize the significance of the larger data sets. However, the individual times were in accordance with the general conclusion drawn from the total time.

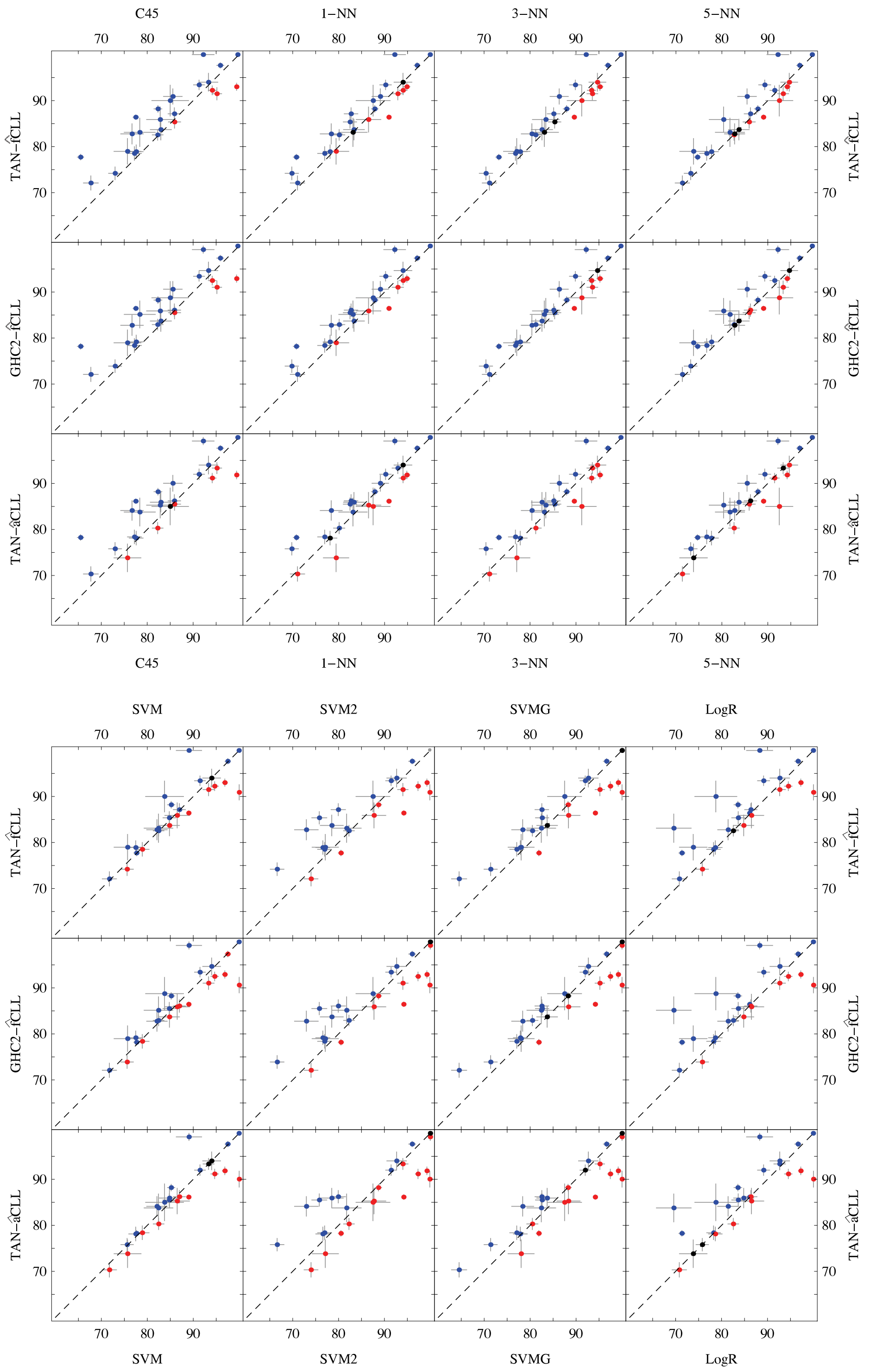


Figure 5: The accuracy of the proposed methods vs. state-of-the-art classifiers.

Classifier	GHC2	TAN	GHC2	TAN	TAN
Struct.	f̂CLL	âCLL	LL	LL	LL
Param.	OFE	OFE	OFE	OFE	ELR
TAN	0.37	1.44	2.13	2.13	0.31
f̂CLL	0.36	0.07	0.02	0.02	0.38
OFE	←	←	⇐	⇐	←
GHC2		1.49	2.26	2.21	0.06
f̂CLL		0.07	0.01	0.01	0.48
OFE		←	⇐	⇐	←
TAN			0.04	-0.34	-1.31
âCLL			0.48	0.37	0.10
OFE			←	↑	↑

Table 3: Comparison of the Bayesian network classifiers against each other, using the Wilcoxon signed-rank test. Each cell of the array gives the Z-score (top) and the corresponding p-value (middle). Arrow points towards the better method, double arrow indicates statistical significance at level $p < 0.05$.

Classifier	C4.5	1-NN	3-NN	5-NN	SVM	SVM2	SVMG	LogR
TAN	3.00	2.25	2.16	2.07	0.43	0.61	0.21	1.80
f̂CLL	<0.01	0.01	0.02	0.02	0.33	0.27	0.42	0.04
OFE	⇐	⇐	⇐	⇐	←	←	←	⇐
GHC2	3.00	2.35	2.20	2.19	0.39	0.74	0.11	1.65
f̂CLL	<0.01	<0.01	0.01	0.01	0.35	0.23	0.45	0.05
OFE	⇐	⇐	⇐	⇐	←	←	←	⇐
TAN	2.26	1.34	1.17	1.31	-0.40	-0.29	-0.55	1.37
âCLL	0.01	0.09	0.12	0.09	0.35	0.38	0.29	0.09
OFE	⇐	←	←	←	↑	↑	↑	←

Table 4: Comparison of the Bayesian network classifiers against other classifiers. Conventions identical to those in Table 3.

OFE: $O(n^2 \log n)$ in the number of features and linear in the number of instances, see Friedman et al. (1997).

Concerning TAN-LL-ELR, the difference is caused by the discriminative parameter learning method (ELR), which is computationally expensive. In our experiments, TAN-LL-ELR was 3 order of magnitude slower than TAN-f̂CLL-OFE. Su and Zhang (2006) report a difference of 6 orders of magnitude, but different data sets were used in their experiments. Likewise, the high computational cost of SVMs was expected. Selection of the regularization parameter using cross-tuning further increases the cost. In our experiments, SVMs were clearly slower than $\hat{f}$CLL-based classifiers. Furthermore, in terms of memory, SVMs with polynomial and RBF kernels, as well as logistic regression, required that the available memory was increased to 1 GB of memory, whereas all other classifiers coped with the default 128 MB.

6. Conclusions and Future Work

We proposed a new decomposable scoring criterion for classification tasks. The new score, called factorized conditional log-likelihood, $\hat{f}$CLL, is based on an approximation of conditional log-likelihood. The new criterion is decomposable, score-equivalent, and allows efficient estimation of both structure and parameters. The computational complexity of the proposed method is of the same order as the traditional log-likelihood criterion. Moreover, the criterion is specifically designed for discriminative learning.

The merits of the new scoring criterion were evaluated and compared to those of common state-of-the-art classifiers, on a large suite of benchmark data sets from the UCI repository. Optimal $\hat{f}$CLL-scored tree-augmented naive Bayes (TAN) classifiers, as well as somewhat more general structures (referred to above as GHC2), performed better than generatively-trained Bayesian network classifiers, as well as C4.5, nearest neighbor, and logistic regression classifiers, with statistical significance. Moreover, $\hat{f}$CLL-optimized classifiers performed better, although the difference is not statistically significant, than those where the Bayesian network parameters were optimized using an earlier discriminative criterion (ELR), as well as support vector machines (with linear, polynomial and RBF kernels). In comparison to the latter methods, our method is considerably more efficient in terms of computational cost, taking 2 to 3 orders of magnitude less time for the data sets in our experiments.

Directions for future work include: studying in detail the asymptotic behavior of $\hat{f}$CLL for TAN and more general models; combining our intermediate approximation, aCLL, with discriminative parameter estimation (ELR); extending aCLL and $\hat{f}$CLL to mixture models; and applications in data clustering.

Acknowledgments

The authors are grateful to the invaluable comments by the anonymous referees. The authors thank Vtor Rocha Vieira, from the Physics Department at IST/TULisbon, for his enthusiasm in cross-checking the analytical integration of the first approximation, and Mrio Figueiredo, from the Electrical Engineering at IST/TULisbon, for his availability in helping with concerns that appeared with respect to this work.

The work of AMC and ALO was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds. The work of AMC was also supported by FCT and EU FEDER via project PneumoSyS (PTDC/SAU-MII/100964/2008). The work of TR and PM was supported in part by the Academy of Finland (Projects MODEST and PRIME) and the European Commission Network of Excellence PASCAL.

Availability: Supplementary material including program code and the data sets used in the experiments can be found at `http://kdbio.inesc-id.pt/~asmc/software/fCLL.html`.

Appendix A. Detailed Proofs

Proof (Theorem 1) We have that

$$\begin{aligned}
S_p(\alpha,\beta,\gamma) &= \int_0^p\int_0^p \frac{1}{p^2}\left(\log\left(\frac{x}{x+y}\right)-(\alpha\log(x)+\beta\log(y)+\gamma)\right)^2 \mathrm{d}y\,\mathrm{d}x\\
&= \frac{1}{12\ln(2)^2}(-\pi^2(-1+\alpha+\beta)\\
&\quad +6(2+4\alpha^2+4\beta^2-4\ln(2)-2\gamma\ln(2)+4\ln(2)^2+8\gamma\ln(2)^2+2\gamma^2\ln^2(2)\\
&\quad +\beta(5-4(2+\gamma)\ln(2))+\alpha(1+4\beta-4(2+\gamma)\ln(2)))\\
&\quad -12(\alpha+\beta)(1+2\alpha+2\beta-4\ln(2)-2\gamma\ln(2))\ln(p)+12(\alpha+\beta)^2\ln^2(p)).
\end{aligned}$$

Moreover, $\nabla.S_p=0$ iff

$$\begin{aligned}
\alpha &= \frac{\pi^2+6}{24},\\
\beta &= \frac{\pi^2-18}{24},\\
\gamma &= \frac{\pi^2}{12\ln(2)}-\left(2+\frac{(\pi^2-6)\log(p)}{12}\right),
\end{aligned}$$

which coincides exactly with (8), (9) and (10), respectively. Now to show that (8), (9) and (10) define a global minimum, take $\delta=(\alpha\log(p)+\beta\log(p)+\gamma)$ and notice that

$$\begin{aligned}
S_p(\alpha,\beta,\gamma) &= \int_0^p\int_0^p \frac{1}{p^2}\left(\log\left(\frac{x}{x+y}\right)-(\alpha\log(x)+\beta\log(y)+\gamma)\right)^2 \mathrm{d}y\,\mathrm{d}x\\
&= \int_0^1\int_0^1 \frac{1}{p^2}\left(\log\left(\frac{px}{px+py}\right)-(\alpha\log(px)+\beta\log(py)+\gamma)\right)^2 p^2\mathrm{d}y\,\mathrm{d}x\\
&= \int_0^1\int_0^1 \left(\log\left(\frac{x}{x+y}\right)-(\alpha\log(x)+\beta\log(y)+(\alpha\log(p)+\beta\log(p)+\gamma))\right)^2 \mathrm{d}y\,\mathrm{d}x\\
&= \int_0^1\int_0^1 \left(\log\left(\frac{x}{x+y}\right)-(\alpha\log(x)+\beta\log(y)+\delta)\right)^2 \mathrm{d}y\,\mathrm{d}x\\
&= S_1(\alpha,\beta,\delta).
\end{aligned}$$

So, S_p has a minimum at (8), (9) and (10) iff S_1 has a minimum at (8), (9) and

$$\delta=\frac{\pi^2}{12\ln(2)}-2.$$

The Hessian of S_1 is

$$\begin{pmatrix}
\frac{4}{\ln^2(2)} & \frac{2}{\ln^2(2)} & -\frac{2}{\ln(2)}\\
\frac{2}{\ln^2(2)} & \frac{4}{\ln^2(2)} & -\frac{2}{\ln(2)}\\
-\frac{2}{\ln(2)} & -\frac{2}{\ln(2)} & 2
\end{pmatrix}$$

and its eigenvalues are

$$\begin{aligned} rcle_1 &= \frac{3+\ln^2(2)+\sqrt{9+2\ln^2(2)+\ln(2)^4}}{\ln^2(2)}, \\ e_2 &= \frac{2}{\ln^2(2)}, \\ e_3 &= \frac{3+\ln^2(2)-\sqrt{9+2\ln^2(2)+\ln(2)^4}}{\ln^2(2)}, \end{aligned}$$

which are all positive. Thus, S_1 has a local minimum in (α,β,δ) and, consequently, S_p has a local minimum in (α,β,γ). Since $\nabla.S_p$ has only one zero, (α,β,γ) is a global minimum of S_p. □

Proof (Theorem 2) We have that

$$\int_0^p\int_0^p \frac{1}{p^2}\left(\log\left(\frac{x}{x+y}\right)-(\alpha\log(x)+\beta\log(y)+\gamma)\right)\mathrm{d}y\,\mathrm{d}x=0$$

for α,β and γ defined as in (8), (9) and (10). Since the MSE coincides with the variance for any unbiased estimator, the proposed approximation is the one with minimum variance. □

Proof (Theorem 3) We have that

$$\sqrt{\int_0^p\int_0^p \frac{1}{p^2}\left(\log\left(\frac{x}{x+y}\right)-(\alpha\log(x)+\beta\log(y)+\gamma)\right)^2\mathrm{d}y\,\mathrm{d}x}=\sqrt{\frac{36+36\pi^2-\pi^4}{288\ln^2(2)}-2}$$

for α,β and γ defined as in (8), (9) and (10), which concludes the proof. □

For the proof of Theorem 4, we recall Gibb's inequality.

Lemma 8 (Gibb's inequality) Let $P(x)$ and $Q(x)$ be two probability distributions over the same domain, then

$$\sum_x P(x)\log(Q(x))\le\sum_x P(x)\log(P(x)).$$

Proof (Theorem 4) We now take advantage of Gibb's inequality to show that the parameters that maximize the $f(B\mid D)$ are those given by the OFE. Observe that

$$\begin{aligned} f(B\mid D) &= \lambda\sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}\sum_{c=0}^{1}N_{ijck}\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right)-\log\left(\frac{N_{ijc}}{N_{ij*}}\right) \\ &= K+\lambda\sum_{i=1}^{n}\sum_{j=1}^{q_i^*}\sum_{k=1}^{r_i}N_{ij*k}\sum_{c=0}^{1}\frac{N_{ijck}}{N_{ij*k}}\log\left(\frac{N_{ijc}\theta_{ijck}}{N_{ijc}\theta_{ijck}+N_{ij(1-c)}\theta_{ij(1-c)k}}\right), \qquad (29) \end{aligned}$$

where K is a constant that does not depend on the parameters θ_{ijck}, and therefore, can be ignored. Moreover, if we take the OFE for the parameters, we have

$$\hat{\theta}_{ijck} = \frac{N_{ijkc}}{N_{ijc}} \text{ and } \hat{\theta}_{ij(1-c)k} = \frac{N_{ijk(1-c)}}{N_{ij(1-c)}}.$$

By plugging the OFE estimates in (29) we obtain

$$\begin{aligned}\hat{f}(G \mid D) &= K + \lambda \sum_{i=1}^{n} \sum_{j=1}^{q_i^*} \sum_{k=1}^{r_i} N_{ij*c} \sum_{c=0}^{1} \frac{N_{ijck}}{N_{ij*k}} \log \left(\frac{N_{ijc} \frac{N_{ijck}}{N_{ijc}}}{N_{ijc} \frac{N_{ijck}}{N_{ijc}} + N_{ij(1-c)} \frac{N_{ij(1-c)k}}{N_{ij(1-c)}}} \right) \\ &= K + \lambda \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ij*k} \sum_{c=0}^{1} \frac{N_{ijck}}{N_{ij*k}} \log \left(\frac{N_{ijck}}{N_{ij*k}} \right).\end{aligned}$$

According to Gibb's inequality, this is the maximum value that $f(B \mid D)$ can attain, and therefore, the parameters that maximize $f(B \mid D)$ are those given by the OFE. □

Proof (Theorem 5) We have that

$$S(\lambda, \rho) = \int_0^1 \left(\log \left(\frac{x}{1-x} \right) - (\lambda \log(x) + \rho) \right)^2 \mathrm{d}x = \frac{6\lambda^2 + \pi^2 + 3\rho^2 \ln^2(2) - \lambda\left(\pi^2 + 6\rho \ln(2)\right)}{3 \ln^2(2)}.$$

Moreover $\nabla . S = 0$ iff

$$\begin{aligned}\lambda &= \tfrac{\pi^2}{6}, \\ \rho &= \tfrac{\pi^2}{6\ln(2)},\end{aligned}$$

which coincides with (18) and (19), respectively. The Hessian of S is

$$\begin{pmatrix} \frac{4}{\ln^2(2)} & -\frac{2}{\ln(2)}, \\ -\frac{2}{\ln(2)} & 2 \end{pmatrix}$$

with eigenvalues

$$\frac{2 + \ln^2(2) \pm \sqrt{4 + \ln^4(2)}}{\ln^2(2)}$$

which are both positive. Hence, there is only one minimum, and (λ, ρ) is the global minimum. □

Proof (Theorem 6) We have that

$$\int_0^1 \left(\log \left(\frac{x}{1-x} \right) - (\lambda \log(x) + \rho) \right) \mathrm{d}x = 0$$

for λ and ρ defined as in Equations (18) and (19). Since the MSE coincides with the variance for any unbiased estimator, the proposed approximation is the one with minimum variance. □

Proof (Theorem 7) By Theorem 2 in Chickering (1995), it is enough to show that for graphs G_1 and G_2 differing only on reversing one covered edge, we have that $\hat{\text{f}}\text{CLL}(G_1 \mid D) = \hat{\text{f}}\text{CLL}(G_2 \mid D)$.

Assume that $X \to Y$ occurs in G_1 and $Y \to X$ occurs in G_2 and that $X \to Y$ is covered, that is, $\Pi_Y^{G_1} = \Pi_X^{G_1} \cup \{X\}$. Since we are only dealing with augment naive Bayes classifiers, X and Y are different from C and so we also have $\Pi_Y^{*G_1} = \Pi_X^{*G_1} \cup \{X\}$. Moreover, take G_0 to be the graph G_1 without the edge $X \to Y$ (which is the same as graph G_2 without the edge $Y \to X$). Then, we have that $\Pi_X^{*G_0} = \Pi_Y^{*G_0} = \Pi^{*G_0}$ and, moreover, the following equalities hold:

$$\begin{array}{ll} \Pi_X^{*G_1} = \Pi^{*G_0}; & \Pi_Y^{*G_2} = \Pi^{*G_0}; \\ \Pi_Y^{*G_1} = \Pi^{*G_0} \cup \{X\}; & \Pi_X^{*G_2} = \Pi^{*G_0} \cup \{Y\}. \end{array}$$

Since $\hat{\mathrm{f}}\mathrm{CLL}$ is a local scoring criterion, $\hat{\mathrm{f}}\mathrm{CLL}(G_1 \mid D)$ can be computed from $\hat{\mathrm{f}}\mathrm{CLL}(G_0 \mid D)$ taking only into account the difference in the contribution of node Y. In this case, by Equation (27), it follows that

$$\begin{aligned} \hat{\mathrm{f}}\mathrm{CLL}(G_1 \mid D) &= \hat{\mathrm{f}}\mathrm{CLL}(G_0 \mid D) - ((\alpha+\beta)NI_{\hat{P}_D}(Y;\Pi^{*G_0} \mid C) - \beta\lambda NI_{\hat{P}_D}(Y;\Pi^{*G_0};C)) \\ &\qquad +((\alpha+\beta)NI_{\hat{P}_D}(Y;\Pi_Y^{*G_1} \mid C) - \beta\lambda NI_{\hat{P}_D}(Y;\Pi_Y^{*G_1};C)) \\ &= \hat{\mathrm{f}}\mathrm{CLL}(G_0 \mid D) + (\alpha+\beta)N(I_{\hat{P}_D}(Y;\Pi^{*G_0} \cup \{X\} \mid C) - I_{\hat{P}_D}(Y;\Pi^{*G_0} \mid C)) \\ &\qquad -\beta\lambda N(I_{\hat{P}_D}(Y;\Pi^{*G_0} \cup \{X\};C) - I_{\hat{P}_D}(Y;\Pi^{*G_0};C)) \end{aligned}$$

and, similarly, that

$$\begin{aligned} \hat{\mathrm{f}}\mathrm{CLL}(G_2 \mid D) &= \hat{\mathrm{f}}\mathrm{CLL}(G_0 \mid D) + (\alpha+\beta)N(I_{\hat{P}_D}(X;\Pi^{*G_0} \cup \{Y\} \mid C) - I_{\hat{P}_D}(X;\Pi^{*G_0} \mid C)) + \\ &\qquad -\beta\lambda N(I_{\hat{P}_D}(X;\Pi^{*G_0} \cup \{Y\};C) - I_{\hat{P}_D}(X;\Pi^{*G_0};C)). \end{aligned}$$

To show that $\hat{\mathrm{f}}\mathrm{CLL}(G_1 \mid D) = \hat{\mathrm{f}}\mathrm{CLL}(G_2 \mid D)$ it suffices to prove that

$$I_{\hat{P}_D}(Y;\Pi^{*G_0} \cup \{X\} \mid C) - I_{\hat{P}_D}(Y;\Pi^{*G_0} \mid C) = I_{\hat{P}_D}(X;\Pi^{*G_0} \cup \{Y\} \mid C) - I_{\hat{P}_D}(X;\Pi^{*G_0} \mid C) \qquad (30)$$

and that

$$I_{\hat{P}_D}(Y;\Pi^{*G_0} \cup \{X\};C) - I_{\hat{P}_D}(Y;\Pi^{*G_0};C) = I_{\hat{P}_D}(X;\Pi^{*G_0} \cup \{Y\};C)) - I_{\hat{P}_D}(X;\Pi^{*G_0};C). \qquad (31)$$

We start by showing (30). In this case, by definition of conditional mutual, we have that

$$\begin{aligned} & I_{\hat{P}_D}(Y;\Pi^{*G_0} \cup \{X\} \mid C) - I_{\hat{P}_D}(Y;\Pi^{*G_0} \mid C) = \\ &= H_{\hat{P}_D}(Y \mid C) + H_{\hat{P}_D}(\Pi^{*G_0} \cup \{X\} \mid C) - H_{\hat{P}_D}(\Pi^{*G_0} \cup \{X,Y\} \mid C) - H_{\hat{P}_D}(Y \mid C) + \\ &\qquad -H_{\hat{P}_D}(\Pi^{*G_0} \mid C) + H_{\hat{P}_D}(\Pi^{*G_0} \cup \{Y\} \mid C) \\ &= -H_{\hat{P}_D}(\Pi^{*G_0} \mid C) + H_{\hat{P}_D}(\Pi^{*G_0} \cup \{X\} \mid C) + H_{\hat{P}_D}(\Pi^{*G_0} \cup \{Y\} \mid C) - H_{\hat{P}_D}(\Pi^{*G_0} \cup \{X,Y\} \mid C) \\ &= I_{\hat{P}_D}(X;\Pi^{*G_0} \cup \{Y\} \mid C) - I_{\hat{P}_D}(X;\Pi^{*G_0} \mid C). \end{aligned}$$

Finally, each term in (31) is, by definition, given by

$$\begin{aligned} I_{\hat{P}_D}(Y;\Pi^{*G_0} \cup \{X\};C) &= I_{\hat{P}_D}(Y;\Pi^{*G_0} \cup \{X\} \mid C) - \underbrace{I_{\hat{P}_D}(Y;\Pi^{*G_0} \cup \{X\})}_{E_1} \\ I_{\hat{P}_D}(Y;\Pi^{*G_0};C) &= I_{\hat{P}_D}(Y;\Pi^{*G_0} \mid C) - \underbrace{I_{\hat{P}_D}(Y;\Pi^{*G_0})}_{E_2} \\ I_{\hat{P}_D}(X;\Pi^{*G_0} \cup \{Y\};C) &= I_{\hat{P}_D}(X;\Pi^{*G_0} \cup \{Y\} \mid C) - \underbrace{I_{\hat{P}_D}(X;\Pi^{*G_0} \cup \{Y\})}_{E_3} \\ I_{\hat{P}_D}(X;\Pi^{*G_0};C) &= I_{\hat{P}_D}(X;\Pi^{*G_0} \mid C) - \underbrace{I_{\hat{P}_D}(X;\Pi^{*G_0})}_{E_4}. \end{aligned}$$

Since by definition of mutual information we have that

$$\begin{aligned}
&I_{\hat{P}_D}(Y;\Pi^{*G_0}\cup\{X\}) - I_{\hat{P}_D}(Y;\Pi^{*G_0}) = \\
&= H_{\hat{P}_D}(Y) + H_{\hat{P}_D}(\Pi^{*G_0}\cup\{X\}) - H_{\hat{P}_D}(\Pi^{*G_0}\cup\{X,Y\}) - H_{\hat{P}_D}(Y) - H_{\hat{P}_D}(\Pi^{*G_0}) + \\
&\qquad + H_{\hat{P}_D}(\Pi^{*G_0}\cup\{Y\}) \\
&= -H_{\hat{P}_D}(\Pi^{*G_0}) + H_{\hat{P}_D}(\Pi^{*G_0}\cup\{X\}) + H_{\hat{P}_D}(\Pi^{*G_0}\cup\{Y\}) - xH_{\hat{P}_D}(\Pi^{*G_0}\cup\{X,Y\}) \\
&= I_{\hat{P}_D}(X;\Pi^{*G_0}\cup\{Y\}) - I_{\hat{P}_D}(X;\Pi^{*G_0}),
\end{aligned}$$

we know that $E_1 - E_2 = E_3 - E_4$. Thus, to prove the identity (31) it remains to show that

$$I_{\hat{P}_D}(Y;\Pi^{*G_0}\cup\{X\}\mid C) - I_{\hat{P}_D}(Y;\Pi^{*G_0}\mid C) = I_{\hat{P}_D}(X;\Pi^{*G_0}\cup\{Y\}\mid C) - I_{\hat{P}_D}(X;\Pi^{*G_0}\mid C),$$

which was already shown (in Equation (30)). This concludes the proof. □

Appendix B. Alternative Justification for Assumption 1

Observe that in the case at hand, we have some information about U_t and V_t, namely the number of times, say N_{U_t} and N_{V_t}, respectively, that U_t and V_t occur in the data set D. Moreover, we also have the number of times, say $N_{R_t} = N - (N_{U_t} + N_{V_t})$, that R_t is found in D. Given these observations, the posterior distribution of (U_t, V_t) under a uniform prior is

$$(U_t, V_t) \sim \text{Dirichlet}(N_{U_t}+1, N_{V_t}+1, N_{R_t}+1). \tag{32}$$

Furthermore, we know that N_{U_t} and N_{V_t} are, in general, a couple (or more) orders of magnitude smaller than N_{R_t}. Due to this fact, most of all probability mass of (32) is found in the square $[0,p]\times[0,p]$ for some small p.

Take as an example the (typical) case where $N_{U_t} = 1, N_{V_t} = 0, N = 500$ and

$$p = E[U_t] + \sqrt{Var[U_t]} \approx E[V_t] + \sqrt{Var[V_t]},$$

and compare the cumulative distribution of Uniform$([0,p]\times[0,p])$ with the cumulative distribution of Dirichlet$(N_{U_t}+1, N_{V_t}+1, N_{R_t}+1)$. (We provide more details in the supplementary material webpage.) Whenever N_{R_t} is much larger than N_{U_t} and N_{V_t}, the cumulative distribution Dirichlet$(N_{U_t}+1, N_{V_t}+1, N_{R_t}+1)$ is close to that of the uniform distribution Uniform$([0,p]\times[0,p])$ for some small p, and hence, we obtain approximately Assumption 1.

Concerning independence, and by assuming that the distribution of (U_t, V_t) is given by Equation (32), it results from the neutrality property of the Dirichlet distribution that

$$V_t \perp\!\!\!\perp \frac{U_t}{1-V_t}.$$

Since V_t is very small we have

$$V_t \perp\!\!\!\perp \frac{U_t}{1-V_t} \approx U_t.$$

Therefore, it is reasonable to assume that U_t and V_t are (approximately) independent.

References

A. J. Bell. The co-information lattice. In *Proc. ICA'03*, pages 921–926, 2003.

J. Bilmes. Dynamic Bayesian multinets. In *Proc. UAI'00*, pages 38–45. Morgan Kaufmann, 2000.

A. M. Carvalho. Scoring function for learning Bayesian networks. Technical report, INESC-ID Tec. Rep. 54/2009, 2009.

A. M. Carvalho, A. L. Oliveira, and M.-F. Sagot. Efficient learning of Bayesian network classifiers: An extension to the TAN classifier. In M. A. Orgun and J. Thornton, editors, *Proc. IA'07*, volume 4830 of *LNCS*, pages 16–25. Springer, 2007.

D. M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proc. UAI'95*, pages 87–98. Morgan Kaufmann, 1995.

D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: AI and Statistics V*, pages 121–130. Springer, 1996.

D. M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2006.

S. Dasgupta. Learning polytrees. In *Proc. UAI'99*, pages 134–141. Morgan Kaufmann, 1999.

L. M. de Campos. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7:2149–2187, 2006.

P. Domingos and M. J. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3):103–130, 1997.

J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B: 233–240, 1967.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. IJCAI'93*, pages 1022–1029. Morgan Kaufmann, 1993.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29 (2-3):131–163, 1997.

R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. In *Proc. AAAI/IAAI'02*, pages 167–173. AAAI Press, 2002.

R. Greiner, X. Su, B. Shen, and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59(3):297–322, 2005.

D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proc. ICML'04*, pages 46–53. ACM Press, 2004.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

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.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

A. Jakulin. *Machine Learning Based on Attribute Interactions*. PhD thesis, University of Ljubljana, 2005.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. IJCAI'95*, pages 1137–1145. Morgan Kaufmann, 1995.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.

P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. BAYDA: Software for Bayesian classification and feature selection. In *Proc. KDD'98*, pages 254–258. AAAI Press, 1998.

E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover, 1976.

W. J. McGill. Multivariate information transmission. *Psychometrika*, 19:97–116, 1954.

C. Meek. Finding a path is harder than finding a tree. *Journal of Artificial Intelligence Research*, 15:383–389, 2001.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, USA, 1988.

S. V. Pemmaraju and S. S. Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press, 2003.

F. Pernkopf and J. A. Bilmes. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *Proc. ICML'05*, pages 657–664. ACM Press, 2005.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59(3):267–296, 2005.

T. Silander, T. Roos, and P. Myllymäki. Learning locally minimax optimal Bayesian networks. *International Journal of Approximate Reasoning*, 51(5):544–557, 2010.

J. Su and H. Zhang. Full Bayesian network classifiers. In *Proc. ICML'06*, pages 897–904. ACM Press, 2006.

J. Su, H. Zhang, C. X. Ling, and S. Matwin. Discriminative parameter learning for Bayesian networks. In *Proc ICML'08*, pages 1016–1023. ACM Press, 2008.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proc. UAI'90*, pages 255–270. Elsevier, 1990.

S. Yang and K.-C. Chang. Comparison of score metrics for Bayesian network learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 32(3):419–428, 2002.

Journal of Machine Learning Research 12 (2011) 2211-2268 Submitted 12/09; Revised 9/10; Published 7/11

Multiple Kernel Learning Algorithms

Mehmet Gönen GONEN@BOUN.EDU.TR
Ethem Alpaydın ALPAYDIN@BOUN.EDU.TR
Department of Computer Engineering
Boğaziçi University
TR-34342 Bebek, İstanbul, Turkey

Editor: Francis Bach

Abstract

In recent years, several methods have been proposed to combine multiple kernels instead of using a single one. These different kernels may correspond to using different notions of similarity or may be using information coming from multiple sources (different representations or different feature subsets). In trying to organize and highlight the similarities and differences between them, we give a taxonomy of and review several multiple kernel learning algorithms. We perform experiments on real data sets for better illustration and comparison of existing algorithms. We see that though there may not be large differences in terms of accuracy, there is difference between them in complexity as given by the number of stored support vectors, the sparsity of the solution as given by the number of used kernels, and training time complexity. We see that overall, using multiple kernels instead of a single one is useful and believe that combining kernels in a nonlinear or data-dependent way seems more promising than linear combination in fusing information provided by simple linear kernels, whereas linear methods are more reasonable when combining complex Gaussian kernels.

Keywords: support vector machines, kernel machines, multiple kernel learning

1. Introduction

The support vector machine (SVM) is a discriminative classifier proposed for binary classification problems and is based on the theory of structural risk minimization (Vapnik, 1998). Given a sample of N independent and identically distributed training instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i$ is the D-dimensional input vector and $y_i \in \{-1, +1\}$ is its class label, SVM basically finds the linear discriminant with the maximum margin in the feature space induced by the mapping function $\Phi\colon \mathbb{R}^D \to \mathbb{R}^S$. The resulting discriminant function is

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b.$$

The classifier can be trained by solving the following quadratic optimization problem:

$$\begin{aligned} \text{minimize } & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^N \xi_i \\ \text{with respect to } & \mathbf{w} \in \mathbb{R}^S,\ \xi \in \mathbb{R}_+^N,\ b \in \mathbb{R} \\ \text{subject to } & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i \quad \forall i \end{aligned}$$

where $\mathbf{w}$ is the vector of weight coefficients, C is a predefined positive trade-off parameter between model simplicity and classification error, ξ is the vector of slack variables, and b is the bias term of the separating hyperplane. Instead of solving this optimization problem directly, the Lagrangian dual function enables us to obtain the following dual formulation:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \underbrace{\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle}_{k(\mathbf{x}_i, \mathbf{x}_j)} \\
\text{with respect to} \quad & \alpha \in \mathbb{R}_{+}^{N} \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned}$$

where $k\colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is named the *kernel function* and α is the vector of dual variables corresponding to each separation constraint. Solving this, we get $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$ and the discriminant function can be rewritten as

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b.$$

There are several kernel functions successfully used in the literature, such as the linear kernel (k_{LIN}), the polynomial kernel (k_{POL}), and the Gaussian kernel (k_{GAU}):

$$\begin{aligned}
k_{LIN}(\mathbf{x}_i, \mathbf{x}_j) &= \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\
k_{POL}(\mathbf{x}_i, \mathbf{x}_j) &= (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^q, \qquad q \in \mathbb{N} \\
k_{GAU}(\mathbf{x}_i, \mathbf{x}_j) &= \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / s^2\right), \qquad s \in \mathbb{R}_{++}.
\end{aligned}$$

There are also kernel functions proposed for particular applications, such as natural language processing (Lodhi et al., 2002) and bioinformatics (Schölkopf et al., 2004).

Selecting the kernel function $k(\cdot, \cdot)$ and its parameters (e.g., q or s) is an important issue in training. Generally, a cross-validation procedure is used to choose the best performing kernel function among a set of kernel functions on a separate validation set different from the training set. In recent years, multiple kernel learning (MKL) methods have been proposed, where we use multiple kernels instead of selecting one specific kernel function and its corresponding parameters:

$$k_{\eta}(\mathbf{x}_i, \mathbf{x}_j) = f_{\eta}(\{k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)\}_{m=1}^{P})$$

where the combination function, $f_{\eta}\colon \mathbb{R}^P \to \mathbb{R}$, can be a linear or a nonlinear function. Kernel functions, $\{k_m\colon \mathbb{R}^{D_m} \times \mathbb{R}^{D_m} \to \mathbb{R}\}_{m=1}^{P}$, take P feature representations (not necessarily different) of data instances: $\mathbf{x}_i = \{\mathbf{x}_i^m\}_{m=1}^{P}$ where $\mathbf{x}_i^m \in \mathbb{R}^{D_m}$, and D_m is the dimensionality of the corresponding feature representation. η parameterizes the combination function and the more common implementation is

$$k_{\eta}(\mathbf{x}_i, \mathbf{x}_j) = f_{\eta}(\{k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)\}_{m=1}^{P} | \eta)$$

where the parameters are used to combine a set of predefined kernels (i.e., we know the kernel functions and corresponding kernel parameters before training). It is also possible to view this as

$$k_{\eta}(\mathbf{x}_i, \mathbf{x}_j) = f_{\eta}(\{k_m(\mathbf{x}_i^m, \mathbf{x}_j^m | \eta)\}_{m=1}^{P})$$

where the parameters integrated into the kernel functions are optimized during training. Most of the existing MKL algorithms fall into the first category and try to combine predefined kernels in an optimal way. We will discuss the algorithms in terms of the first formulation but give the details of the algorithms that use the second formulation where appropriate.

The reasoning is similar to combining different classifiers: Instead of choosing a single kernel function and putting all our eggs in the same basket, it is better to have a set and let an algorithm do the picking or combination. There can be two uses of MKL: (a) Different kernels correspond to different notions of similarity and instead of trying to find which works best, a learning method does the picking for us, or may use a combination of them. Using a specific kernel may be a source of bias, and in allowing a learner to choose among a set of kernels, a better solution can be found. (b) Different kernels may be using inputs coming from different representations possibly from different sources or modalities. Since these are different representations, they have different measures of similarity corresponding to different kernels. In such a case, combining kernels is one possible way to combine multiple information sources. Noble (2004) calls this method of combining kernels *intermediate combination* and contrasts this with *early combination* (where features from different sources are concatenated and fed to a single learner) and *late combination* (where different features are fed to different classifiers whose decisions are then combined by a fixed or trained combiner).

There is significant amount of work in the literature for combining multiple kernels. Section 2 identifies the key properties of the existing MKL algorithms in order to construct a taxonomy, highlighting similarities and differences between them. Section 3 categorizes and discusses the existing MKL algorithms with respect to this taxonomy. We give experimental results in Section 4 and conclude in Section 5. The lists of acronyms and notation used in this paper are given in Appendices A and B, respectively.

2. Key Properties of Multiple Kernel Learning

We identify and explain six key properties of the existing MKL algorithms in order to obtain a meaningful categorization. We can think of these six dimensions (though not necessarily orthogonal) defining a space in which we can situate the existing MKL algorithms and search for structure (i.e., groups) to better see the similarities and differences between them. These properties are the learning method, the functional form, the target function, the training method, the base learner, and the computational complexity.

2.1 The Learning Method

The existing MKL algorithms use different learning methods for determining the kernel combination function. We basically divide them into five major categories:

1. *Fixed rules* are functions without any parameters (e.g., summation or multiplication of the kernels) and do not need any training.

2. *Heuristic approaches* use a parameterized combination function and find the parameters of this function generally by looking at some measure obtained from each kernel function separately. These measures can be calculated from the kernel matrices or taken as the performance values of the single kernel-based learners trained separately using each kernel.

3. *Optimization approaches* also use a parametrized combination function and learn the parameters by solving an optimization problem. This optimization can be integrated to a kernel-based learner or formulated as a different mathematical model for obtaining only the combination parameters.

4. *Bayesian approaches* interpret the kernel combination parameters as random variables, put priors on these parameters, and perform inference for learning them and the base learner parameters.

5. *Boosting approaches*, inspired from ensemble and boosting methods, iteratively add a new kernel until the performance stops improving.

2.2 The Functional Form

There are different ways in which the combination can be done and each has its own combination parameter characteristics. We group functional forms of the existing MKL algorithms into three basic categories:

1. *Linear combination* methods are the most popular and have two basic categories: unweighted sum (i.e., using sum or mean of the kernels as the combined kernel) and weighted sum. In the weighted sum case, we can linearly parameterize the combination function:

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = f_\eta(\{k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)\}_{m=1}^P | \eta) = \sum_{m=1}^{P} \eta_m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)$$

   where η denotes the kernel weights. Different versions of this approach differ in the way they put restrictions on η: the linear sum (i.e., $\eta \in \mathbb{R}^P$), the conic sum (i.e., $\eta \in \mathbb{R}_+^P$), or the convex sum (i.e., $\eta \in \mathbb{R}_+^P$ and $\sum_{m=1}^P \eta_m = 1$). As can be seen, the conic sum is a special case of the linear sum and the convex sum is a special case of the conic sum. The conic and convex sums have two advantages over the linear sum in terms of interpretability. First, when we have positive kernel weights, we can extract the relative importance of the combined kernels by looking at them. Second, when we restrict the kernel weights to be nonnegative, this corresponds to scaling the feature spaces and using the concatenation of them as the combined feature representation:

$$\Phi_\eta(\mathbf{x}) = \begin{pmatrix} \sqrt{\eta_1}\Phi_1(\mathbf{x}^1) \\ \sqrt{\eta_2}\Phi_2(\mathbf{x}^2) \\ \vdots \\ \sqrt{\eta_P}\Phi_P(\mathbf{x}^P) \end{pmatrix}$$

   and the dot product in the combined feature space gives the combined kernel:

$$\langle \Phi_\eta(\mathbf{x}_i), \Phi_\eta(\mathbf{x}_j) \rangle = \begin{pmatrix} \sqrt{\eta_1}\Phi_1(\mathbf{x}_i^1) \\ \sqrt{\eta_2}\Phi_2(\mathbf{x}_i^2) \\ \vdots \\ \sqrt{\eta_P}\Phi_P(\mathbf{x}_i^P) \end{pmatrix}^\top \begin{pmatrix} \sqrt{\eta_1}\Phi_1(\mathbf{x}_j^1) \\ \sqrt{\eta_2}\Phi_2(\mathbf{x}_j^2) \\ \vdots \\ \sqrt{\eta_P}\Phi_P(\mathbf{x}_j^P) \end{pmatrix} = \sum_{m=1}^{P} \eta_m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m).$$

The combination parameters can also be restricted using extra constraints, such as the ℓ_p-norm on the kernel weights or trace restriction on the combined kernel matrix, in addition to their domain definitions. For example, the ℓ_1-norm promotes sparsity on the kernel level, which can be interpreted as feature selection when the kernels use different feature subsets.

2. *Nonlinear combination* methods use nonlinear functions of kernels, namely, multiplication, power, and exponentiation.

3. *Data-dependent combination* methods assign specific kernel weights for each data instance. By doing this, they can identify local distributions in the data and learn proper kernel combination rules for each region.

2.3 The Target Function

We can optimize different target functions when selecting the combination function parameters. We group the existing target functions into three basic categories:

1. *Similarity-based functions* calculate a similarity metric between the combined kernel matrix and an optimum kernel matrix calculated from the training data and select the combination function parameters that maximize the similarity. The similarity between two kernel matrices can be calculated using kernel alignment, Euclidean distance, Kullback-Leibler (KL) divergence, or any other similarity measure.

2. *Structural risk functions* follow the structural risk minimization framework and try to minimize the sum of a regularization term that corresponds to the model complexity and an error term that corresponds to the system performance. The restrictions on kernel weights can be integrated into the regularization term. For example, structural risk function can use the ℓ_1-norm, the ℓ_2-norm, or a mixed-norm on the kernel weights or feature spaces to pick the model parameters.

3. *Bayesian functions* measure the quality of the resulting kernel function constructed from candidate kernels using a Bayesian formulation. We generally use the likelihood or the posterior as the target function and find the maximum likelihood estimate or the maximum a posteriori estimate to select the model parameters.

2.4 The Training Method

We can divide the existing MKL algorithms into two main groups in terms of their training methodology:

1. *One-step methods* calculate both the combination function parameters and the parameters of the combined base learner in a single pass. One can use a sequential approach or a simultaneous approach. In the sequential approach, the combination function parameters are determined first, and then a kernel-based learner is trained using the combined kernel. In the simultaneous approach, both set of parameters are learned together.

2. *Two-step methods* use an iterative approach where each iteration, first we update the combination function parameters while fixing the base learner parameters, and then we update the base learner parameters while fixing the combination function parameters. These two steps are repeated until convergence.

2.5 The Base Learner

There are many kernel-based learning algorithms proposed in the literature and all of them can be transformed into an MKL algorithm, in one way or another.

The most commonly used base learners are SVM and support vector regression (SVR), due to their empirical success, their ease of applicability as a building block in two-step methods, and their ease of transformation to other optimization problems as a one-step training method using the simultaneous approach. Kernel Fisher discriminant analysis (KFDA), regularized kernel discriminant analysis (RKDA), and kernel ridge regression (KRR) are three other popular methods used in MKL.

Multinomial probit and Gaussian process (GP) are generally used in Bayesian approaches. New inference algorithms are developed for modified probabilistic models in order to learn both the combination function parameters and the base learner parameters.

2.6 The Computational Complexity

The computational complexity of an MKL algorithm mainly depends on its training method (i.e., whether it is one-step or two-step) and the computational complexity of its base learner.

One-step methods using fixed rules and heuristics generally do not spend much time to find the combination function parameters, and the overall complexity is determined by the complexity of the base learner to a large extent. One-step methods that use optimization approaches to learn combination parameters have high computational complexity, due to the fact that they are generally modeled as a semidefinite programming (SDP) problem, a quadratically constrained quadratic programming (QCQP) problem, or a second-order cone programming (SOCP) problem. These problems are much harder to solve than a quadratic programming (QP) problem used in the case of the canonical SVM.

Two-step methods update the combination function parameters and the base learner parameters in an alternating manner. The combination function parameters are generally updated by solving an optimization problem or using a closed-form update rule. Updating the base learner parameters usually requires training a kernel-based learner using the combined kernel. For example, they can be modeled as a semi-infinite linear programming (SILP) problem, which uses a generic linear programming (LP) solver and a canonical SVM solver in the inner loop.

3. Multiple Kernel Learning Algorithms

In this section, we categorize the existing MKL algorithms in the literature into 12 groups depending on the six key properties discussed in Section 2. We first give a summarizing table (see Tables 1 and 2) containing 49 representative references and then give a more detailed discussion of each group in a separate section reviewing a total of 96 references.

3.1 Fixed Rules

Fixed rules obtain $k_\eta(\cdot,\cdot)$ using $f_\eta(\cdot)$ and then train a canonical kernel machine with the kernel matrix calculated using $k_\eta(\cdot,\cdot)$. For example, we can obtain a valid kernel by taking the *summation* or *multiplication* of two valid kernels (Cristianini and Shawe-Taylor, 2000):

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) + k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)$$
$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) k_2(\mathbf{x}_i^2, \mathbf{x}_j^2). \quad (1)$$

We know that a matrix $\mathbf{K}$ is positive semidefinite if and only if $\upsilon^\top \mathbf{K} \upsilon \geq 0$, for all $\upsilon \in \mathbb{R}^N$. Trivially, we can see that $k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) + k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)$ gives a positive semidefinite kernel matrix:

$$\upsilon^\top \mathbf{K}_\eta \upsilon = \upsilon^\top (\mathbf{K}_1 + \mathbf{K}_2) \upsilon = \upsilon^\top \mathbf{K}_1 \upsilon + \upsilon^\top \mathbf{K}_2 \upsilon \geq 0$$

and $k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)$ also gives a positive semidefinite kernel due to the fact that the element-wise product between two positive semidefinite matrices results in another positive semidefinite matrix:

$$\upsilon^\top \mathbf{K}_\eta \upsilon = \upsilon^\top (\mathbf{K}_1 \odot \mathbf{K}_2) \upsilon \geq 0.$$

We can apply the rules in (1) recursively to obtain the rules for more than two kernels. For example, the summation or multiplication of P kernels is also a valid kernel:

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P} k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)$$
$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \prod_{m=1}^{P} k_m(\mathbf{x}_i^m, \mathbf{x}_j^m).$$

Pavlidis et al. (2001) report that on a gene functional classification task, training an SVM with an unweighted sum of heterogeneous kernels gives better results than the combination of multiple SVMs each trained with one of these kernels.

We need to calculate the similarity between pairs of objects such as genes or proteins especially in bioinformatics applications. *Pairwise kernels* are proposed to express the similarity between pairs in terms of similarities between individual objects. Two pairs are said to be similar when each object in one pair is similar to one object in the other pair. This approach can be encoded as a pairwise kernel using a kernel function between individual objects, called the *genomic kernel* (Ben-Hur and Noble, 2005), as follows:

$$k^P(\{\mathbf{x}_i^a, \mathbf{x}_j^a\}, \{\mathbf{x}_i^b, \mathbf{x}_j^b\}) = k(\mathbf{x}_i^a, \mathbf{x}_i^b) k(\mathbf{x}_j^a, \mathbf{x}_j^b) + k(\mathbf{x}_i^a, \mathbf{x}_j^b) k(\mathbf{x}_j^a, \mathbf{x}_i^b).$$

Ben-Hur and Noble (2005) combine pairwise kernels in two different ways: (a) using an unweighted sum of different pairwise kernels:

$$k_\eta^P(\{\mathbf{x}_i^a, \mathbf{x}_j^a\}, \{\mathbf{x}_i^b, \mathbf{x}_j^b\}) = \sum_{m=1}^{P} k_m^P(\{\mathbf{x}_i^a, \mathbf{x}_j^a\}, \{\mathbf{x}_i^b, \mathbf{x}_j^b\})$$

and (b) using an unweighted sum of different genomic kernels in the pairwise kernel:

$$\begin{aligned} & k_\eta^P(\{\mathbf{x}_i^a, \mathbf{x}_j^a\}, \{\mathbf{x}_i^b, \mathbf{x}_j^b\}) \\ &\quad = \left(\sum_{m=1}^{P} k_m(\mathbf{x}_i^a, \mathbf{x}_i^b)\right)\left(\sum_{m=1}^{P} k_m(\mathbf{x}_j^a, \mathbf{x}_j^b)\right) + \left(\sum_{m=1}^{P} k_m(\mathbf{x}_i^a, \mathbf{x}_j^b)\right)\left(\sum_{m=1}^{P} k_m(\mathbf{x}_j^a, \mathbf{x}_i^b)\right) \\ &\quad = k_\eta(\mathbf{x}_i^a, \mathbf{x}_i^b) k_\eta(\mathbf{x}_j^a, \mathbf{x}_j^b) + k_\eta(\mathbf{x}_i^a, \mathbf{x}_j^b) k_\eta(\mathbf{x}_j^a, \mathbf{x}_i^b). \end{aligned}$$

The combined pairwise kernels improve the classification performance for protein-protein interaction prediction task.

Sec.	Representative References	Learning Method	Functional Form	Target Function	Training Method	Base Learner	Computational Complexity
3.1	Pavlidis et al. (2001)	Fixed	Lin. (unwei.)	None	1-step (seq.)	SVM	QP
	Ben-Hur and Noble (2005)	Fixed	Lin. (unwei.)	None	1-step (seq.)	SVM	QP
3.2	de Diego et al. (2004, 2010a)	Heuristic	Nonlinear	Val. error	2-step	SVM	QP
	Moguerza et al. (2004); de Diego et al. (2010a)	Heuristic	Data-dep.	None	1-step (seq.)	SVM	QP
	Tanabe et al. (2008)	Heuristic	Lin. (convex)	None	1-step (seq.)	SVM	QP
	Qiu and Lane (2009)	Heuristic	Lin. (convex)	None	1-step (seq.)	SVR	QP
	Qiu and Lane (2009)	Heuristic	Lin. (convex)	None	1-step (seq.)	SVM	QP
3.3	Lanckriet et al. (2004a)	Optim.	Lin. (linear)	Similarity	1-step (seq.)	SVM	SDP+QP
	Igel et al. (2007)	Optim.	Lin. (linear)	Similarity	1-step (seq.)	SVM	Grad.+QP
	Cortes et al. (2010a)	Optim.	Lin. (linear)	Similarity	1-step (seq.)	SVM	Mat. Inv.+QP
3.4	Lanckriet et al. (2004a)	Optim.	Lin. (conic)	Similarity	1-step (seq.)	SVM	QCQP+QP
	Kandola et al. (2002)	Optim.	Lin. (conic)	Similarity	1-step (seq.)	SVM	QP+QP
	Cortes et al. (2010a)	Optim.	Lin. (conic)	Similarity	1-step (seq.)	SVM	QP+QP
3.5	He et al. (2008)	Optim.	Lin. (convex)	Similarity	1-step (seq.)	SVM	QP+QP
	Tanabe et al. (2008)	Optim.	Lin. (convex)	Similarity	1-step (seq.)	SVM	QP+QP
	Ying et al. (2009)	Optim.	Lin. (convex)	Similarity	1-step (seq.)	SVM	Grad.+QP
3.6	Lanckriet et al. (2002)	Optim.	Lin. (linear)	Str. risk	1-step (seq.)	SVM	SDP+QP
	Qiu and Lane (2005)	Optim.	Lin. (linear)	Str. risk	1-step (seq.)	SVR	SDP+QP
	Conforti and Guido (2010)	Optim.	Lin. (linear)	Str. risk	1-step (seq.)	SVM	SDP+QP
3.7	Lanckriet et al. (2004a)	Optim.	Lin. (conic)	Str. risk	1-step (seq.)	SVM	QCQP+QP
	Fung et al. (2004)	Optim.	Lin. (conic)	Str. risk	2-step	KFDA	QP+Mat. Inv.
	Tsuda et al. (2004)	Optim.	Lin. (conic)	Str. risk	2-step	KFDA	Grad.+Mat. Inv.
	Qiu and Lane (2005)	Optim.	Lin. (conic)	Str. risk	1-step (seq.)	SVR	QCQP+QP
	Varma and Ray (2007)	Optim.	Lin. (conic)	Str. risk	1-step (sim.)	SVM	SOCP
	Varma and Ray (2007)	Optim.	Lin. (conic)	Str. risk	2-step	SVM	Grad.+QP
	Cortes et al. (2009)	Optim.	Lin. (conic)	Str. risk	2-step	KRR	Grad.+Mat. Inv.
	Kloft et al. (2010a)	Optim.	Lin. (conic)	Str. risk	2-step	SVM	Newton+QP
	Xu et al. (2010b)	Optim.	Lin. (conic)	Str. risk	1-step (sim.)	SVM	Grad.
	Kloft et al. (2010b); Xu et al. (2010a)	Optim.	Lin. (conic)	Str. risk	2-step	SVM	Analytical+QP
	Conforti and Guido (2010)	Optim.	Lin. (conic)	Str. risk	1-step (seq.)	SVM	QCQP+QP

Table 1: Representative MKL algorithms.

Sec.	Representative References	Learning Method	Functional Form	Target Function	Training Method	Base Learner	Computational Complexity
3.8	Bousquet and Herrmann (2003)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	Grad.+QP
	Bach et al. (2004)	Optim.	Lin. (convex)	Str. risk	1-step (sim.)	SVM	SOCP
	Sonnenburg et al. (2006a,b)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	LP+QP
	Kim et al. (2006)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	KFDA	SDP+Mat. Inv.
	Ye et al. (2007a)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	RKDA	SDP+Mat. Inv.
	Ye et al. (2007b)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	RKDA	QCQP+Mat. Inv.
	Ye et al. (2008)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	RKDA	SILP+Mat. Inv.
	Rakotomamonjy et al. (2007, 2008)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	Grad.+QP
	Chapelle and Rakotomamonjy (2008)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	QP+QP
	Kloft et al. (2010b); Xu et al. (2010a)	Optim.	Lin. (convex)	Str. risk	2-step	SVM	Analytical+QP
	Conforti and Guido (2010)	Optim.	Lin. (convex)	Str. risk	1-step (seq.)	SVM	QCQP+QP
3.9	Lee et al. (2007)	Optim.	Nonlinear	Str. risk	1-step (sim.)	SVM	QP
	Varma and Babu (2009)	Optim.	Nonlinear	Str. risk	2-step	SVM	Grad.+QP
	Cortes et al. (2010b)	Optim.	Nonlinear	Str. risk	2-step	KRR	Grad.+Mat. Inv.
3.10	Lewis et al. (2006b)	Optim.	Data-dep.	Str. risk	1-step (sim.)	SVM	QP
	Gönen and Alpaydın (2008)	Optim.	Data-dep.	Str. risk	2-step	SVM	Grad.+QP
	Yang et al. (2009a)	Optim.	Data-dep.	Str. risk	2-step	SVM	Grad.+QP
	Yang et al. (2009b, 2010)	Optim.	Data-dep.	Str. risk	2-step	SVM	SILP+QP
3.11	Girolami and Rogers (2005)	Bayesian	Lin. (conic)	Likelihood	Inference	KRR	Approximation
	Girolami and Zhong (2007)	Bayesian	Lin. (conic)	Likelihood	Inference	GP	Approximation
	Christoudias et al. (2009)	Bayesian	Data-dep.	Likelihood	Inference	GP	Approximation
3.12	Bennett et al. (2002)	Boosting	Data-dep.	Str. risk	$P \times$ 1-step	KRR	Mat. Inv.
	Crammer et al. (2003)	Boosting	Lin. (conic)	Str. risk	$P \times$ 1-step	Percept.	Eigenvalue Prob.
	Bi et al. (2004)	Boosting	Lin. (linear)	Str. risk	$P \times$ 1-step	SVM	QP

Table 2: Representative MKL algorithms (continued).

3.2 Heuristic Approaches

de Diego et al. (2004, 2010a) define a functional form of combining two kernels:

$$\mathbf{K}_\eta = \frac{1}{2}(\mathbf{K}_1 + \mathbf{K}_2) + f(\mathbf{K}_1 - \mathbf{K}_2)$$

where the term $f(\mathbf{K}_1 - \mathbf{K}_2)$ represents the difference of information between what $\mathbf{K}_1$ and $\mathbf{K}_2$ provide for classification. They investigate three different functions:

$$\begin{aligned}
k_\eta(\mathbf{x}_i, \mathbf{x}_j) &= \frac{1}{2}(k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) + k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)) + \tau y_i y_j |k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) - k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)| \\
k_\eta(\mathbf{x}_i, \mathbf{x}_j) &= \frac{1}{2}(k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) + k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)) + \tau y_i y_j (k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) - k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)) \\
\mathbf{K}_\eta &= \frac{1}{2}(\mathbf{K}_1 + \mathbf{K}_2) + \tau(\mathbf{K}_1 - \mathbf{K}_2)(\mathbf{K}_1 - \mathbf{K}_2)
\end{aligned}$$

where $\tau \in \mathbb{R}_+$ is the parameter that represents the weight assigned to the term $f(\mathbf{K}_1 - \mathbf{K}_2)$ (selected through cross-validation) and the first two functions do not ensure having positive semidefinite kernel matrices. It is also possible to combine more than two kernel functions by applying these rules recursively.

Moguerza et al. (2004) and de Diego et al. (2010a) propose a matrix functional form of combining kernels:

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P} \eta_m(\mathbf{x}_i, \mathbf{x}_j) k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)$$

where $\eta_m(\cdot,\cdot)$ assigns a weight to $k_m(\cdot,\cdot)$ according to $\mathbf{x}_i$ and $\mathbf{x}_j$. They propose different heuristics to estimate the weighing function values using conditional class probabilities, $\Pr(y_i = y_j | \mathbf{x}_i)$ and $\Pr(y_j = y_i | \mathbf{x}_j)$, calculated with a nearest-neighbor approach. However, each kernel function corresponds to a different neighborhood and $\eta_m(\cdot,\cdot)$ is calculated on the neighborhood induced by $k_m(\cdot,\cdot)$. For an unlabeled data instance $\mathbf{x}$, they take its class label once as $+1$ and once as -1, calculate the discriminant values $f(\mathbf{x}|y = +1)$ and $f(\mathbf{x}|y = -1)$, and assign it to the class that has more confidence in its decision (i.e., by selecting the class label with greater $yf(\mathbf{x}|y)$ value). de Diego et al. (2010b) use this method to fuse information from several feature representations for face verification. Combining kernels in a data-dependent manner outperforms the classical fusion techniques such as feature-level and score-level methods in their experiments.

We can also use a linear combination instead of a data-dependent combination and formulate the combined kernel function as follows:

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P} \eta_m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)$$

where we select the kernel weights by looking at the performance values obtained by each kernel separately. For example, Tanabe et al. (2008) propose the following rule in order to choose the kernel weights for classification problems:

$$\eta_m = \frac{\pi_m - \delta}{\sum\limits_{h=1}^{P} (\pi_h - \delta)}$$

where π_m is the accuracy obtained using only $\mathbf{K}_m$, and δ is the threshold that should be less than or equal to the minimum of the accuracies obtained from single-kernel learners. Qiu and Lane (2009) propose two simple heuristics to select the kernel weights for regression problems:

$$\eta_m = \frac{R_m}{\sum\limits_{h=1}^{P} R_h} \quad \forall m$$

$$\eta_m = \frac{\sum\limits_{h=1}^{P} M_h - M_m}{(P-1)\sum\limits_{h=1}^{P} M_h} \quad \forall m$$

where R_m is the Pearson correlation coefficient between the true outputs and the predicted labels generated by the regressor using the kernel matrix $\mathbf{K}_m$, and M_m is the mean square error generated by the regressor using the kernel matrix $\mathbf{K}_m$. These three heuristics find a convex combination of the input kernels as the combined kernel.

Cristianini et al. (2002) define a notion of similarity between two kernels called *kernel alignment*. The empirical alignment of two kernels is calculated as follows:

$$\mathrm{A}(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle_F \langle \mathbf{K}_2, \mathbf{K}_2 \rangle_F}}$$

where $\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F = \sum_{i=1}^{N} \sum_{j=1}^{N} k_1(\mathbf{x}_i^1, \mathbf{x}_j^1) k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)$. This similarity measure can be seen as the cosine of the angle between $\mathbf{K}_1$ and $\mathbf{K}_2$. $\mathbf{y}\mathbf{y}^\top$ can be defined as *ideal kernel* for a binary classification task, and the alignment between a kernel and the ideal kernel becomes

$$\mathrm{A}(\mathbf{K}, \mathbf{y}\mathbf{y}^\top) = \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^\top \rangle_F}{\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F \langle \mathbf{y}\mathbf{y}^\top, \mathbf{y}\mathbf{y}^\top \rangle_F}} = \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^\top \rangle_F}{N\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F}}.$$

Kernel alignment has one key property due to concentration (i.e., the probability of deviation from the mean decays exponentially), which enables us to keep high alignment on a test set when we optimize it on a training set.

Qiu and Lane (2009) propose the following simple heuristic for classification problems to select the kernel weights using kernel alignment:

$$\eta_m = \frac{\mathrm{A}(\mathbf{K}_m, \mathbf{y}\mathbf{y}^\top)}{\sum\limits_{h=1}^{P} \mathrm{A}(\mathbf{K}_h, \mathbf{y}\mathbf{y}^\top)} \quad \forall m \tag{2}$$

where we obtain the combined kernel as a convex combination of the input kernels.

3.3 Similarity Optimizing Linear Approaches with Arbitrary Kernel Weights

Lanckriet et al. (2004a) propose to optimize the kernel alignment as follows:

$$\begin{aligned} \text{maximize} \quad & \text{A}(\mathbf{K}_{\eta}^{\text{tra}}, \mathbf{y}\mathbf{y}^{\top}) \\ \text{with respect to} \quad & \mathbf{K}_{\eta} \in \mathbb{S}^N \\ \text{subject to} \quad & \text{tr}\left(\mathbf{K}_{\eta}\right) = 1 \\ & \mathbf{K}_{\eta} \succeq 0 \end{aligned}$$

where the trace of the combined kernel matrix is arbitrarily set to 1. This problem can be converted into the following SDP problem using arbitrary kernel weights in the combination:

$$\begin{aligned} \text{maximize} \quad & \left\langle \sum_{m=1}^{P} \eta_m \mathbf{K}_m^{\text{tra}}, \mathbf{y}\mathbf{y}^{\top} \right\rangle_F \\ \text{with respect to} \quad & \eta \in \mathbb{R}^P, \ \ \mathbf{A} \in \mathbb{S}^N \\ \text{subject to} \quad & \text{tr}\left(\mathbf{A}\right) \leq 1 \\ & \begin{pmatrix} \mathbf{A} & \sum\limits_{m=1}^{P} \eta_m \mathbf{K}_m^{\top} \\ \sum\limits_{m=1}^{P} \eta_m \mathbf{K}_m & \mathbf{I} \end{pmatrix} \succeq 0 \\ & \sum_{m=1}^{P} \eta_m \mathbf{K}_m \succeq 0. \end{aligned}$$

Igel et al. (2007) propose maximizing the kernel alignment using gradient-based optimization. They calculate the gradients with respect to the kernel parameters as

$$\frac{\partial A(\mathbf{K}_{\eta}, \mathbf{y}\mathbf{y}^{\top})}{\partial \eta_m} = \frac{\left\langle \frac{\partial \mathbf{K}_{\eta}}{\partial \eta_m}, \mathbf{y}\mathbf{y}^{\top} \right\rangle_F \langle \mathbf{K}_{\eta}, \mathbf{K}_{\eta} \rangle_F - \langle \mathbf{K}_{\eta}, \mathbf{y}\mathbf{y}^{\top} \rangle_F \left\langle \frac{\partial \mathbf{K}_{\eta}}{\partial \eta_m}, \mathbf{K}_{\eta} \right\rangle_F}{N \sqrt{\langle \mathbf{K}_{\eta}, \mathbf{K}_{\eta} \rangle_F^3}}.$$

In a transcription initiation site detection task for bacterial genes, they obtain better results by optimizing the kernel weights of the combined kernel function that is composed of six sequence kernels, using the gradient above.

Cortes et al. (2010a) give a different kernel alignment definition, which they call *centered-kernel alignment*. The empirical centered-alignment of two kernels is calculated as follows:

$$\text{CA}(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1^c, \mathbf{K}_2^c \rangle_F}{\sqrt{\langle \mathbf{K}_1^c, \mathbf{K}_1^c \rangle_F \langle \mathbf{K}_2^c, \mathbf{K}_2^c \rangle_F}}$$

where $\mathbf{K}^c$ is the centered version of $\mathbf{K}$ and can be calculated as

$$\mathbf{K}^c = \mathbf{K} - \frac{1}{N}\mathbf{1}\mathbf{1}^{\top}\mathbf{K} - \frac{1}{N}\mathbf{K}\mathbf{1}\mathbf{1}^{\top} + \frac{1}{N^2}(\mathbf{1}^{\top}\mathbf{K}\mathbf{1})\mathbf{1}\mathbf{1}^{\top}$$

where **1** is the vector of ones with proper dimension. Cortes et al. (2010a) also propose to optimize the centered-kernel alignment as follows:

$$\begin{aligned} \text{maximize} \;\; & \text{CA}(\mathbf{K}_{\eta}, \mathbf{y}\mathbf{y}^{\top}) \\ \text{with respect to} \;\; & \eta \in \mathcal{M} \end{aligned} \tag{3}$$

where $\mathcal{M} = \{\eta \colon \|\eta\|_2 = 1\}$. This optimization problem (3) has an analytical solution:

$$\eta = \frac{\mathbf{M}^{-1}\mathbf{a}}{\|\mathbf{M}^{-1}\mathbf{a}\|_2} \tag{4}$$

where $\mathbf{M} = \{\langle \mathbf{K}_m^c, \mathbf{K}_h^c \rangle_F\}_{m,h=1}^{P}$ and $\mathbf{a} = \{\langle \mathbf{K}_m^c, \mathbf{y}\mathbf{y}^{\top} \rangle_F\}_{m=1}^{P}$.

3.4 Similarity Optimizing Linear Approaches with Nonnegative Kernel Weights

Kandola et al. (2002) propose to maximize the alignment between a nonnegative linear combination of kernels and the ideal kernel. The alignment can be calculated as follows:

$$\text{A}(\mathbf{K}_{\eta}, \mathbf{y}\mathbf{y}^{\top}) = \frac{\sum\limits_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^{\top} \rangle_F}{N \sqrt{\sum\limits_{m=1}^{P} \sum\limits_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F}}.$$

We should choose the kernel weights that maximize the alignment and this idea can be cast into the following optimization problem:

$$\begin{aligned} \text{maximize} \;\; & \text{A}(\mathbf{K}_{\eta}, \mathbf{y}\mathbf{y}^{\top}) \\ \text{with respect to} \;\; & \eta \in \mathbb{R}_+^P \end{aligned}$$

and this problem is equivalent to

$$\begin{aligned} \text{maximize} \;\; & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^{\top} \rangle_F \\ \text{with respect to} \;\; & \eta \in \mathbb{R}_+^P \\ \text{subject to} \;\; & \sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F = c. \end{aligned}$$

Using the Lagrangian function, we can convert it into the following unconstrained optimization problem:

$$\begin{aligned} \text{maximize} \;\; & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^{\top} \rangle_F - \mu \left(\sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F - c \right) \\ \text{with respect to} \;\; & \eta \in \mathbb{R}_+^P. \end{aligned}$$

Kandola et al. (2002) take $\mu = 1$ arbitrarily and add a regularization term to the objective function in order to prevent overfitting. The resulting QP is very similar to the hard margin SVM optimization problem and is expected to give sparse kernel combination weights:

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^\top \rangle_F - \sum_{m=1}^{P}\sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F - \lambda \sum_{m=1}^{P} \eta_m^2 \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \end{aligned}$$

where we only learn the kernel combination weights.

Lanckriet et al. (2004a) restrict the kernel weights to be nonnegative and their SDP formulation reduces to the following QCQP problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m^{\text{tra}}, \mathbf{y}\mathbf{y}^\top \rangle_F \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P}\sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F \leq 1. \end{aligned} \tag{5}$$

Cortes et al. (2010a) also restrict the kernel weights to be nonnegative by changing the definition of $\mathcal{M}$ in (3) to $\{\eta \colon \|\eta\|_2 = 1, \; \eta \in \mathbb{R}_+^P\}$ and obtain the following QP:

$$\begin{aligned} \text{minimize} \quad & \mathbf{v}^\top \mathbf{M} \mathbf{v} - 2\mathbf{v}^\top \mathbf{a} \\ \text{with respect to} \quad & \mathbf{v} \in \mathbb{R}_+^P \end{aligned} \tag{6}$$

where the kernel weights are given by $\eta = \mathbf{v}/\|\mathbf{v}\|_2$.

3.5 Similarity Optimizing Linear Approaches with Kernel Weights on a Simplex

He et al. (2008) choose to optimize the distance between the combined kernel matrix and the ideal kernel, instead of optimizing the kernel alignment measure, using the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \langle \mathbf{K}_\eta - \mathbf{y}\mathbf{y}^\top, \mathbf{K}_\eta - \mathbf{y}\mathbf{y}^\top \rangle_F^2 \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1. \end{aligned}$$

This problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & \sum_{m=1}^{P}\sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F - 2 \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \mathbf{y}\mathbf{y}^\top \rangle_F \\ \text{with respect to} \quad & \eta \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1. \end{aligned} \tag{7}$$

Nguyen and Ho (2008) propose another quality measure called feature space-based kernel matrix evaluation measure (FSM) defined as

$$\text{FSM}(\mathbf{K}, \mathbf{y}) = \frac{s_+ + s_-}{\|\mathbf{m}_+ - \mathbf{m}_-\|_2}$$

where $\{s_+, s_-\}$ are the standard deviations of the positive and negative classes, and $\{\mathbf{m}_+, \mathbf{m}_-\}$ are the class centers in the feature space. Tanabe et al. (2008) optimize the kernel weights for the convex combination of kernels by minimizing this measure:

$$\begin{aligned} \text{minimize} \;\; & \text{FSM}(\mathbf{K}_\eta, \mathbf{y}) \\ \text{with respect to} \;\; & \eta \in \mathbb{R}_+^P \\ \text{subject to} \;\; & \sum_{m=1}^{P} \eta_m = 1. \end{aligned}$$

This method gives similar performance results when compared to the SMO-like algorithm of Bach et al. (2004) for a protein-protein interaction prediction problem using much less time and memory.

Ying et al. (2009) follow an information-theoretic approach based on the KL divergence between the combined kernel matrix and the optimal kernel matrix:

$$\begin{aligned} \text{minimize} \;\; & \text{KL}(\mathcal{N}(\mathbf{0}, \mathbf{K}_\eta) \| \mathcal{N}(\mathbf{0}, \mathbf{y}\mathbf{y}^\top)) \\ \text{with respect to} \;\; & \eta \in \mathbb{R}_+^P \\ \text{subject to} \;\; & \sum_{m=1}^{P} \eta_m = 1 \end{aligned}$$

where $\mathbf{0}$ is the vector of zeros with proper dimension. The kernel combinations weights can be optimized using a projected gradient-descent method.

3.6 Structural Risk Optimizing Linear Approaches with Arbitrary Kernel Weights

Lanckriet et al. (2002) follow a direct approach in order to optimize the unrestricted kernel combination weights. The *implausibility* of a kernel matrix, $\omega(\mathbf{K})$, is defined as the objective function value obtained after solving a canonical SVM optimization problem (Here we only consider the soft margin formulation, which uses the ℓ_1-norm on slack variables):

$$\begin{aligned} \text{maximize} \;\; & \omega(\mathbf{K}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{with respect to} \;\; & \alpha \in \mathbb{R}_+^N \\ \text{subject to} \;\; & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \forall i. \end{aligned}$$

The combined kernel matrix is selected from the following set:

$$\mathcal{K}_L = \left\{ \mathbf{K} \colon \mathbf{K} = \sum_{m=1}^{P} \eta_m \mathbf{K}_m, \;\; \mathbf{K} \succeq 0, \;\; \text{tr}(\mathbf{K}) \leq c \right\}$$

where the selected kernel matrix is forced to be positive semidefinite.

The resulting optimization problem that minimizes the implausibility of the combined kernel matrix (the objective function value of the corresponding soft margin SVM optimization problem) is formulated as

$$\begin{aligned}\text{minimize} \quad & \omega(\mathbf{K}_\eta^{\text{tra}}) \\ \text{with respect to} \quad & \mathbf{K}_\eta \in \mathcal{K}_L \\ \text{subject to} \quad & \operatorname{tr}\left(\mathbf{K}_\eta\right) = c\end{aligned}$$

where $\mathbf{K}_\eta^{\text{tra}}$ is the kernel matrix calculated only over the training set and this problem can be cast into the following SDP formulation:

$$\begin{aligned}\text{minimize} \quad & t \\ \text{with respect to} \quad & \eta \in \mathbb{R}^P, \ t \in \mathbb{R}, \ \lambda \in \mathbb{R}, \ \nu \in \mathbb{R}_+^N, \ \delta \in \mathbb{R}_+^N \\ \text{subject to} \quad & \operatorname{tr}\left(\mathbf{K}_\eta\right) = c \\ & \begin{pmatrix} (\mathbf{y}\mathbf{y}^\top) \odot \mathbf{K}_\eta^{\text{tra}} & \mathbf{1} + \nu - \delta + \lambda \mathbf{y} \\ (\mathbf{1} + \nu - \delta + \lambda \mathbf{y})^\top & t - 2C\delta^\top \mathbf{1} \end{pmatrix} \succeq 0 \\ & \mathbf{K}_\eta \succeq 0.\end{aligned}$$

This optimization problem is defined for a transductive learning setting and we need to be able to calculate the kernel function values for the test instances as well as the training instances.

Lanckriet et al. (2004a,c) consider predicting function classifications associated with yeast proteins. Different kernels calculated on heterogeneous genomic data, namely, amino acid sequences, protein-protein interactions, genetic interactions, protein complex data, and expression data, are combined using an SDP formulation. This gives better results than SVMs trained with each kernel in nine out of 13 experiments. Qiu and Lane (2005) extends ε-tube SVR to a QCQP formulation for regression problems. Conforti and Guido (2010) propose another SDP formulation that removes trace restriction on the combined kernel matrix and introduces constraints over the kernel weights for an inductive setting.

3.7 Structural Risk Optimizing Linear Approaches with Nonnegative Kernel Weights

Lanckriet et al. (2004a) restrict the combination weights to have nonnegative values by selecting the combined kernel matrix from

$$\mathcal{K}_P = \left\{ \mathbf{K} \colon \mathbf{K} = \sum_{m=1}^{P} \eta_m \mathbf{K}_m, \ \eta \geq 0, \ \mathbf{K} \succeq 0, \ \operatorname{tr}(\mathbf{K}) \leq c \right\}$$

and reduce the SDP formulation to the following QCQP problem by selecting the combined kernel matrix from $\mathcal{K}_P$ instead of $\mathcal{K}_L$:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}ct - \sum_{i=1}^{N} \alpha_i \\
\text{with respect to} \quad & \alpha \in \mathbb{R}_{+}^{N}, \ t \in \mathbb{R} \\
\text{subject to} \quad & \operatorname{tr}(\mathbf{K}_m)t \geq \alpha^{\top}((\mathbf{y}\mathbf{y}^{\top}) \odot \mathbf{K}_m^{\text{tra}})\alpha \qquad \forall m \\
& \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned}$$

where we can jointly find the support vector coefficients and the kernel combination weights. This optimization problem is also developed for a transductive setting, but we can simply take the number of test instances as zero and find the kernel combination weights for an inductive setting. The interior-point methods used to solve this QCQP formulation also return the optimal values of the dual variables that correspond to the optimal kernel weights. Qiu and Lane (2005) give also a QCQP formulation of regression using ε-tube SVR. The QCQP formulation is used for predicting siRNA efficacy by combining kernels over heterogeneous data sources (Qiu and Lane, 2009). Zhao et al. (2009) develop a multiple kernel learning method for clustering problems using the maximum margin clustering idea of Xu et al. (2005) and a nonnegative linear combination of kernels.

Lanckriet et al. (2004a) combine two different kernels obtained from heterogeneous information sources, namely, bag-of-words and graphical representations, on the Reuters-21578 data set. Combining these two kernels with positive weights outperforms the single-kernel results obtained with SVM on four tasks out of five. Lanckriet et al. (2004b) use a QCQP formulation to integrate multiple kernel functions calculated on heterogeneous views of the genome data obtained through different experimental procedures. These views include amino acid sequences, hydropathy profiles, gene expression data and known protein-protein interactions. The prediction task is to recognize the particular classes of proteins, namely, membrane proteins and ribosomal proteins. The QCQP approach gives significantly better results than any single kernel and the unweighted sum of kernels. The assigned kernel weights also enable us to extract the relative importance of the data sources feeding the separate kernels. This approach assigns near zero weights to random kernels added to the candidate set of kernels before training. Dehak et al. (2008) combine three different kernels obtained on the same features and get better results than score fusion for speaker verification problem.

A similar result about unweighted and weighted linear kernel combinations is also obtained by Lewis et al. (2006a). They compare the performances of unweighted and weighted sums of kernels on a gene functional classification task. Their results can be summarized with two guidelines: (a) When all kernels or data sources are informative, we should use the unweighted sum rule. (b) When some of the kernels or the data sources are noisy or irrelevant, we should optimize the kernel weights.

Fung et al. (2004) propose an iterative algorithm using the kernel Fisher discriminant analysis as the base learner to combine heterogeneous kernels in a linear manner with nonnegative weights. The proposed method requires solving a simple nonsingular system of linear equations of size $(N+1)$ and a QP problem having P decision variables at each iteration. On a colorectal cancer diagnosis task, this method obtains similar results using much less computation time compared to selecting a kernel for standard kernel Fisher discriminant analysis.

Tsuda et al. (2004) learn the kernel combination weights by minimizing an approximation of the cross-validation error for kernel Fisher discriminant analysis. In order to update the kernel combination weights, cross-validation error should be approximated with a differentiable error function. They use the sigmoid function for error approximation and derive the update rules of the kernel weights. This procedure requires inverting a $N \times N$ matrix and calculating the gradients at each step. They combine heterogeneous data sources using kernels, which are mixed linearly and non-linearly, for bacteria classification and gene function prediction tasks. Fisher discriminant analysis with the combined kernel matrix that is optimized using the cross-validation error approximation, gives significantly better results than single kernels for both tasks.

In order to consider the capacity of the resulting classifier, Tan and Wang (2004) optimize the nonnegative combination coefficients using the minimal upper bound of the Vapnik-Chervonenkis dimension as the target function.

Varma and Ray (2007) propose a formulation for combining kernels using a linear combination with regularized nonnegative weights. The regularization on the kernel combination weights is achieved by adding a term to the objective function and integrating a set of constraints. The primal optimization problem with these two modifications can be given as

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\|\mathbf{w}_\eta\|_2^2 + C\sum_{i=1}^{N}\xi_i + \sum_{m=1}^{P}\sigma_m\eta_m \\
\text{with respect to} \quad & \mathbf{w}_\eta \in \mathbb{R}^{S_\eta},\ \ \xi \in \mathbb{R}_+^N,\ \ b \in \mathbb{R},\ \ \eta \in \mathbb{R}_+^P \\
\text{subject to} \quad & y_i(\langle \mathbf{w}_\eta, \Phi_\eta(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i \qquad \forall i \\
& \mathbf{A}\eta \geq \mathbf{p}
\end{aligned}$$

where $\Phi_\eta(\cdot)$ corresponds to the feature space that implicitly constructs the combined kernel function $k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P}\eta_m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)$ and $\mathbf{w}_\eta$ is the vector of weight coefficients assigned to $\Phi_\eta(\cdot)$. The parameters $\mathbf{A} \in \mathbb{R}^{R\times P}$, $\mathbf{p} \in \mathbb{R}^R$, and $\sigma \in \mathbb{R}^P$ encode our prior information about the kernel weights. For example, assigning higher σ_i values to some of the kernels effectively eliminates them by assigning zero weights to them. The corresponding dual formulation is derived as the following SOCP problem:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N}\alpha_i - \mathbf{p}^\top\delta \\
\text{with respect to} \quad & \alpha \in \mathbb{R}_+^N,\ \ \delta \in \mathbb{R}_+^P \\
\text{subject to} \quad & \sigma_m - \delta^\top\mathbf{A}(:,k) \geq \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \qquad \forall m \\
& \sum_{i=1}^{N}\alpha_i y_i = 0 \qquad \forall m \\
& C \geq \alpha_i \geq 0 \qquad \forall i.
\end{aligned}$$

Instead of solving this SOCP problem directly, Varma and Ray (2007) also propose an alternating optimization problem that performs projected gradient updates for kernel weights and solves a QP problem to find the support vector coefficients at each iteration. The primal optimization problem for given η is written as

$$\begin{aligned} \text{minimize} \;\; & J(\eta) = \frac{1}{2}\|\mathbf{w}_\eta\|_2^2 + C\sum_{i=1}^{N}\xi_i + \sum_{m=1}^{P}\sigma_m\eta_m \\ \text{with respect to} \;\; & \mathbf{w}_\eta \in \mathbb{R}^{S_\eta}, \;\; \xi \in \mathbb{R}_+^N, \;\; b \in \mathbb{R} \\ \text{subject to} \;\; & y_i(\langle \mathbf{w}_\eta, \Phi_\eta(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i \quad \forall i \end{aligned}$$

and the corresponding dual optimization problem is

$$\begin{aligned} \text{maximize} \;\; & J(\eta) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \underbrace{\left(\sum_{m=1}^{P}\eta_m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)\right)}_{k_\eta(\mathbf{x}_i, \mathbf{x}_j)} + \sum_{m=1}^{P}\sigma_m\eta_m \\ \text{with respect to} \;\; & \alpha \in \mathbb{R}_+^N \\ \text{subject to} \;\; & \sum_{i=1}^{N}\alpha_i y_i = 0 \quad \forall m \\ & C \geq \alpha_i \geq 0 \quad \forall i. \end{aligned}$$

The gradients with respect to the kernel weights are calculated as

$$\frac{\partial J(\eta)}{\partial \eta_m} = \sigma_m - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \frac{\partial k_\eta(\mathbf{x}_i, \mathbf{x}_j)}{\partial \eta_m} = \sigma_m - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \quad \forall m$$

and these gradients are used to update the kernel weights while considering nonnegativity and other constraints.

Usually, the kernel weights are constrained by a trace or the ℓ_1-norm regularization. Cortes et al. (2009) discuss the suitability of the ℓ_2-norm for MKL. They combine kernels with ridge regression using the ℓ_2-norm regularization over the kernel weights. They conclude that using the ℓ_1-norm improves the performance for a small number of kernels, but degrades the performance when combining a large number of kernels. However, the ℓ_2-norm never decreases the performance and increases it significantly for larger sets of candidate kernels. Yan et al. (2009) compare the ℓ_1-norm and the ℓ_2-norm for image and video classification tasks, and conclude that the ℓ_2-norm should be used when the combined kernels carry complementary information.

Kloft et al. (2010a) generalize the MKL formulation for arbitrary ℓ_p-norms with $p \geq 1$ by regularizing over the kernel coefficients (done by adding $\mu\|\eta\|_p^p$ to the objective function) or equivalently, constraining them ($\|\eta\|_p^p \leq 1$). The resulting optimization problem is

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \left(\sum_{m=1}^{P} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \right)^{\frac{p-1}{p}} \right)^{\frac{p}{p-1}} \\
\text{with respect to} \quad & \alpha \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \quad \forall i
\end{aligned}$$

and they solve this problem using alternative optimization strategies based on Newton-descent and cutting planes. Xu et al. (2010b) add an entropy regularization term instead of constraining the norm of the kernel weights and derive an efficient and smooth optimization framework based on Nesterov's method.

Kloft et al. (2010b) and Xu et al. (2010a) propose an efficient optimization method for arbitrary ℓ_p-norms with $p \geq 1$. Although they approach the problem from different perspectives, they find the same closed-form solution for updating the kernel weights at each iteration. Kloft et al. (2010b) use a block coordinate-descent method and Xu et al. (2010a) use the equivalence between group Lasso and MKL, as shown by Bach (2008) to derive the update equation. Both studies formulate an alternating optimization method that solves an SVM at each iteration and update the kernel weights as follows:

$$\eta_m = \frac{\|\mathbf{w}_m\|_2^{\frac{2}{p+1}}}{\left(\sum\limits_{h=1}^{P} \|\mathbf{w}_h\|_2^{\frac{2p}{p+1}} \right)^{\frac{1}{p}}} \tag{8}$$

where $\|\mathbf{w}_m\|_2^2 = \eta_m^2 \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)$ from the duality conditions.

When we restrict the kernel weights to be nonnegative, the SDP formulation of Conforti and Guido (2010) reduces to a QCQP problem.

Lin et al. (2009) propose a dimensionality reduction method that uses multiple kernels to embed data instances from different feature spaces to a unified feature space. The method is derived from a graph embedding framework using kernel matrices instead of data matrices. The learning phase is performed using a two-step alternate optimization procedure that updates the dimensionality reduction coefficients and the kernel weights in turn. McFee and Lanckriet (2009) propose a method for learning a unified space from multiple kernels calculated over heterogeneous data sources. This method uses a partial order over pairwise distances as the input and produces an embedding using graph-theoretic tools. The kernel (data source) combination rule is learned by solving an SDP problem and all input instances are mapped to the constructed common embedding space.

Another possibility is to allow only binary η_m for kernel selection. We get rid of kernels whose $\eta_m = 0$ and use the kernels whose $\eta_m = 1$. Xu et al. (2009b) define a combined kernel over the set of kernels calculated on each feature independently and perform feature selection using this definition.

The defined kernel function can be expressed as

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{D} \eta_m k(\mathbf{x}_i[m], \mathbf{x}_j[m])$$

where $[\cdot]$ indexes the elements of a vector and $\eta \in \{0,1\}^D$. For efficient learning, η is relaxed into the continuous domain (i.e., $1 \geq \eta \geq 0$). Following Lanckriet et al. (2004a), an SDP formulation is derived and this formulation is cast into a QCQP problem to reduce the time complexity.

3.8 Structural Risk Optimizing Linear Approaches with Kernel Weights on a Simplex

We can think of kernel combination as a weighted average of kernels and consider $\eta \in \mathbb{R}_+^P$ and $\sum_{m=1}^{P} \eta_m = 1$. Joachims et al. (2001) show that combining two kernels is beneficial if both of them achieve approximately the same performance and use different data instances as support vectors. This makes sense because in combination, we want kernels to be useful by themselves and complementary. In a web page classification experiment, they show that combining the word and the hyperlink representations through the convex combination of two kernels (i.e., $\eta_2 = 1 - \eta_1$) can achieve better classification accuracy than each of the kernels.

Chapelle et al. (2002) calculate the derivative of the margin and the derivative of the radius (of the smallest sphere enclosing the training points) with respect to a kernel parameter, θ:

$$\frac{\partial \|\mathbf{w}\|_2^2}{\partial \theta} = -\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta}$$
$$\frac{\partial R^2}{\partial \theta} = \sum_{i=1}^{N} \beta_i \frac{\partial k(\mathbf{x}_i, \mathbf{x}_i)}{\partial \theta} - \sum_{i=1}^{N}\sum_{j=1}^{N} \beta_i \beta_j \frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta}$$

where α is obtained by solving the canonical SVM optimization problem and β is obtained by solving the QP problem defined by Vapnik (1998). These derivatives can be used to optimize the individual parameters (e.g., scaling coefficient) on each feature using an alternating optimization procedure (Weston et al., 2001; Chapelle et al., 2002; Grandvalet and Canu, 2003). This strategy is also a multiple kernel learning approach, because the optimized parameters can be interpreted as the kernel parameters and we combine these kernel values over all features.

Bousquet and Herrmann (2003) rewrite the gradient of the margin by replacing $\mathbf{K}$ with $\mathbf{K}_\eta$ and taking the derivative with respect to the kernel weights gives

$$\frac{\partial \|\mathbf{w}_\eta\|_2^2}{\partial \eta_m} = -\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \frac{\partial k_\eta(\mathbf{x}_i, \mathbf{x}_j)}{\partial \eta_m} = -\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \quad \forall m$$

where $\mathbf{w}_\eta$ is the weight vector obtained using $\mathbf{K}_\eta$ in training. In an iterative manner, an SVM is trained to obtain α, then η is updated using the calculated gradient while considering nonnegativity (i.e., $\eta \in \mathbb{R}_+^P$) and normalization (i.e., $\sum_{m=1}^{P} \eta_m = 1$). This procedure considers the performance (in terms of margin maximization) of the resulting classifier, which uses the combined kernel matrix.

Bach et al. (2004) propose a modified primal formulation that uses the weighted ℓ_1-norm on feature spaces and the ℓ_2-norm within each feature space. The modified primal formulation is

$$\begin{aligned} \text{minimize} \ & \frac{1}{2}\left(\sum_{m=1}^{P} d_m \|\mathbf{w}_m\|_2\right)^2 + C\sum_{i=1}^{N} \xi_i \\ \text{with respect to} \ & \mathbf{w}_m \in \mathbb{R}^{S_m},\ \ \xi \in \mathbb{R}_+^N,\ \ b \in \mathbb{R} \\ \text{subject to} \ & y_i\left(\sum_{m=1}^{P} \langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m)\rangle + b\right) \geq 1 - \xi_i \quad \forall i \end{aligned}$$

where the feature space constructed using $\Phi_m(\cdot)$ has the dimensionality S_m and the weight d_m. When we consider this optimization problem as an SOCP problem, we obtain the following dual formulation:

$$\begin{aligned} \text{minimize} \ & \frac{1}{2}\gamma^2 - \sum_{i=1}^{N} \alpha_i \\ \text{with respect to} \ & \gamma \in \mathbb{R},\ \ \alpha \in \mathbb{R}_+^N \\ \text{subject to} \ & \gamma^2 d_m^2 \geq \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \quad \forall m \\ & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \forall i \end{aligned} \tag{9}$$

where we again get the optimal kernel weights from the optimal dual variables and the weights satisfy $\sum_{m=1}^{P} d_m^2 \eta_m = 1$. The dual problem is exactly equivalent to the QCQP formulation of Lanckriet et al. (2004a) when we take $d_m = \sqrt{\text{tr}(\mathbf{K}_m)/c}$. The advantage of the SOCP formulation is that Bach et al. (2004) devise an SMO-like algorithm by adding a Moreau-Yosida regularization term, $1/2\sum_{m=1}^{P} a_m^2 \|\mathbf{w}_m\|_2^2$, to the primal objective function and deriving the corresponding dual formulation. Using the ℓ_1-norm on feature spaces, Yamanishi et al. (2007) combine tree kernels for identifying human glycans into four blood components: leukemia cells, erythrocytes, plasma, and serum. Except on plasma task, representing glycans as rooted trees and combining kernels improve performance in terms of the area under the ROC curve. Özen et al. (2009) use the formulation of Bach et al. (2004) to combine different feature subsets for protein stability prediction problem and extract information about the importance of these subsets by looking at the learned kernel weights.

Bach (2009) develops a method for learning linear combinations of an exponential number of kernels, which can be expressed as product of sums. The method is applied to nonlinear variable selection and efficiently explores the large feature spaces in polynomial time.

Sonnenburg et al. (2006a,b) rewrite the QCQP formulation of Bach et al. (2004):

$$\begin{aligned}
\text{minimize} \quad & \gamma \\
\text{with respect to} \quad & \gamma \in \mathbb{R}, \ \ \alpha \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \quad \forall i \\
& \gamma \geq \underbrace{\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) - \sum_{i=1}^{N} \alpha_i}_{S_m(\alpha)} \quad \forall m
\end{aligned}$$

and convert this problem into the following SILP problem:

$$\begin{aligned}
\text{maximize} \quad & \theta \\
\text{with respect to} \quad & \theta \in \mathbb{R}, \ \ \eta \in \mathbb{R}_+^P \\
\text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1 \\
& \sum_{m=1}^{P} \eta_m S_m(\alpha) \geq \theta \quad \forall \alpha \in \{\alpha \colon \alpha \in \mathbb{R}^N, \ \alpha^\top \mathbf{y} = 0, \ C \geq \alpha \geq 0\}
\end{aligned}$$

where the problem has infinitely many constraints due to the possible values of α.

The SILP formulation has lower computational complexity compared to the SDP and QCQP formulations. Sonnenburg et al. (2006a,b) use a column generation approach to solve the resulting SILPs using a generic LP solver and a canonical SVM solver in the inner loop. Both the LP solver and the SVM solver can use the previous optimal values for hot-start to obtain the new optimal values faster. These allow us to use the SILP formulation to learn the kernel combination weights for hundreds of kernels on hundreds of thousands of training instances efficiently. For example, they perform training on a real-world splice data set with millions of instances from computational biology with string kernels. They also generalize the idea to regression, one-class classification, and strictly convex and differentiable loss functions.

Kim et al. (2006) show that selecting the optimal kernel from the set of convex combinations over the candidate kernels can be formulated as a convex optimization problem. This formulation is more efficient than the iterative approach of Fung et al. (2004). Ye et al. (2007a) formulate an SDP problem inspired by Kim et al. (2006) for learning an optimal kernel over a convex set of candidate kernels for RKDA. The SDP formulation can be modified so that it can jointly optimize the kernel weights and the regularization parameter. Ye et al. (2007b, 2008) derive QCQP and SILP formulations equivalent to the previous SDP problem in order to reduce the time complexity. These three formulations are directly applicable to multiclass classification because it uses RKDA as the base learner.

De Bie et al. (2007) derive a QCQP formulation of one-class classification using a convex combination of multiple kernels. In order to prevent the combined kernel from overfitting, they also propose a modified mathematical model that defines lower limits for the kernel weights. Hence, each kernel in the set of candidate kernels is used in the combined kernel and we obtain a more regularized solution.

Zien and Ong (2007) develop a QCQP formulation and convert this formulation in two different SILP problems for multiclass classification. They show that their formulation is the multiclass generalization of the previously developed binary classification methods of Bach et al. (2004) and Sonnenburg et al. (2006b). The proposed multiclass formulation is tested on different bioinformatics applications such as bacterial protein location prediction (Zien and Ong, 2007) and protein subcellular location prediction (Zien and Ong, 2007, 2008), and outperforms individual kernels and unweighted sum of kernels. Hu et al. (2009) combine the MKL formulation of Zien and Ong (2007) and the sparse kernel learning method of Wu et al. (2006). This hybrid approach learns the optimal kernel weights and also obtains a sparse solution.

Rakotomamonjy et al. (2007, 2008) propose a different primal problem for MKL and use a projected gradient method to solve this optimization problem. The proposed primal formulation is

$$\begin{aligned}
\text{minimize } & \frac{1}{2}\sum_{m=1}^{P}\frac{1}{\eta_m}\|\mathbf{w}_m\|_2^2 + C\sum_{i=1}^{N}\xi_i \\
\text{with respect to } & \mathbf{w}_m \in \mathbb{R}^{S_m},\ \xi \in \mathbb{R}_+^N,\ b \in \mathbb{R},\ \eta \in \mathbb{R}_+^P \\
\text{subject to } & y_i\left(\sum_{m=1}^{P}\langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m)\rangle + b\right) \geq 1 - \xi_i \quad \forall i \\
& \sum_{m=1}^{P}\eta_m = 1
\end{aligned}$$

and they define the optimal SVM objective function value given η as $J(\eta)$:

$$\begin{aligned}
\text{minimize } & J(\eta) = \frac{1}{2}\sum_{m=1}^{P}\frac{1}{\eta_m}\|\mathbf{w}_m\|_2^2 + C\sum_{i=1}^{N}\xi_i \\
\text{with respect to } & \mathbf{w}_m \in \mathbb{R}^{S_m},\ \xi \in \mathbb{R}_+^N,\ b \in \mathbb{R} \\
\text{subject to } & y_i\left(\sum_{m=1}^{P}\langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m)\rangle + b\right) \geq 1 - \xi_i \quad \forall i.
\end{aligned}$$

Due to strong duality, one can also calculate $J(\eta)$ using the dual formulation:

$$\begin{aligned}
\text{maximize } & J(\eta) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \underbrace{\left(\sum_{m=1}^{P}\eta_m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m)\right)}_{k_\eta(\mathbf{x}_i, \mathbf{x}_j)} \\
\text{with respect to } & \alpha \in \mathbb{R}_+^N \\
\text{subject to } & \sum_{i=1}^{N}\alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \quad \forall i.
\end{aligned}$$

The primal formulation can be seen as the following constrained optimization problem:

$$\begin{aligned} \text{minimize } & J(\eta) \\ \text{with respect to } & \eta \in \mathbb{R}_+^P \\ \text{subject to } & \sum_{m=1}^{P} \eta_m = 1. \end{aligned} \tag{10}$$

The overall procedure to solve this problem, called SIMPLEMKL, consists of two main steps: (a) solving a canonical SVM optimization problem with given η and (b) updating η using the following gradient calculated with α found in the first step:

$$\frac{\partial J(\eta)}{\partial \eta_m} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \frac{\partial k_\eta(\mathbf{x}_i^m, \mathbf{x}_j^m)}{\partial \eta_m} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \quad \forall m.$$

The gradient update procedure must consider the nonnegativity and normalization properties of the kernel weights. The derivative with respect to the kernel weights is exactly equivalent (up to a multiplicative constant) to the gradient of the margin calculated by Bousquet and Herrmann (2003). The overall algorithm is very similar to the algorithm used by Sonnenburg et al. (2006a,b) to solve an SILP formulation. Both algorithms use a canonical SVM solver in order to calculate α at each step. The difference is that they use different updating procedures for η, namely, a projected gradient update and solving an LP. Rakotomamonjy et al. (2007, 2008) show that SIMPLEMKL is more stable than solving the SILP formulation. SIMPLEMKL can be generalized to regression, one-class and multiclass classification (Rakotomamonjy et al., 2008).

Chapelle and Rakotomamonjy (2008) propose a second order method, called HESSIANMKL, extending SIMPLEMKL. HESSIANMKL updates kernel weights at each iteration using a constrained Newton step found by solving a QP problem. Chapelle and Rakotomamonjy (2008) show that HESSIANMKL converges faster than SIMPLEMKL.

Xu et al. (2009a) propose a hybrid method that combines the SILP formulation of Sonnenburg et al. (2006b) and SIMPLEMKL of Rakotomamonjy et al. (2008). The SILP formulation does not regularize the kernel weights obtained from the cutting plane method and SIMPLEMKL uses the gradient calculated only in the last iteration. The proposed model overcomes both disadvantages and finds the kernel weights for the next iteration by solving a small QP problem; this regularizes the solution and uses the past information.

The alternating optimization method proposed by Kloft et al. (2010b) and Xu et al. (2010a) learns a convex combination of kernels when we use the ℓ_1-norm for regularizing the kernel weights. When we take $p = 1$, the update equation in (8) becomes

$$\eta_m = \frac{\|\mathbf{w}_m\|_2}{\sum_{h=1}^{P} \|\mathbf{w}_h\|_2}. \tag{11}$$

The SDP formulation of Conforti and Guido (2010) reduces to a QCQP problem when we use a convex combination of the base kernels.

Longworth and Gales (2008, 2009) introduce an extra regularization term to the objective function of SIMPLEMKL (Rakotomamonjy et al., 2008). This modification allows changing the level of sparsity of the combined kernels. The extra regularization term is

$$\lambda \sum_{m=1}^{P} \left(\eta_m - \frac{1}{P} \right)^2 = \lambda \sum_{m=1}^{P} \eta_m^2 - \frac{\lambda}{P} =^{+} \lambda \sum_{m=1}^{P} \eta_m^2$$

where λ is regularization parameter that determines the solution sparsity. For example, large values of λ force the mathematical model to use all the kernels with a uniform weight, whereas small values produce sparse combinations.

Micchelli and Pontil (2005) try to learn the optimal kernel over the convex hull of predefined basic kernels by minimizing a regularization functional. Their analysis shows that any optimizing kernel can be expressed as the convex combination of basic kernels. Argyriou et al. (2005, 2006) build practical algorithms for learning a suboptimal kernel when the basic kernels are continuously parameterized by a compact set. This continuous parameterization allows selecting kernels from basically an infinite set, instead of a finite number of basic kernels.

Instead of selecting kernels from a predefined finite set, we can increase the number of candidate kernels in an iterative manner. We can basically select kernels from an uncountably infinite set constructed by considering base kernels with different kernel parameters (Özöğür-Akyüz and Weber, 2008; Gehler and Nowozin, 2008). Gehler and Nowozin (2008) propose a forward selection algorithm that finds the kernel weights for a fixed size of candidate kernels using one of the methods described above, then adds a new kernel to the set of candidate kernels, until convergence.

Most MKL methods do not consider the group structure between the kernels combined. For example, a group of kernels may be calculated on the same set of features and even if we assign a nonzero weight to only one of them, we have to extract the features in the testing phase. When kernels have such a group structure, it is reasonable to pick all or none of them in the combined kernel. Szafranski et al. (2008, 2010) follow this idea and derive an MKL method by changing the mathematical model used by Rakotomamonjy et al. (2007). Saketha Nath et al. (2010) propose another MKL method that considers the group structure between the kernels and this method assumes that every kernel group carries important information. The proposed formulation enforces the ℓ_∞-norm at the group level and the ℓ_1-norm within each group. By doing this, each group is used in the final learner, but sparsity is promoted among kernels in each group. They formulate the problem as an SCOP problem and give a highly efficient optimization algorithm that uses a mirror-descent approach.

Subrahmanya and Shin (2010) generalize group-feature selection to kernel selection by introducing a log-based concave penalty term for obtaining extra sparsity; this is called sparse multiple kernel learning (SMKL). The reason for adding this concave penalty term is explained as the lack of ability of convex MKL methods to obtain sparse formulations. They show that SMKL obtains more sparse solutions than convex formulations for signal processing applications.

Most of the structural risk optimizing linear approaches can be casted into a general framework (Kloft et al., 2010a,b). The unified optimization problem with the Tikhonov regularization can be written as

$$\text{minimize } \frac{1}{2} \sum_{m=1}^{P} \frac{\|\mathbf{w}_m\|_2^2}{\eta_m} + C \sum_{i=1}^{N} L \left(\sum_{m=1}^{P} \langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m) \rangle + b, y_i \right) + \mu \|\eta\|_p^p$$

$$\text{with respect to } \mathbf{w}_m \in \mathbb{R}^{S_m},\ b \in \mathbb{R},\ \eta \in \mathbb{R}_+^P$$

where $L(\cdot,\cdot)$ is the loss function used. Alternatively, we can use the Ivanov regularization instead of the Tikhonov regularization by integrating an additional constraint into the optimization problem:

$$\begin{aligned} \text{minimize } & \frac{1}{2}\sum_{m=1}^{P}\frac{\|\mathbf{w}_m\|_2^2}{\eta_m} + C\sum_{i=1}^{N} L\left(\sum_{m=1}^{P}\langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m)\rangle + b, y_i\right) \\ \text{with respect to } & \mathbf{w}_m \in \mathbb{R}^{S_m},\ b \in \mathbb{R},\ \eta \in \mathbb{R}_+^P \\ \text{subject to } & \|\eta\|_p^p \leq 1. \end{aligned}$$

Figure 1 lists the MKL algorithms that can be casted into the general framework described above. Zien and Ong (2007) show that their formulation is equivalent to those of Bach et al. (2004) and Sonnenburg et al. (2006a,b). Using unified optimization problems given above and the results of Zien and Ong (2007), Kloft et al. (2010a,b) show that the formulations with $p = 1$ in Figure 1 fall into the same equivalence class and introduce a new formulation with $p \geq 1$. The formulation of Xu et al. (2010a) is also equivalent to those of Kloft et al. (2010a,b).

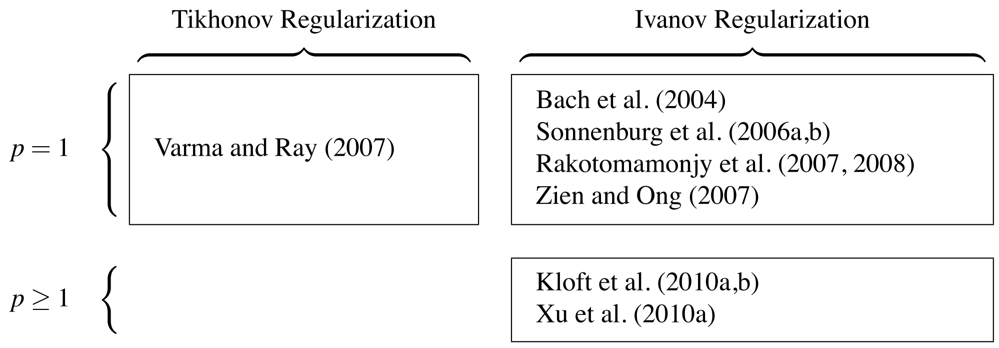


Figure 1: MKL algorithms that can be casted into the general framework described.

3.9 Structural Risk Optimizing Nonlinear Approaches

Ong et al. (2003) propose to learn a kernel function instead of a kernel matrix. They define a kernel function in the space of kernels called a *hyperkernel*. Their construction includes convex combinations of an infinite number of pointwise nonnegative kernels. Hyperkernels are generalized to different machine learning problems such as binary classification, regression, and one-class classification (Ong and Smola, 2003; Ong et al., 2005). When they use the regularized risk functional as the empirical quality functional to be optimized, the learning phase can be performed by solving an SDP problem. Tsang and Kwok (2006) convert the resulting optimization problems into SOCP problems in order to reduce the time complexity of the training phase.

Varma and Babu (2009) propose a generalized formulation called generalized multiple kernel learning (GMKL) that contains two regularization terms and a loss function in the objective function. This formulation regularizes both the hyperplane weights and the kernel combination weights. The loss function can be one of the classical loss functions, such as, hinge loss for classification, or ε-loss for regression. The proposed primal formulation applied to binary classification problem with hinge loss and the regularization function, $r(\cdot)$, can be written as

$$\begin{aligned} \text{minimize} \ & \frac{1}{2}\|\mathbf{w}_\eta\|_2^2 + C\sum_{i=1}^N \xi_i + r(\eta) \\ \text{with respect to} \ & \mathbf{w}_\eta \in \mathbb{R}^{S_\eta}, \ \xi \in \mathbb{R}_+^N, \ b \in \mathbb{R}, \ \eta \in \mathbb{R}_+^P \\ \text{subject to} \ & y_i(\langle \mathbf{w}_\eta, \Phi_\eta(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i \quad \forall i \end{aligned}$$

where $\Phi_\eta(\cdot)$ corresponds to the feature space that implicitly constructs the combined kernel function $k_\eta(\cdot,\cdot)$ and $\mathbf{w}_\eta$ is the vector of weight coefficients assigned to $\Phi_\eta(\cdot)$. This problem, different from the primal problem of SIMPLEMKL, is not convex, but the solution strategy is the same. The objective function value of the primal formulation given η is used as the target function:

$$\begin{aligned} \text{minimize} \ & J(\eta) = \frac{1}{2}\|\mathbf{w}_\eta\|_2^2 + C\sum_{i=1}^N \xi_i + r(\eta) \\ \text{with respect to} \ & \mathbf{w}_\eta \in \mathbb{R}^{S_\eta}, \ \xi \in \mathbb{R}_+^N, \ b \in \mathbb{R} \\ \text{subject to} \ & y_i(\langle \mathbf{w}_\eta, \Phi_\eta(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i \quad \forall i \end{aligned}$$

and the following dual formulation is used for the gradient step:

$$\begin{aligned} \text{maximize} \ & J(\eta) = \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j k_\eta(\mathbf{x}_i,\mathbf{x}_j) + r(\eta) \\ \text{with respect to} \ & \alpha \in \mathbb{R}_+^N \\ \text{subject to} \ & \sum_{i=1}^N \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \forall i. \end{aligned}$$

The regularization function $r(\cdot)$ and $k_\eta(\cdot,\cdot)$ can be any differentiable function of η with continuous derivative. The gradient with respect to the kernel weights is calculated as

$$\frac{\partial J(\eta)}{\partial \eta_m} = \frac{\partial r(\eta)}{\partial \eta_m} - \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j \frac{\partial k_\eta(\mathbf{x}_i,\mathbf{x}_j)}{\partial \eta_m} \quad \forall m.$$

Varma and Babu (2009) perform gender identification experiments on a face image data set by combining kernels calculated on each individual feature, and hence, for kernels whose η_m goes to 0, they perform feature selection. SIMPLEMKL and GMKL are trained with the kernel functions $k_\eta^S(\cdot,\cdot)$ and $k_\eta^P(\cdot,\cdot)$, respectively:

$$\begin{aligned} k_\eta^S(\mathbf{x}_i,\mathbf{x}_j) &= \sum_{m=1}^D \eta_m \exp\left(-\gamma_m\left(\mathbf{x}_i[m] - \mathbf{x}_j[m]\right)^2\right) \\ k_\eta^P(\mathbf{x}_i,\mathbf{x}_j) &= \prod_{m=1}^D \exp\left(-\eta_m\left(\mathbf{x}_i[m] - \mathbf{x}_j[m]\right)^2\right) = \exp\left(\sum_{m=1}^D -\eta_m\left(\mathbf{x}_i[m] - \mathbf{x}_j[m]\right)^2\right). \end{aligned}$$

They show that GMKL with $k_\eta^P(\cdot,\cdot)$ performs significantly better than SIMPLEMKL with $k_\eta^S(\cdot,\cdot)$. We see that using $k_\eta^P(\cdot,\cdot)$ as the combined kernel function is equivalent to using different scaling parameters on each feature and using an RBF kernel over these scaled features with unit radius, as done by Grandvalet and Canu (2003).

Cortes et al. (2010b) develop a nonlinear kernel combination method based on KRR and polynomial combination of kernels. They propose to combine kernels as follows:

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\mathbf{q} \in \mathcal{Q}} \eta_{q_1 q_2 \dots q_P} k_1(\mathbf{x}_i^1, \mathbf{x}_j^1)^{q_1} k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)^{q_2} \dots k_P(\mathbf{x}_i^P, \mathbf{x}_j^P)^{q_P}$$

where $\mathcal{Q} = \{\mathbf{q} \colon \mathbf{q} \in \mathbb{Z}_+^P, \ \sum_{m=1}^P q_m \leq d\}$ and $\eta_{q_1 q_2 \dots q_P} \geq 0$. The number of parameters to be learned is too large and the combined kernel is simplified in order to reduce the learning complexity:

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\mathbf{q} \in \mathcal{R}} \eta_1^{q_1} \eta_2^{q_2} \dots \eta_P^{q_P} k_1(\mathbf{x}_i^1, \mathbf{x}_j^1)^{q_1} k_2(\mathbf{x}_i^2, \mathbf{x}_j^2)^{q_2} \dots k_P(\mathbf{x}_i^P, \mathbf{x}_j^P)^{q_P}$$

where $\mathcal{R} = \{\mathbf{q} \colon \mathbf{q} \in \mathbb{Z}_+^P, \ \sum_{m=1}^P q_m = d\}$ and $\eta \in \mathbb{R}^P$. For example, when $d = 2$, the combined kernel function becomes

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^P \sum_{h=1}^P \eta_m \eta_h k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) k_h(\mathbf{x}_i^h, \mathbf{x}_j^h). \tag{12}$$

The combination weights are optimized using the following min-max optimization problem:

$$\underset{\eta \in \mathcal{M}}{\text{minimize}} \ \underset{\alpha \in \mathbb{R}^N}{\text{maximize}} \ -\alpha^\top (\mathbf{K}_\eta + \lambda \mathbf{I}) \alpha + 2 \mathbf{y}^\top \alpha$$

where $\mathcal{M}$ is a positive, bounded, and convex set. Two possible choices for the set $\mathcal{M}$ are the ℓ_1-norm and ℓ_2-norm bounded sets defined as

$$\mathcal{M}_1 = \{\eta \colon \eta \in \mathbb{R}_+^P, \ \|\eta - \eta_0\|_1 \leq \Lambda\} \tag{13}$$

$$\mathcal{M}_2 = \{\eta \colon \eta \in \mathbb{R}_+^P, \ \|\eta - \eta_0\|_2 \leq \Lambda\} \tag{14}$$

where η_0 and Λ are two model parameters. A projection-based gradient-descent algorithm can be used to solve this min-max optimization problem. At each iteration, α is obtained by solving a KRR problem with the current kernel matrix and η is updated with the gradients calculated using α while considering the bound constraints on η due to $\mathcal{M}_1$ or $\mathcal{M}_2$.

Lee et al. (2007) follow a different approach and combine kernels using a compositional method that constructs a $(P \times N) \times (P \times N)$ compositional kernel matrix. This matrix and the training instances replicated P times are used to train a canonical SVM.

3.10 Structural Risk Optimizing Data-Dependent Approaches

Lewis et al. (2006b) use a latent variable generative model using the maximum entropy discrimination to learn data-dependent kernel combination weights. This method combines a generative probabilistic model with a discriminative large margin method.

Gönen and Alpaydın (2008) propose a data-dependent formulation called localized multiple kernel learning (LMKL) that combines kernels using weights calculated from a gating model. The proposed primal optimization problem is

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\sum_{m=1}^{P} \|\mathbf{w}_m\|_2^2 + C\sum_{i=1}^{N} \xi_i \\
\text{with respect to} \quad & \mathbf{w}_m \in \mathbb{R}^{S_m},\ \xi \in \mathbb{R}_+^N,\ b \in \mathbb{R},\ \mathbf{V} \in \mathbb{R}^{P \times (D_{\mathcal{G}}+1)} \\
\text{subject to} \quad & y_i \left(\sum_{m=1}^{P} \eta_m(\mathbf{x}_i|\mathbf{V}) \langle \mathbf{w}_m, \Phi_m(\mathbf{x}_i^m) \rangle + b \right) \geq 1 - \xi_i \qquad \forall i
\end{aligned}$$

where the gating model $\eta_m(\cdot|\cdot)$, parameterized by $\mathbf{V}$, assigns a weight to the feature space obtained with $\Phi_m(\cdot)$. This optimization problem is not convex and a two-step alternate optimization procedure is used to find the classifier parameters and the gating model parameters. When we fix the gating model parameters, the problem becomes convex and we obtain the following dual problem:

$$\begin{aligned}
\text{maximize} \quad & J(\mathbf{V}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_{\eta}(\mathbf{x}_i, \mathbf{x}_j) \\
\text{with respect to} \quad & \alpha \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned}$$

where the combined kernel matrix is represented as

$$k_{\eta}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P} \eta_m(\mathbf{x}_i|\mathbf{V}) k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \eta_m(\mathbf{x}_j|\mathbf{V}).$$

Assuming that the regions of expertise of kernels are linearly separable, we can express the gating model using softmax function:

$$\eta_m(\mathbf{x}|\mathbf{V}) = \frac{\exp(\langle \mathbf{v}_m, \mathbf{x}^{\mathcal{G}} \rangle + v_{m0})}{\sum_{h=1}^{P} \exp(\langle \mathbf{v}_h, \mathbf{x}^{\mathcal{G}} \rangle + v_{h0})} \qquad \forall m \tag{15}$$

where $\mathbf{V} = \{\mathbf{v}_m, v_{m0}\}_{m=1}^{P}$, $\mathbf{x}^{\mathcal{G}} \in \mathbb{R}^{D_{\mathcal{G}}}$ is the representation of the input instance in the feature space in which we learn the gating model and there are $P \times (D_{\mathcal{G}}+1)$ parameters where $D_{\mathcal{G}}$ is the dimensionality of the gating feature space. The softmax gating model uses kernels in a competitive manner and generally a single kernel is active for each input. We may also use the sigmoid function instead of softmax and thereby allow multiple kernels to be used in a cooperative manner:

$$\eta_m(\mathbf{x}|\mathbf{V}) = \frac{1}{\exp(-\langle \mathbf{v}_m, \mathbf{x}^{\mathcal{G}} \rangle - v_{m0})} \qquad \forall m. \tag{16}$$

The gating model parameters are updated at each iteration by calculating $\partial J(\mathbf{V})/\partial \mathbf{V}$ and performing a gradient-descent step (Gönen and Alpaydın, 2008).

Inspired from LMKL, two methods that learn a data-dependent kernel function are used for image recognition applications (Yang et al., 2009a,b, 2010); they differ in their gating models that are constants rather than functions of the input. Yang et al. (2009a) divide the training set into clusters as a preprocessing step, and then cluster-specific kernel weights are learned using alternating optimization. The combined kernel function can be written as

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P} \eta_{c_i}^m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \eta_{c_j}^m$$

where $\eta_{c_i}^m$ corresponds to the weight of kernel $k_m(\cdot, \cdot)$ in the cluster $\mathbf{x}_i$ belongs to. The kernel weights of the cluster that the test instance is assigned to are used in the testing phase. Yang et al. (2009b, 2010) use instance-specific kernel weights instead of cluster-specific weights. The corresponding combined kernel function is

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{P} \eta_i^m k_m(\mathbf{x}_i^m, \mathbf{x}_j^m) \eta_j^m$$

where η_i^m corresponds to the weight of kernel $k_m(\cdot, \cdot)$ for $\mathbf{x}_i$ and these instance-specific weights are optimized using alternating optimization over the training set. In the testing phase, the kernel weights for a test instance are all taken to be equal.

3.11 Bayesian Approaches

Girolami and Rogers (2005) formulate a Bayesian hierarchical model and derive variational Bayes estimators for classification and regression problems. The proposed decision function can be formulated as

$$f(\mathbf{x}) = \sum_{i=0}^{N} \alpha_i \sum_{m=1}^{P} \eta_m k_m(\mathbf{x}_i^m, \mathbf{x}^m)$$

where η is modeled with a Dirichlet prior and α is modeled with a zero-mean Gaussian with an inverse gamma variance prior. Damoulas and Girolami (2009b) extend this method by adding auxiliary variables and developing a Gibbs sampler. Multinomial probit likelihood is used to obtain an efficient sampling procedure. Damoulas and Girolami (2008, 2009a) apply these methods to different bioinformatics problems, such as protein fold recognition and remote homology problems, and improve the prediction performances for these tasks.

Girolami and Zhong (2007) use the kernel combination idea for the covariance matrices in GPs. Instead of using a single covariance matrix, they define a weighted sum of covariance matrices calculated over different data sources. A joint inference is performed for both the GP coefficients and the kernel combination weights.

Similar to LMKL, Christoudias et al. (2009) develop a Bayesian approach for combining different feature representations in a data-dependent way under the GP framework. A common covariance function is obtained by combining the covariances of feature representations in a nonlinear manner. This formulation can identify the noisy data instances for each feature representation and prevent them from being used. Classification is performed using the standard GP approach with the common covariance function.

3.12 Boosting Approaches

Inspired from ensemble and boosting methods, Bennett et al. (2002) modify the decision function in order to use multiple kernels:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \sum_{m=1}^{P} \alpha_i^m k_m(\mathbf{x}_i^m, \mathbf{x}^m) + b.$$

The parameters $\{\alpha^m\}_{m=1}^P$ and b of the KRR model are learned using gradient-descent in the function space. The columns of the combined kernel matrix are generated on the fly from the heterogeneous kernels. Bi et al. (2004) develop column generation boosting methods for binary classification and regression problems. At each iteration, the proposed methods solve an LP or a QP on a working set depending on the regularization term used.

Crammer et al. (2003) modify the boosting methodology to work with kernels by rewriting two loss functions for a pair of data instances by considering the pair as a single instance:

$$\begin{aligned}\text{ExpLoss}(k(\mathbf{x}_i, \mathbf{x}_j), y_i y_j) &= \exp(-y_i y_j k(\mathbf{x}_i, \mathbf{x}_j))\\ \text{LogLoss}(k(\mathbf{x}_i, \mathbf{x}_j), y_i y_j) &= \log(1 + \exp(-y_i y_j k(\mathbf{x}_i, \mathbf{x}_j))).\end{aligned}$$

We iteratively update the combined kernel matrix using one of these two loss functions.

4. Experiments

In order to compare several MKL algorithms, we perform 10 different experiments on four data sets that are composed of different feature representations. We use both the linear kernel and the Gaussian kernel in our experiments; we will give our results with the linear kernel first and then compare them with the results of the Gaussian kernel. The kernel matrices are normalized to unit diagonal before training.

4.1 Compared Algorithms

We implement two single-kernel SVM and 16 representative MKL algorithms in MATLAB[1] and solve the optimization problems with the MOSEK optimization software (Mosek, 2011).

We train SVMs on each feature representation singly and report the results of the one with the highest average validation accuracy, which will be referred as SVM (best). We also train an SVM on the concatenation of all feature representations, which will be referred as SVM (all).

RBMKL denotes rule-based MKL algorithms discussed in Section 3.1. RBMKL (mean) trains an SVM with the mean of the combined kernels. RBMKL (product) trains an SVM with the product of the combined kernels.

ABMKL denotes alignment-based MKL algorithms. For determining the kernel weights, ABMKL (ratio) uses the heuristic in (2) of Section 3.2 (Qiu and Lane, 2009), ABMKL (conic) solves the QCQP problem in (5) of Section 3.4 (Lanckriet et al., 2004a), and ABMKL (convex) solves the QP problem in (7) of Section 3.5 (He et al., 2008). In the second step, all methods train an SVM with the kernel calculated with these weights.

CABMKL denotes centered-alignment-based MKL algorithms. In the first step, CABMKL (linear) uses the analytical solution in (4) of Section 3.3 (Cortes et al., 2010a) and CABMKL (conic) solves

1. Implementations are available at `http://www.cmpe.boun.edu.tr/~gonen/mkl`.

the QP problem in (6) of Section 3.4 (Cortes et al., 2010a) for determining the kernel weights. In the second step, both methods train an SVM with the kernel calculated with these weights.

MKL is the original MKL algorithm of Bach et al. (2004) that is formulated as the SOCP problem in (9) of Section 3.8. SimpleMKL is the iterative algorithm of Rakotomamonjy et al. (2008) that uses projected gradient updates and trains SVMs at each iteration to solve the optimization problem in (10) of Section 3.8.

GMKL is the generalized MKL algorithm of Varma and Babu (2009) discussed in Section 3.9. In our implementation, $k_\eta(\cdot,\cdot)$ is the convex combination of base kernels and $r(\cdot)$ is taken as $1/2(\eta - \mathbf{1}/P)^\top(\eta - \mathbf{1}/P)$.

GLMKL denotes the group Lasso-based MKL algorithms proposed by Kloft et al. (2010b) and Xu et al. (2010a). GLMKL ($p = 1$) updates the kernel weights using (11) of Section 3.8 and learns a convex combination of the kernels. GLMKL ($p = 2$) updates the kernel weights setting $p = 2$ in (8) of Section 3.7 and learns a conic combination of the kernels.

NLMKL denotes the nonlinear MKL algorithm of Cortes et al. (2010b) discussed in Section 3.9 with the exception of replacing the KRR in the inner loop with an SVM as the base learner. NLMKL uses the quadratic kernel given in (12). NLMKL ($p = 1$) and NLMKL ($p = 2$) select the kernel weights from the sets $\mathcal{M}_1$ in (13) and $\mathcal{M}_2$ in (14), respectively. In our implementation, η_0 is taken as $\mathbf{0}$ and Λ is assigned to 1 arbitrarily.

LMKL denotes the localized MKL algorithm of Gönen and Alpaydın (2008) discussed in Section 3.10. LMKL (softmax) uses the softmax gating model in (15), whereas LMKL (sigmoid) uses the sigmoid gating model in (16). Both methods use the concatenation of all feature representations in the gating model.

4.2 Experimental Methodology

Our experimental methodology is as follows: Given a data set, if learning and test sets are not supplied separately, a random one-third is reserved as the test set and the remaining two-thirds is used as the learning set. If the learning set has more than 1000 data instances, it is resampled using 5×2 cross-validation to generate 10 training and validation sets, with stratification, otherwise, we use 30-fold cross-validation. The validation sets of all folds are used to optimize the common hyperparameter C (trying values 0.01, 0.1, 1, 10, and 100). The best hyperparameter configuration (the one that has the highest average accuracy on the validation folds) is used to train the final learners on the training folds. Their test accuracies, support vector percentages, active kernel[2] counts, and numbers of calls to the optimization toolbox for solving an SVM optimization problem or a more complex optimization problem[3] are measured; we report their averages and standard deviations. The active kernel count and the number of calls to the optimization toolbox for SVM (best) are taken as 1 and P, respectively, because it uses only one of the feature representations but needs to train the individual SVMs on all feature representations before choosing the best. Similarly, the active kernel count and the number of calls to the optimization toolbox for SVM (all) are taken as P and 1, respectively, because it uses all of the feature representations but trains a single SVM.

2. A kernel is *active*, if it needs to be calculated to make a prediction for an unseen test instance.
3. All algorithms except the MKL formulation of Bach et al. (2004), MKL, solve QP problems when they call the optimization toolbox, whereas MKL solves an SOCP problem.

The test accuracies and support vector percentages are compared using the 5×2 cv paired F test (Alpaydın, 1999) or the paired t test according to the resampling scheme used. The active kernel counts and the number of calls to the optimization toolbox are compared using the Wilcoxon's signed-rank test (Wilcoxon, 1945). For all statistical tests, the significance level, α, is taken as 0.05. We want to test if by combining kernels, we get accuracy higher than any of the single kernels. In the result tables, a superscript a denotes that the performance values of SVM (best) and the compared algorithm are statistically significantly different, where $\overline{\text{a}}$ and $\underline{\text{a}}$ denote that the compared algorithm has statistically significantly higher and lower average than SVM (best), respectively. Similarly, we want to test if an algorithm is better than a straightforward concatenation of the input features, SVM (all), and if it is better than fixed combination, namely, RBMKL (mean); for those, we use the superscripts b and c, respectively.

4.3 Protein Fold Prediction Experiments

We perform experiments on the Protein Fold (PROTEIN) prediction data set[4] from the MKL Repository, composed of 10 different feature representations and two kernels for 694 instances (311 for training and 383 for testing). The properties of these feature representations are summarized in Table 3. We construct a binary classification problem by combining the major structural classes $\{\alpha, \beta\}$ into one class and $\{\alpha/\beta, \alpha+\beta\}$ into another class. Due to the small size of this data set, we use 30-fold cross-validation and the paired t test. We do three experiments on this data set using three different subsets of kernels.

Name	Dimension	Data Source
COM	20	Amino-acid composition
SEC	21	Predicted secondary structure
HYD	21	Hydrophobicity
VOL	21	Van der Waals volume
POL	21	Polarity
PLZ	21	Polarizability
L1	22	Pseudo amino-acid composition at interval 1
L4	28	Pseudo amino-acid composition at interval 4
L14	48	Pseudo amino-acid composition at interval 14
L30	80	Pseudo amino-acid composition at interval 30
BLO	311	Smith-Waterman scores with the BLOSUM 62 matrix
PAM	311	Smith-Waterman scores with the PAM 50 matrix

Table 3: Multiple feature representations in the PROTEIN data set.

Table 4 lists the performance values of all algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ). All combination algorithms except RBMKL (product) and GMKL outperform SVM (best) by more than four per cent in terms of average test accuracy. NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) are the only four algorithms that obtain more than 80 per cent average test accuracy and are statistically significantly more accurate than SVM (best), SVM (all), and RBMKL (mean). Nonlinear combination algorithms, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), have the disadvantage that they store statistically significantly more

4. Available at `http://mkl.ucsd.edu/dataset/protein-fold-prediction`.

support vectors than all other algorithms. ABMKL (conic) and CABMKL (conic) are the two MKL algorithms that perform kernel selection and use less than five kernels on the average, while the others use all six kernels, except CABMKL (linear) which uses five kernels in one of 30 folds. The two-step algorithms, except GMKL, LMKL (softmax), and LMKL (sigmoid), need to solve fewer than 20 SVM problems on the average. GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer optimization problems than all the other two-step algorithms. LMKL (softmax) and LMKL (sigmoid) solve many SVM problems; the large standard deviations for this performance value are mainly due to the random initialization of the gating model parameters and it takes longer for some folds to converge.

Table 5 summarizes the performance values of all algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30). All combination algorithms except RBMKL (product) outperform SVM (best) by more than two per cent in terms of average test accuracy. NLMKL ($p = 1$) and NLMKL ($p = 2$) are the only two algorithm that obtain more than 85 per cent average test accuracy and are statistically significantly more accurate than SVM (best), SVM (all), and RBMKL (mean). When the number of kernels combined becomes large as in this experiment, as a result of multiplication, RBMKL (product) starts to have very small kernel values at the off-diagonal entries of the combined kernel matrix. This causes the classifier to behave like a nearest-neighbor classifier by storing many support vectors and to perform badly in terms of average test accuracy. As observed in the previous experiment, the nonlinear combination algorithms, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), store statistically significantly more support vectors than all other algorithms. ABMKL (conic), ABMKL (convex), CABMKL (linear), CABMKL (conic), MKL, SimpleMKL, and GMKL are the seven MKL algorithms that perform kernel selection and use fewer than 10 kernels on the average, while others use all 10 kernels. Similar to the results of the previous experiment, GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer optimization problems than all the other two-step algorithms and the very high standard deviations for LMKL (softmax) and LMKL (sigmoid) are also observed in this experiment.

Table 6 gives the performance values of all algorithms on the PROTEIN data set with a larger set of kernels, namely, (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30-BLO-PAM). All combination algorithms except RBMKL (product) outperform SVM (best) by more than three per cent in terms of average test accuracy. NLMKL ($p = 1$) and NLMKL ($p = 2$) are the only two algorithms that obtain more than 87 per cent average test accuracy. In this experiment, ABMKL (ratio), GMKL, GLMKL ($p = 1$), GLMKL ($p = 2$), NLMKL ($p = 1$), NLMKL ($p = 2$), and LMKL (sigmoid) are statistically significantly more accurate than SVM (best), SVM (all), and RBMKL (mean). As noted in the two previous experiments, the nonlinear combination algorithms, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), store statistically significantly more support vectors than all other algorithms. ABMKL (conic), ABMKL (convex), CABMKL (linear), CABMKL (conic), MKL, SimpleMKL, and GMKL are the seven MKL algorithms that perform kernel selection and use fewer than 12 kernels on the average, while others use all 12 kernels, except GLMKL ($p = 1$) which uses 11 kernels in one of 30 folds. Similar to the results of the two previous experiments, GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer optimization problems than all the other two-step algorithms, but the very high standard deviations for LMKL (softmax) and LMKL (sigmoid) are not observed in this experiment.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	72.06±0.74 $^{\underline{bc}}$	58.29±1.00 $^{\underline{bc}}$	1.00±0.00 $^{\underline{bc}}$	6.00± 0.00 $^{\overline{bc}}$
SVM (all)	79.13±0.45$^{\overline{a}\ \overline{c}}$	62.14±1.04$^{\overline{a}\ \overline{c}}$	6.00±0.00	1.00± 0.00
RBMKL (mean)	78.01±0.63	60.89±1.02	6.00±0.00	1.00± 0.00
RBMKL (product)	72.35±0.95 $^{\underline{bc}}$	100.00±0.00$^{\overline{abc}}$	6.00±0.00	1.00± 0.00
ABMKL (conic)	79.03±0.92$^{\overline{a}\ \overline{c}}$	49.96±1.01$^{\underline{abc}}$	4.60±0.50$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (convex)	76.90±1.17$^{\overline{a}\underline{bc}}$	29.54±0.89$^{\underline{abc}}$	6.00±0.00	1.00± 0.00
ABMKL (ratio)	78.06±0.62	56.95±1.07$^{\underline{abc}}$	6.00±0.00	1.00± 0.00
CABMKL (linear)	79.51±0.78$^{\overline{abc}}$	49.81±0.82$^{\underline{abc}}$	5.97±0.18$^{\overline{a}\underline{bc}}$	1.00± 0.00
CABMKL (conic)	79.28±0.97$^{\overline{a}\ \overline{c}}$	49.84±0.77$^{\underline{abc}}$	4.73±0.52$^{\overline{a}\underline{bc}}$	1.00± 0.00
MKL	76.38±1.19$^{\overline{a}\underline{bc}}$	29.65±1.02$^{\underline{abc}}$	6.00±0.00	1.00± 0.00
SimpleMKL	76.34±1.24$^{\overline{a}\underline{bc}}$	29.62±1.08$^{\underline{abc}}$	6.00±0.00	18.83± 4.27$^{\overline{abc}}$
GMKL	74.96±0.50$^{\overline{a}\underline{bc}}$	79.85±0.70$^{\overline{abc}}$	2.37±0.56$^{\overline{a}\underline{bc}}$	37.10± 3.23$^{\overline{abc}}$
GLMKL ($p=1$)	77.71±0.96	55.80±0.95$^{\underline{abc}}$	6.00±0.00	6.10± 0.31$^{\overline{abc}}$
GLMKL ($p=2$)	77.20±0.42$^{\overline{a}\underline{bc}}$	75.34±0.70$^{\overline{abc}}$	6.00±0.00	5.00± 0.00$^{\underline{a}\overline{bc}}$
NLMKL ($p=1$)	83.49±0.76$^{\overline{abc}}$	85.67±0.86$^{\overline{abc}}$	6.00±0.00	17.50± 0.51$^{\overline{abc}}$
NLMKL ($p=2$)	82.30±0.62$^{\overline{abc}}$	89.57±0.77$^{\overline{abc}}$	6.00±0.00	13.40± 4.41$^{\overline{abc}}$
LMKL (softmax)	80.24±1.37$^{\overline{abc}}$	27.24±1.76$^{\underline{abc}}$	6.00±0.00	85.27±41.77$^{\overline{abc}}$
LMKL (sigmoid)	81.91±0.92$^{\overline{abc}}$	30.95±2.74$^{\underline{abc}}$	6.00±0.00	103.90±62.69$^{\overline{abc}}$

Table 4: Performances of single-kernel SVM and representative MKL algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ) using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	72.15±0.68 $^{\underline{bc}}$	47.50±1.25 $^{\overline{b}\underline{c}}$	1.00±0.00 $^{\underline{bc}}$	10.00± 0.00 $^{\overline{bc}}$
SVM (all)	79.63±0.74$^{\overline{a}\ \underline{c}}$	43.45±1.00$^{\underline{a}\ \underline{c}}$	10.00±0.00	1.00± 0.00
RBMKL (mean)	81.32±0.74	61.67±1.31	10.00±0.00	1.00± 0.00
RBMKL (product)	53.04±0.21$^{\underline{abc}}$	100.00±0.00$^{\overline{abc}}$	10.00±0.00	1.00± 0.00
ABMKL (conic)	80.45±0.68$^{\overline{ab}\underline{c}}$	48.16±1.08$^{\overline{ab}\underline{c}}$	6.90±0.66$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (convex)	77.47±0.62$^{\overline{a}\underline{bc}}$	87.86±0.76$^{\overline{abc}}$	9.03±0.61$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (ratio)	76.22±1.14$^{\overline{a}\underline{bc}}$	35.54±1.01$^{\underline{abc}}$	10.00±0.00	1.00± 0.00
CABMKL (linear)	77.15±0.63$^{\overline{a}\underline{bc}}$	73.84±0.80$^{\overline{abc}}$	9.90±0.31$^{\overline{a}\underline{bc}}$	1.00± 0.00
CABMKL (conic)	81.02±0.67	48.32±0.86$^{\overline{ab}\underline{c}}$	6.93±0.74$^{\overline{a}\underline{bc}}$	1.00± 0.00
MKL	79.74±1.02$^{\overline{a}\ \underline{c}}$	56.00±0.85$^{\overline{ab}\underline{c}}$	8.73±0.52$^{\overline{a}\underline{bc}}$	1.00± 0.00
SimpleMKL	74.53±0.90$^{\overline{a}\underline{bc}}$	80.22±1.05$^{\overline{abc}}$	4.73±1.14$^{\overline{a}\underline{bc}}$	23.83± 7.46$^{\overline{abc}}$
GMKL	74.68±0.68$^{\overline{a}\underline{bc}}$	80.36±0.83$^{\overline{abc}}$	5.73±0.91$^{\overline{a}\underline{bc}}$	29.10± 8.47$^{\overline{abc}}$
GLMKL ($p=1$)	79.77±0.86$^{\overline{a}\ \underline{c}}$	55.94±0.93$^{\overline{ab}\underline{c}}$	10.00±0.00	6.87± 0.57$^{\underline{a}\overline{bc}}$
GLMKL ($p=2$)	78.00±0.43$^{\overline{a}\underline{bc}}$	72.49±1.00$^{\overline{abc}}$	10.00±0.00	5.03± 0.18$^{\underline{a}\overline{bc}}$
NLMKL ($p=1$)	85.38±0.70$^{\overline{abc}}$	93.84±0.51$^{\overline{abc}}$	10.00±0.00	14.77± 0.43$^{\overline{abc}}$
NLMKL ($p=2$)	85.40±0.69$^{\overline{abc}}$	93.86±0.51$^{\overline{abc}}$	10.00±0.00	18.00± 0.00$^{\overline{abc}}$
LMKL (softmax)	81.11±1.82	36.00±3.61$^{\underline{abc}}$	10.00±0.00	34.40±23.12$^{\overline{abc}}$
LMKL (sigmoid)	81.90±2.01	51.94±2.14$^{\overline{ab}\underline{c}}$	10.00±0.00	31.63±13.17$^{\overline{abc}}$

Table 5: Performances of single-kernel SVM and representative MKL algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30) using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	78.37±1.08 $^{\underline{bc}}$	93.09±0.73 $^{\overline{bc}}$	1.00±0.00 $^{\underline{bc}}$	12.00± 0.00 $^{\overline{bc}}$
SVM (all)	82.01±0.76$^{\overline{a}\ \underline{c}}$	89.32±0.99$^{\underline{a}\ \overline{c}}$	12.00±0.00	1.00± 0.00
RBMKL (mean)	83.57±0.59	65.94±0.93	12.00±0.00	1.00± 0.00
RBMKL (product)	53.04±0.21$^{\underline{abc}}$	100.00±0.00$^{\overline{abc}}$	12.00±0.00	1.00± 0.00
ABMKL (conic)	83.52±0.94	63.07±1.35$^{\underline{abc}}$	7.30±0.88$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (convex)	83.76±1.02	64.36±1.56$^{\underline{abc}}$	6.87±0.94$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (ratio)	85.65±0.67$^{\overline{abc}}$	57.87±1.24$^{\underline{abc}}$	12.00±0.00	1.00± 0.00
CABMKL (linear)	83.48±0.92	68.00±1.48$^{\underline{ab}\overline{c}}$	11.87±0.35$^{\overline{a}\underline{bc}}$	1.00± 0.00
CABMKL (conic)	83.43±0.95	62.12±1.63$^{\underline{abc}}$	8.43±0.73$^{\overline{a}\underline{bc}}$	1.00± 0.00
MKL	83.55±1.25	81.75±1.06$^{\underline{ab}\overline{c}}$	7.67±0.76$^{\overline{a}\underline{bc}}$	1.00± 0.00
SimpleMKL	83.96±1.20	86.41±0.98$^{\underline{ab}\overline{c}}$	9.83±0.91$^{\overline{a}\underline{bc}}$	54.53± 9.92$^{\overline{abc}}$
GMKL	85.67±0.91$^{\overline{abc}}$	79.53±2.71$^{\underline{ab}\overline{c}}$	9.93±0.74$^{\overline{a}\underline{bc}}$	47.40±10.81$^{\overline{abc}}$
GLMKL ($p=1$)	85.96±0.96$^{\overline{abc}}$	79.06±1.04$^{\underline{ab}\overline{c}}$	11.97±0.18$^{\overline{a}\underline{bc}}$	14.77± 0.57$^{\overline{abc}}$
GLMKL ($p=2$)	85.02±1.20$^{\overline{abc}}$	62.06±1.02$^{\underline{abc}}$	12.00±0.00	5.60± 0.67$^{\underline{a}\overline{bc}}$
NLMKL ($p=1$)	87.00±0.66$^{\overline{abc}}$	96.78±0.32$^{\overline{abc}}$	12.00±0.00	4.83± 0.38$^{\underline{a}\overline{bc}}$
NLMKL ($p=2$)	87.28±0.65$^{\overline{abc}}$	96.64±0.32$^{\overline{abc}}$	12.00±0.00	17.77± 0.43$^{\overline{abc}}$
LMKL (softmax)	83.72±1.35	37.55±2.54$^{\underline{abc}}$	12.00±0.00	25.97± 5.75$^{\overline{abc}}$
LMKL (sigmoid)	85.06±0.83$^{\overline{abc}}$	48.99±1.59$^{\underline{abc}}$	12.00±0.00	25.40± 9.36$^{\overline{abc}}$

Table 6: Performances of single-kernel SVM and representative MKL algorithms on the PROTEIN data set with (COM-SEC-HYD-VOL-POL-PLZ-L1-L4-L14-L30-BLO-PAM) using the linear kernel.

4.4 Pendigits Digit Recognition Experiments

We perform experiments on the Pendigits (PENDIGITS) digit recognition data set[5] from the MKL Repository, composed of four different feature representations for 10,992 instances (7,494 for training and 3,498 for testing). The properties of these feature representations are summarized in Table 7. Two binary classification problems are generated from the PENDIGITS data set: In the PENDIGITS-EO data set, we separate even digits from odd digits; in the PENDIGITS-SL data set, we separate small (‘0’ - ‘4’) digits from large (‘5’ - ‘9’) digits.

Name	Dimension	Data Source
DYN	16	8 successive pen points on two-dimensional coordinate system
STA4	16	4×4 image bitmap representation
STA8	64	8×8 image bitmap representation
STA16	256	16×16 image bitmap representation

Table 7: Multiple feature representations in the PENDIGITS data set.

Table 8 summarizes the performance values of all algorithms on the PENDIGITS-EO data set. We see that SVM (best) is outperformed (by more than three per cent) by all other algorithms in

5. Available at `http://mkl.ucsd.edu/dataset/pendigits`.

terms of average test accuracy, which implies that integrating different information sources helps. RBMKL (product), NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) achieve statistically significantly higher average test accuracies than the other MKL algorithms. NLMKL ($p = 1$) and NLMKL ($p = 2$) are the only two algorithms that get more than 99 percent average test accuracy and improve the average test accuracy of RBMKL (mean) statistically significantly, by nearly six per cent. When we look at the percentages of support vectors stored, we see that RBMKL (product) stores statistically significantly more support vectors than the other algorithms, whereas LMKL (softmax) and LMKL (sigmoid) store statistically significantly fewer support vectors. All combination algorithms except ABMKL (convex) use four kernels in all folds. All two-step algorithms except LMKL (softmax) and LMKL (sigmoid) need to solve less than 15 SVM optimization problems on the average. As observed before, LMKL (softmax) and LMKL (sigmoid) have very high standard deviations in the number of SVM optimization calls due to the random initialization of the gating model parameters; note that convergence may be slow at times, but the standard deviations of the test accuracy are small.

Table 9 lists the performance values of all algorithms on the PENDIGITS-SL data set. We again see that SVM (best) is outperformed (more than five per cent) by all other algorithms in terms of average test accuracy. RBMKL (product), NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) achieve statistically significantly higher average test accuracies than the other MKL algorithms. Similar to the results on the PENDIGITS-EO data set, NLMKL ($p = 1$) and NLMKL ($p = 2$) are the only two algorithms that get more than 99 percent average test accuracy by improving the average test accuracy of RBMKL (mean) nearly eight per cent for this experiment. As observed on the PENDIGITS-EO data set, we see that RBMKL (product) stores statistically significantly more support vectors than the other algorithms, whereas LMKL (softmax) and LMKL (sigmoid) store fewer support vectors. All combination algorithms except ABMKL (convex) use four kernels in all folds, whereas this latter uses exactly three kernels in all folds by eliminating STA8 representation. All two-step algorithms except LMKL (softmax) and LMKL (sigmoid) need to solve less than 20 SVM optimization problems on the average. GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer SVM problems than the other two-step algorithms.

4.5 Multiple Features Digit Recognition Experiments

We perform experiments on the Multiple Features (MULTIFEAT) digit recognition data set[6] from the UCI Machine Learning Repository, composed of six different feature representations for 2,000 handwritten numerals. The properties of these feature representations are summarized in Table 10. Two binary classification problems are generated from the MULTIFEAT data set: In the MULTIFEAT-EO data set, we separate even digits from odd digits; in the MULTIFEAT-SL data set, we separate small ('0' - '4') digits from large ('5' - '9') digits. We do two experiments on these data set using two different subsets of feature representations.

Table 11 gives the performance values of all algorithms on the MULTIFEAT-EO data set with (FOU-KAR-PIX-ZER). Though all algorithms except CABMKL (linear) have higher average test accuracies than SVM (best); only LMKL (sigmoid) is statistically significantly more accurate than SVM (best), SVM (all), and RBMKL (mean). Note that even though RBMKL (product) is not more accurate than SVM (all) or RBMKL (mean), nonlinear and data-dependent algorithms, namely, NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid), are more accurate than these two

6. Available at `http://archive.ics.uci.edu/ml/datasets/Multiple+Features`.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	$88.93\pm0.28^{\ \underline{bc}}$	$20.90\pm1.22^{\ \ \overline{c}}$	$1.00\pm0.00^{\ \underline{bc}}$	$4.00\pm\ 0.00^{\ \overline{bc}}$
SVM (all)	$92.12\pm0.42^{\overline{a}\ \underline{c}}$	$22.22\pm0.72^{\ \ \overline{c}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
RBMKL (mean)	93.34 ± 0.28	18.91 ± 0.67	4.00 ± 0.00	$1.00\pm\ 0.00$
RBMKL (product)	$98.46\pm0.16^{\overline{abc}}$	$51.08\pm0.48^{\overline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
ABMKL (conic)	93.40 ± 0.15	$17.52\pm0.73^{\underline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
ABMKL (convex)	93.53 ± 0.26	$13.83\pm0.75^{\underline{abc}}$	$3.90\pm0.32^{\overline{a}\underline{bc}}$	$1.00\pm\ 0.00$
ABMKL (ratio)	93.35 ± 0.20	18.89 ± 0.68	4.00 ± 0.00	$1.00\pm\ 0.00$
CABMKL (linear)	93.42 ± 0.16	$17.48\pm0.74^{\underline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
CABMKL (conic)	93.42 ± 0.16	$17.48\pm0.74^{\underline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
MKL	93.28 ± 0.29	$19.20\pm0.67^{\ \underline{b}\overline{c}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
SimpleMKL	93.29 ± 0.27	19.04 ± 0.71	4.00 ± 0.00	$8.70\pm\ 3.92^{\overline{abc}}$
GMKL	93.28 ± 0.26	19.08 ± 0.72	4.00 ± 0.00	$8.60\pm\ 3.66^{\overline{abc}}$
GLMKL ($p=1$)	93.34 ± 0.27	19.02 ± 0.73	4.00 ± 0.00	$3.20\pm\ 0.63^{\underline{a}\overline{bc}}$
GLMKL ($p=2$)	93.32 ± 0.25	$16.91\pm0.61^{\underline{abc}}$	4.00 ± 0.00	$3.80\pm\ 0.42^{\underline{a}\overline{bc}}$
NLMKL ($p=1$)	$99.36\pm0.08^{\overline{abc}}$	19.55 ± 0.48	4.00 ± 0.00	$11.60\pm\ 6.26^{\overline{abc}}$
NLMKL ($p=2$)	$99.38\pm0.07^{\overline{abc}}$	19.79 ± 0.52	4.00 ± 0.00	$10.90\pm\ 4.31^{\overline{abc}}$
LMKL (softmax)	$97.14\pm0.39^{\overline{abc}}$	$7.25\pm0.65^{\underline{abc}}$	4.00 ± 0.00	$97.70\pm55.48^{\overline{abc}}$
LMKL (sigmoid)	$97.80\pm0.20^{\overline{abc}}$	$11.71\pm0.71^{\underline{abc}}$	4.00 ± 0.00	$87.70\pm47.30^{\overline{abc}}$

Table 8: Performances of single-kernel SVM and representative MKL algorithms on the PENDIGITS-EO data set using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	$84.44\pm0.49^{\ \underline{bc}}$	$39.31\pm0.77^{\ \overline{bc}}$	$1.00\pm0.00^{\ \underline{bc}}$	$4.00\pm\ 0.00^{\ \overline{bc}}$
SVM (all)	$89.48\pm0.67^{\overline{a}\ \underline{c}}$	$19.55\pm0.61^{\underline{a}\ \overline{c}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
RBMKL (mean)	91.11 ± 0.34	16.22 ± 0.59	4.00 ± 0.00	$1.00\pm\ 0.00$
RBMKL (product)	$98.37\pm0.11^{\overline{abc}}$	$60.28\pm0.69^{\overline{abc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
ABMKL (conic)	90.97 ± 0.49	$20.93\pm0.46^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
ABMKL (convex)	90.85 ± 0.51	$24.59\pm0.69^{\underline{a}\overline{bc}}$	$3.00\pm0.00^{\overline{a}\underline{bc}}$	$1.00\pm\ 0.00$
ABMKL (ratio)	91.12 ± 0.32	16.23 ± 0.57	4.00 ± 0.00	$1.00\pm\ 0.00$
CABMKL (linear)	91.02 ± 0.47	$20.89\pm0.49^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
CABMKL (conic)	91.02 ± 0.47	$20.90\pm0.50^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
MKL	90.85 ± 0.45	$23.59\pm0.56^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$1.00\pm\ 0.00$
SimpleMKL	90.84 ± 0.50	$23.48\pm0.55^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$14.50\pm\ 3.92^{\overline{abc}}$
GMKL	90.85 ± 0.47	$23.46\pm0.54^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$15.60\pm\ 3.34^{\overline{abc}}$
GLMKL ($p=1$)	90.90 ± 0.46	$23.33\pm0.57^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$4.90\pm\ 0.57^{\overline{abc}}$
GLMKL ($p=2$)	91.12 ± 0.44	$20.40\pm0.55^{\underline{a}\overline{bc}}$	4.00 ± 0.00	$4.00\pm\ 0.00^{\ \overline{bc}}$
NLMKL ($p=1$)	$99.11\pm0.10^{\overline{abc}}$	17.37 ± 0.17	4.00 ± 0.00	$18.10\pm\ 0.32^{\overline{abc}}$
NLMKL ($p=2$)	$99.07\pm0.12^{\overline{abc}}$	17.66 ± 0.23	4.00 ± 0.00	$10.90\pm\ 3.70^{\overline{abc}}$
LMKL (softmax)	$97.77\pm0.54^{\overline{abc}}$	$5.72\pm0.46^{\underline{abc}}$	4.00 ± 0.00	$116.60\pm73.34^{\overline{abc}}$
LMKL (sigmoid)	$97.13\pm0.40^{\overline{abc}}$	$6.69\pm0.27^{\underline{abc}}$	4.00 ± 0.00	$119.00\pm45.04^{\overline{abc}}$

Table 9: Performances of single-kernel SVM and representative MKL algorithms on the PENDIGITS-SL data set using the linear kernel.

Name	Dimension	Data Source
FAC	216	Profile correlations
FOU	76	Fourier coefficients of the shapes
KAR	64	Karhunen-Loève coefficients
MOR	6	Morphological features
PIX	240	Pixel averages in 2×3 windows
ZER	47	Zernike moments

Table 10: Multiple feature representations in the MULTIFEAT data set.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	$95.96\pm0.50^{\underline{\text{bc}}}$	$21.37\pm0.81^{\underline{\text{c}}}$	$1.00\pm0.00^{\underline{\text{bc}}}$	$4.00\pm0.00^{\overline{\text{bc}}}$
SVM (all)	97.79 ± 0.25	$21.63\pm0.73^{\underline{\text{c}}}$	4.00 ± 0.00	1.00 ± 0.00
RBMKL (mean)	97.94 ± 0.29	23.42 ± 0.79	4.00 ± 0.00	1.00 ± 0.00
RBMKL (product)	$96.43\pm0.38^{\underline{\text{bc}}}$	$92.11\pm1.18^{\overline{\text{abc}}}$	4.00 ± 0.00	1.00 ± 0.00
ABMKL (conic)	97.85 ± 0.25	$19.40\pm1.02^{\underline{\text{abc}}}$	$2.00\pm0.00^{\overline{\text{a}}\underline{\text{bc}}}$	1.00 ± 0.00
ABMKL (convex)	$95.97\pm0.57^{\underline{\text{bc}}}$	$21.45\pm0.92^{\underline{\text{c}}}$	$1.20\pm0.42^{\underline{\text{bc}}}$	1.00 ± 0.00
ABMKL (ratio)	97.82 ± 0.32	$22.33\pm0.57^{\overline{\text{b}}\underline{\text{c}}}$	4.00 ± 0.00	1.00 ± 0.00
CABMKL (linear)	$95.78\pm0.37^{\underline{\text{bc}}}$	$19.25\pm1.09^{\underline{\text{bc}}}$	4.00 ± 0.00	1.00 ± 0.00
CABMKL (conic)	97.85 ± 0.25	$19.37\pm1.03^{\underline{\text{abc}}}$	$2.00\pm0.00^{\overline{\text{a}}\underline{\text{bc}}}$	1.00 ± 0.00
MKL	97.88 ± 0.31	$21.01\pm0.87^{\underline{\text{c}}}$	$3.50\pm0.53^{\overline{\text{a}}\underline{\text{bc}}}$	1.00 ± 0.00
SimpleMKL	97.87 ± 0.32	$20.90\pm0.94^{\underline{\text{c}}}$	$3.40\pm0.70^{\overline{\text{a}}\underline{\text{bc}}}$	$22.50\pm6.65^{\overline{\text{abc}}}$
GMKL	97.88 ± 0.31	$21.00\pm0.88^{\underline{\text{c}}}$	$3.50\pm0.53^{\overline{\text{a}}\underline{\text{bc}}}$	$25.90\pm10.05^{\overline{\text{abc}}}$
GLMKL ($p = 1$)	97.90 ± 0.25	$21.31\pm0.78^{\underline{\text{c}}}$	4.00 ± 0.00	$11.10\pm0.74^{\overline{\text{abc}}}$
GLMKL ($p = 2$)	98.01 ± 0.24	$19.19\pm0.61^{\underline{\text{bc}}}$	4.00 ± 0.00	$4.90\pm0.32^{\overline{\text{abc}}}$
NLMKL ($p = 1$)	98.67 ± 0.22	$56.91\pm1.17^{\overline{\text{abc}}}$	4.00 ± 0.00	$4.50\pm1.84^{\overline{\text{bc}}}$
NLMKL ($p = 2$)	98.61 ± 0.24	$53.61\pm1.20^{\overline{\text{abc}}}$	4.00 ± 0.00	$5.60\pm3.03^{\overline{\text{bc}}}$
LMKL (softmax)	98.16 ± 0.50	$17.40\pm1.17^{\underline{\text{abc}}}$	4.00 ± 0.00	$36.70\pm14.11^{\overline{\text{abc}}}$
LMKL (sigmoid)	$98.94\pm0.29^{\overline{\text{abc}}}$	$15.23\pm1.08^{\underline{\text{abc}}}$	4.00 ± 0.00	$88.20\pm36.00^{\overline{\text{abc}}}$

Table 11: Performances of single-kernel SVM and representative MKL algorithms on the MULTIFEAT-EO data set with (FOU-KAR-PIX-ZER) using the linear kernel.

algorithms. Alignment-based and centered-alignment-based MKL algorithms, namely, ABMKL (ratio), ABMKL (conic), ABMKL (convex), CABMKL (linear) and CABMKL (convex), are not more accurate than RBMKL (mean). We see that ABMKL (convex) and CABMKL (linear) are statistically significantly less accurate than SVM (all) and RBMKL (mean). If we compare the algorithms in terms of support vector percentages, we note that MKL algorithms that use products of the combined kernels, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), store statistically significantly more support vectors than all other algorithms. If we look at the active kernel counts, 10 out of 16 MKL algorithms use all four kernels. The two-step algorithms solve statistically significantly more optimization problems than the one-step algorithms.

Table 12 summarizes the performance values of all algorithms on the MULTIFEAT-EO data set with (FAC-FOU-KAR-MOR-PIX-ZER). We note that NLMKL ($p = 1$) and LMKL (sigmoid) are the two MKL algorithms that achieve average test accuracy greater than or equal to 99 per cent, while NLMKL ($p = 1$), NLMKL ($p = 2$), and LMKL (sigmoid) are statistically significantly more accurate than RBMKL (mean). All other MKL algorithms except RBMKL (product) and CABMKL (linear) achieve average test accuracies between 98 per cent and 99 per cent. Similar to the results of the previous experiment, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$) store statistically significantly more support vectors than all other algorithms. When we look at the number of active kernels, ABMKL (convex) selects only one kernel and this is the same kernel that SVM (best) picks. ABMKL (conic) and CABMKL (conic) use three kernels, whereas all other algorithms use more than five kernels on the average. GLMKL ($p = 1$), GLMKL ($p = 2$), NLMKL ($p = 1$), and NLMKL ($p = 2$) solve fewer optimization problems than the other two-step algorithms, namely, SimpleMKL, GMKL, LMKL (softmax), and LMKL (sigmoid).

Table 13 lists the performance values of all algorithms on the MULTIFEAT-SL data set with (FOU-KAR-PIX-ZER). SVM (best) is outperformed by the other algorithms on the average and this shows that, for this data set, combining multiple information sources, independently of the combination algorithm used, improves the average test accuracy. RBMKL (product), NLMKL ($p = 1$), NLMKL ($p = 2$), and LMKL (sigmoid) are the four MKL algorithms that achieve statistically significantly higher average test accuracies than RBMKL (best), SVM (all), RBMKL (mean). NLMKL ($p = 1$) and NLMKL ($p = 2$) are the two best algorithms and are statistically significantly more accurate than all other algorithms, except LMKL (sigmoid). However, NLMKL ($p = 1$) and NLMKL ($p = 2$) store statistically significantly more support vectors than all other algorithms, except RBMKL (product). All MKL algorithms use all of the kernels and the two-step algorithms solve statistically significantly more optimization problems than the one-step algorithms.

Table 14 gives the performance values of all algorithms on the MULTIFEAT-SL data set with (FAC-FOU-KAR-MOR-PIX-ZER). GLMKL ($p = 2$), NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) are the five MKL algorithms that achieve higher average test accuracies than RBMKL (mean). CABMKL (linear) is the only algorithm that has statistically significantly lower average test accuracy than SVM (best). No MKL algorithm achieves statistically significantly higher average test accuracies than SVM (best), SVM (all), and RBMKL (mean). MKL algorithms with nonlinear combination rules, namely, RBMKL (product), NLMKL ($p = 1$) and NLMKL ($p = 2$), again use more support vectors than the other algorithms, whereas LMKL with a data-dependent combination approach stores statistically significantly fewer support vectors. ABMKL (conic), ABMKL (convex), and CABMKL (conic) are the three MKL algorithms that perform kernel selection and use fewer than five kernels on the average, while others use all of the kernels. GLMKL ($p = 1$) and GLMKL ($p = 2$) solve statistically significantly fewer optimization problems than all the other two-step algorithms and the very high standard deviations for LMKL (softmax) and LMKL (sigmoid) are also observed in this experiment.

4.6 Internet Advertisements Experiments

We perform experiments on the Internet Advertisements (ADVERT) data set[7] from the UCI Machine Learning Repository, composed of five different feature representations (different bags of words); there is also some additional geometry information of the images, but we ignore them in our experiments due to missing values. After removing the data instances with missing values, we have a total

7. Available at `http://archive.ics.uci.edu/ml/datasets/Internet+Advertisements`.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	98.39±0.36	10.30±0.83 $^{\underline{\mathrm{bc}}}$	1.00±0.00 $^{\underline{\mathrm{bc}}}$	6.00± 0.00 $^{\overline{\mathrm{bc}}}$
SVM (all)	98.24±0.40	14.44±0.74	6.00±0.00	1.00± 0.00
RBMKL (mean)	98.09±0.31	15.16±0.83	6.00±0.00	1.00± 0.00
RBMKL (product)	95.87±0.31$^{\underline{\mathrm{abc}}}$	100.00±0.00$^{\overline{\mathrm{abc}}}$	6.00±0.00	1.00± 0.00
ABMKL (conic)	98.24±0.38	13.08±0.93	3.00±0.00$^{\overline{\mathrm{a}}\underline{\mathrm{bc}}}$	1.00± 0.00
ABMKL (convex)	98.39±0.36	10.30±0.83 $^{\underline{\mathrm{bc}}}$	1.00±0.00 $^{\underline{\mathrm{bc}}}$	1.00± 0.00
ABMKL (ratio)	98.19±0.25	14.11±0.64	6.00±0.00	1.00± 0.00
CABMKL (linear)	96.90±0.34	16.89±0.91$^{\overline{\mathrm{abc}}}$	5.90±0.32$^{\overline{\mathrm{a}}\underline{\mathrm{bc}}}$	1.00± 0.00
CABMKL (conic)	98.15±0.41	12.54±0.75	3.00±0.00$^{\overline{\mathrm{a}}\underline{\mathrm{bc}}}$	1.00± 0.00
MKL	98.31±0.34	14.88±0.81	5.40±0.70$^{\overline{\mathrm{a}}\underline{\mathrm{bc}}}$	1.00± 0.00
SimpleMKL	98.25±0.37	14.89±0.70	5.60±0.52$^{\overline{\mathrm{a}}\underline{\mathrm{bc}}}$	37.50±12.09$^{\overline{\mathrm{abc}}}$
GMKL	98.24±0.34	14.33±0.85$^{\overline{\mathrm{a}}\ \underline{\mathrm{c}}}$	5.60±0.52$^{\overline{\mathrm{a}}\underline{\mathrm{bc}}}$	31.70±10.79$^{\overline{\mathrm{abc}}}$
GLMKL ($p=1$)	98.28±0.31	14.44±0.87$^{\overline{\mathrm{a}}\ \underline{\mathrm{c}}}$	6.00±0.00	9.30± 1.25$^{\overline{\mathrm{abc}}}$
GLMKL ($p=2$)	98.37±0.28	17.04±0.80$^{\overline{\mathrm{abc}}}$	6.00±0.00	4.90± 0.32$^{\underline{\mathrm{a}}\overline{\mathrm{bc}}}$
NLMKL ($p=1$)	99.00±0.16 $^{\overline{\mathrm{c}}}$	47.50±1.27$^{\overline{\mathrm{abc}}}$	6.00±0.00	8.30± 2.71$^{\overline{\mathrm{abc}}}$
NLMKL ($p=2$)	98.93±0.18 $^{\overline{\mathrm{c}}}$	46.78±1.07$^{\overline{\mathrm{abc}}}$	6.00±0.00	12.00± 3.16$^{\overline{\mathrm{abc}}}$
LMKL (softmax)	98.34±0.25	11.36±1.83	6.00±0.00	94.90±24.73$^{\overline{\mathrm{abc}}}$
LMKL (sigmoid)	99.24±0.18 $^{\overline{\mathrm{c}}}$	17.88±1.06	6.00±0.00	94.90±57.64$^{\overline{\mathrm{abc}}}$

Table 12: Performances of single-kernel SVM and representative MKL algorithms on the MULTIFEAT-EO data set with (FAC-FOU-KAR-MOR-PIX-ZER) using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	90.54±1.12 $^{\underline{\mathrm{bc}}}$	28.90±1.69 $^{\underline{\mathrm{b}}\overline{\mathrm{c}}}$	1.00±0.00 $^{\underline{\mathrm{bc}}}$	4.00± 0.00 $^{\overline{\mathrm{bc}}}$
SVM (all)	94.45±0.44	40.26±1.28$^{\overline{\mathrm{a}}\ \overline{\mathrm{c}}}$	4.00±0.00	1.00± 0.00
RBMKL (mean)	95.00±0.76	24.73±1.19	4.00±0.00	1.00± 0.00
RBMKL (product)	96.51±0.31$^{\overline{\mathrm{abc}}}$	95.31±0.60$^{\overline{\mathrm{abc}}}$	4.00±0.00	1.00± 0.00
ABMKL (conic)	95.12±0.36	33.44±1.20$^{\overline{\mathrm{a}}\underline{\mathrm{b}}\overline{\mathrm{c}}}$	4.00±0.00	1.00± 0.00
ABMKL (convex)	94.51±0.59	24.34±1.19	4.00±0.00	1.00± 0.00
ABMKL (ratio)	94.93±0.73	24.88±1.02	4.00±0.00	1.00± 0.00
CABMKL (linear)	95.10±0.38	33.44±1.24$^{\overline{\mathrm{a}}\underline{\mathrm{b}}\overline{\mathrm{c}}}$	4.00±0.00	1.00± 0.00
CABMKL (conic)	95.10±0.38	33.44±1.24$^{\overline{\mathrm{a}}\underline{\mathrm{b}}\overline{\mathrm{c}}}$	4.00±0.00	1.00± 0.00
MKL	94.81±0.67	24.46±1.13	4.00±0.00	1.00± 0.00
SimpleMKL	94.84±0.64	24.40±1.18	4.00±0.00	15.50± 8.11$^{\overline{\mathrm{abc}}}$
GMKL	94.84±0.64	24.41±1.18	4.00±0.00	15.60± 8.07$^{\overline{\mathrm{abc}}}$
GLMKL ($p=1$)	94.84±0.69	24.34±1.27	4.00±0.00	6.20± 1.03$^{\overline{\mathrm{abc}}}$
GLMKL ($p=2$)	95.18±0.32	32.34±1.36$^{\overline{\mathrm{a}}\underline{\mathrm{b}}\overline{\mathrm{c}}}$	4.00±0.00	4.20± 0.63 $^{\overline{\mathrm{bc}}}$
NLMKL ($p=1$)	98.64±0.25$^{\overline{\mathrm{abc}}}$	50.17±1.31$^{\overline{\mathrm{abc}}}$	4.00±0.00	9.20± 4.80$^{\overline{\mathrm{abc}}}$
NLMKL ($p=2$)	98.63±0.28$^{\overline{\mathrm{abc}}}$	57.02±1.26$^{\overline{\mathrm{abc}}}$	4.00±0.00	9.10± 3.28$^{\overline{\mathrm{abc}}}$
LMKL (softmax)	96.24±0.90	24.16±3.29	4.00±0.00	41.70±31.28$^{\overline{\mathrm{abc}}}$
LMKL (sigmoid)	97.16±0.60$^{\overline{\mathrm{abc}}}$	20.18±1.06$^{\underline{\mathrm{abc}}}$	4.00±0.00	75.50±28.38$^{\overline{\mathrm{abc}}}$

Table 13: Performances of single-kernel SVM and representative MKL algorithms on the MULTIFEAT-SL data set with (FOU-KAR-PIX-ZER) using the linear kernel.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	94.99±0.85 $^{\underline{\text{bc}}}$	17.96±0.89 $^{\underline{\text{bc}}}$	1.00±0.00 $^{\underline{\text{bc}}}$	6.00± 0.00 $^{\overline{\text{bc}}}$
SVM (all)	97.69±0.44	23.34±1.13	6.00±0.00	1.00± 0.00
RBMKL (mean)	97.67±0.50	20.98±0.84	6.00±0.00	1.00± 0.00
RBMKL (product)	96.01±0.17 $^{\underline{\text{bc}}}$	97.58±0.48$^{\overline{\text{abc}}}$	6.00±0.00	1.00± 0.00
ABMKL (conic)	96.84±0.39	27.49±0.92$^{\overline{\text{abc}}}$	4.50±0.53$^{\overline{\text{a}}\underline{\text{bc}}}$	1.00± 0.00
ABMKL (convex)	96.46±0.34	33.78±0.90$^{\overline{\text{abc}}}$	4.60±0.52$^{\overline{\text{a}}\underline{\text{bc}}}$	1.00± 0.00
ABMKL (ratio)	97.66±0.46	20.95±0.88	6.00±0.00	1.00± 0.00
CABMKL (linear)	89.18±0.81$^{\underline{\text{abc}}}$	57.22±1.47$^{\overline{\text{abc}}}$	6.00±0.00	1.00± 0.00
CABMKL (conic)	96.84±0.39	27.57±0.95$^{\overline{\text{abc}}}$	4.50±0.53$^{\overline{\text{a}}\underline{\text{bc}}}$	1.00± 0.00
MKL	97.40±0.37	32.59±0.82$^{\overline{\text{abc}}}$	6.00±0.00	1.00± 0.00
SimpleMKL	97.51±0.37	32.53±0.94$^{\overline{\text{abc}}}$	6.00±0.00	14.40± 3.27$^{\overline{\text{abc}}}$
GMKL	97.51±0.35	32.73±1.01$^{\overline{\text{abc}}}$	6.00±0.00	14.20± 4.59$^{\overline{\text{abc}}}$
GLMKL ($p = 1$)	97.51±0.28	32.49±0.93$^{\overline{\text{abc}}}$	6.00±0.00	6.70± 0.95 $^{\overline{\text{bc}}}$
GLMKL ($p = 2$)	97.81±0.22	25.19±1.06$^{\overline{\text{abc}}}$	6.00±0.00	5.00± 0.82$^{\underline{\text{a}}\overline{\text{bc}}}$
NLMKL ($p = 1$)	98.79±0.28	38.44±0.96$^{\overline{\text{abc}}}$	6.00±0.00	12.10± 3.98$^{\overline{\text{abc}}}$
NLMKL ($p = 2$)	98.82±0.20	43.99±0.99$^{\overline{\text{abc}}}$	6.00±0.00	10.70± 4.62$^{\overline{\text{abc}}}$
LMKL (softmax)	97.79±0.62	14.71±1.10 $^{\underline{\text{bc}}}$	6.00±0.00	59.00±31.42$^{\overline{\text{abc}}}$
LMKL (sigmoid)	98.48±0.70	16.10±2.09 $^{\underline{\text{bc}}}$	6.00±0.00	107.60±76.90$^{\overline{\text{abc}}}$

Table 14: Performances of single-kernel SVM and representative MKL algorithms on the MULTIFEAT-SL data set with (FAC-FOU-KAR-MOR-PIX-ZER) using the linear kernel.

of 3,279 images in the data set. The properties of these feature representations are summarized in Table 15. The classification task is to predict whether an image is an advertisement or not.

Name	Dimension	Data Source
URL	457	Phrases occurring in the URL
ORIGURL	495	Phrases occurring in the URL of the image
ANCURL	472	Phrases occurring in the anchor text
ALT	111	Phrases occurring in the alternative text
CAPTION	19	Phrases occurring in the caption terms

Table 15: Multiple feature representations in the ADVERT data set.

Table 16 lists the performance values of all algorithms on the ADVERT data set. We can see that all MKL algorithms except RBMKL (product) achieve similar average test accuracies. However, no MKL algorithm is statistically significantly more accurate than RBMKL (mean), and ABMKL (convex) is statistically significantly worse. We see again that algorithms that combine kernels by multiplying them, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), store statistically significantly more support vectors than other MKL algorithms. 10 out of 16 MKL algorithms use all five kernels; ABMKL (conic) and ABMKL (convex) eliminate two representations, namely, URL and ORIGURL. GMKL ($p = 1$) and GMKL ($p = 2$) solve statistically significantly fewer optimization problems than the other two-step algorithms.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	95.45±0.31	64.90± 5.41 $^{\overline{bc}}$	1.00±0.00 $^{\underline{bc}}$	5.00± 0.00 $^{\overline{bc}}$
SVM (all)	96.43±0.24	41.99± 1.76	5.00±0.00	1.00± 0.00
RBMKL (mean)	96.53±0.58	34.40± 4.25	5.00±0.00	1.00± 0.00
RBMKL (product)	89.98±0.49$^{\underline{abc}}$	96.61± 1.71$^{\overline{abc}}$	5.00±0.00	1.00± 0.00
ABMKL (conic)	95.69±0.27	44.16± 2.65$^{\underline{a}\ \overline{c}}$	3.00±0.00$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (convex)	95.10±0.52 $^{\underline{bc}}$	58.07± 2.47 $^{\overline{bc}}$	3.00±0.00$^{\overline{a}\underline{bc}}$	1.00± 0.00
ABMKL (ratio)	96.23±0.61	35.07± 2.92	5.00±0.00	1.00± 0.00
CABMKL (linear)	95.86±0.19	36.43± 1.50	5.00±0.00	1.00± 0.00
CABMKL (conic)	95.84±0.19	38.06± 2.36	4.40±0.52$^{\overline{a}\underline{bc}}$	1.00± 0.00
MKL	96.32±0.50	35.82± 4.35	4.10±0.32$^{\overline{a}\underline{bc}}$	1.00± 0.00
SimpleMKL	96.37±0.46	33.78± 4.40	4.60±0.52$^{\overline{a}\underline{bc}}$	27.00± 7.39$^{\overline{abc}}$
GMKL	96.40±0.49	33.18± 3.49	4.70±0.48$^{\overline{a}\underline{bc}}$	27.20± 7.94$^{\overline{abc}}$
GLMKL ($p=1$)	96.35±0.55	32.81± 3.56	5.00±0.00	5.40± 1.07 $^{\overline{bc}}$
GLMKL ($p=2$)	96.56±0.32	35.62± 1.55	5.00±0.00	4.90± 0.74 $^{\overline{bc}}$
NLMKL ($p=1$)	95.96±0.50	67.63± 3.46 $^{\overline{bc}}$	5.00±0.00	15.90± 5.38$^{\overline{abc}}$
NLMKL ($p=2$)	96.13±0.31	65.70± 3.03 $^{\overline{bc}}$	5.00±0.00	13.00± 0.00$^{\overline{abc}}$
LMKL (softmax)	95.68±0.53	24.18± 5.74	5.00±0.00	38.80±24.11$^{\overline{abc}}$
LMKL (sigmoid)	95.49±0.48	18.22±12.16	5.00±0.00	56.60±53.70$^{\overline{abc}}$

Table 16: Performances of single-kernel SVM and representative MKL algorithms on the ADVERT data set using the linear kernel.

4.7 Overall Comparison

After comparing algorithms for each experiment separately, we give an overall comparison on 10 experiments using the nonparametric Friedman's test on rankings with the Tukey's honestly significant difference criterion as the post-hoc test (Demšar, 2006).

Figure 2 shows the overall comparison between the algorithms in terms of misclassification error. First of all, we see that combining multiple information sources clearly improves the classification performance because SVM (best) is worse than all other algorithms. GLMKL ($p = 2$), NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) are statistically significantly more accurate than SVM (best). MKL algorithms using a trained, weighted combination on the average seem a little worse (but not statistically significantly) than the untrained, unweighted sum, namely, RBMKL (mean). NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) are more accurate (but not statistically significantly) than RBMKL (mean). These results seem to suggest that if we want to improve the classification accuracy of MKL algorithms, we should investigate nonlinear and data-dependent approaches to better exploit information provided by different kernels.

Figure 3 illustrates the overall comparison between the algorithms in terms of the support vector percentages. We note that algorithms are clustered into three groups: (a) nonlinear MKL algorithms, (b) single-kernel SVM and linear MKL algorithms, and (c) data-dependent MKL algorithms. Nonlinear MKL algorithms, namely, RBMKL (product), NLMKL ($p = 1$) and NLMKL ($p = 2$), store more (but not statistically significantly) support vectors than single-kernel SVM and linear MKL algorithms, whereas they store statistically significantly more support vectors than

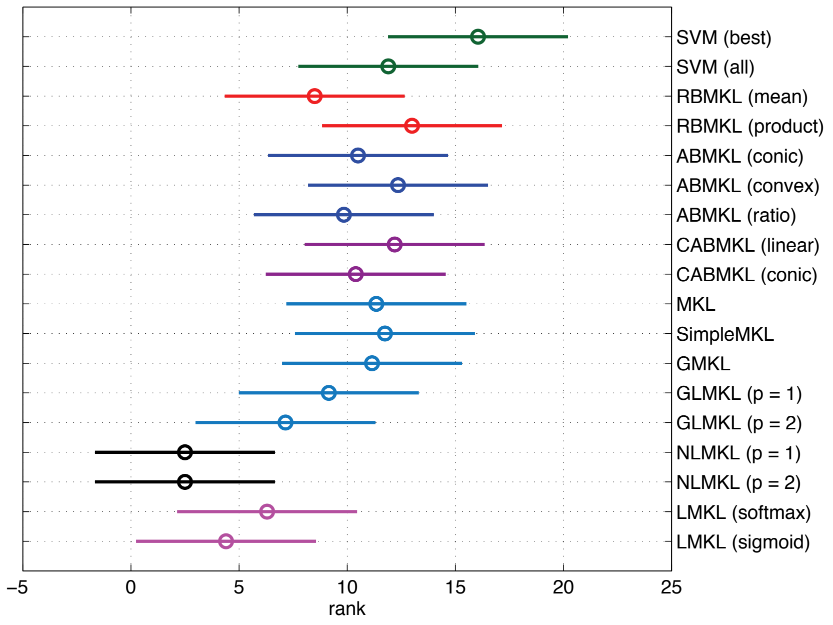


Figure 2: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of misclassification error using the linear kernel.

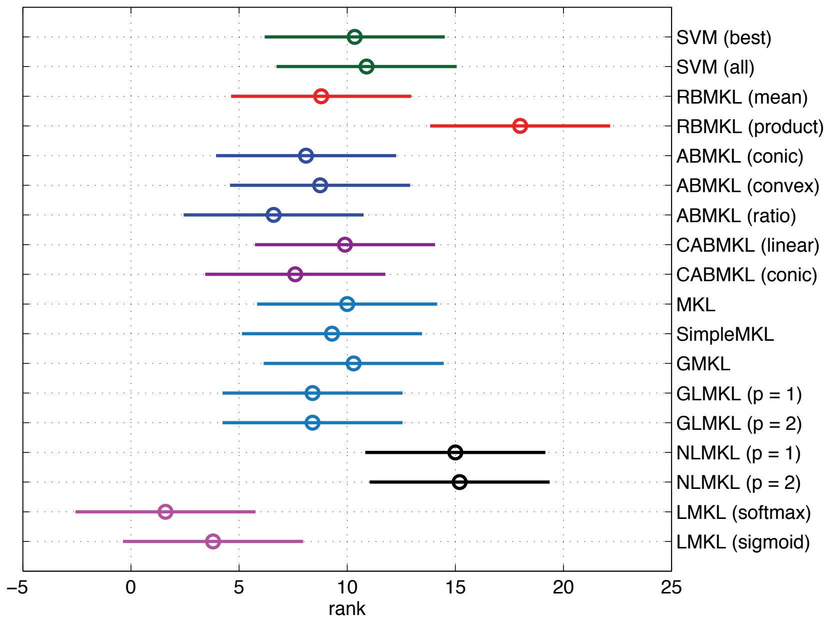


Figure 3: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of support vector percentages using the linear kernel.

data-dependent MKL algorithms. Data-dependent MKL algorithms, namely, LMKL (softmax) and LMKL (sigmoid), store fewer (but not statistically significantly) support vectors than single-kernel SVM and linear MKL algorithms, whereas LMKL (softmax) stores statistically significantly fewer support vectors than SVM (best) and SVM (all).

Figure 4 gives the overall comparison between the algorithms in terms of active kernel counts. We see that ABMKL (conic), ABMKL (convex), CABMKL (linear), CABMKL (conic), MKL, SimpleMKL, and GMKL use fewer kernels (statistically significantly in the case of the first two algorithms) than other combination algorithms. Even if we optimize the alignment and centered-alignment measures without any regularization on kernel weights using ABMKL (conic), ABMKL (convex), and CABMKL (conic), we obtain more sparse (but not statistically significantly) kernel combinations than MKL and SimpleMKL, which regularize kernel weights using the ℓ_1-norm. Trained nonlinear and data-dependent MKL algorithms, namely, NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid), tend to use all of the kernels without eliminating any of them, whereas data-dependent algorithms use the kernels in different parts of the feature space with the help of the gating model.

Figure 5 shows the overall comparison between the algorithms in terms of the optimization toolbox call counts. We clearly see that the two-step algorithms need to solve more optimization problems than the other combination algorithms. SimpleMKL, GMKL, NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid) require solving statistically significantly more optimization problems than the one-step algorithms, whereas the differences between the one-step algorithms and GLMKL ($p = 1$) and GLMKL ($p = 2$) are not statistically significant.

4.8 Overall Comparison Using Gaussian Kernel

We also replicate the same set of experiments, except on PENDIGITS data set, using three different Gaussian kernels for each feature representation. We select the kernel widths as $\{\sqrt{D_m}/2, \sqrt{D_m}, 2\sqrt{D_m}\}$ where D_m is the dimensionality of the corresponding feature representation.

Figure 6 shows the overall comparison between the algorithms in terms of misclassification error. We see that no MKL algorithm is statistically significantly better than RBMKL (mean) and conclude that combining complex Gaussian kernels does not help much. ABMKL (ratio), MKL, SimpleMKL, GMKL, GLMKL ($p = 1$), and GLMKL ($p = 2$) obtain accuracy results comparable to RBMKL (mean). As an important result, we see that nonlinear and data-dependent MKL algorithms, namely, NLMKL ($p = 1$), NLMKL ($p = 2$), LMKL (softmax), and LMKL (sigmoid), are outperformed (but not statistically significantly) by RBMKL (mean). If we have highly nonlinear kernels such as Gaussian kernels, there is no need to combine them in a nonlinear or data-dependent way.

Figure 7 illustrates the overall comparison between the algorithms in terms of the support vector percentages. Different from the results obtained with simple linear kernels, algorithms do not exhibit a clear grouping. However, data-dependent MKL algorithms, namely, LMKL (softmax) and LMKL (sigmoid), tend to use fewer support vectors, whereas nonlinear MKL algorithms, namely, RBMKL (product), NLMKL ($p = 1$), and NLMKL ($p = 2$), tend to store more support vectors than other algorithms.

Figure 8 gives the overall comparison between the algorithms in terms of active kernel counts. ABMKL (ratio), GLMKL ($p = 2$), NLMKL ($p = 1$), NLMKL ($p = 2$), and LMKL (sigmoid) do not eliminate any of the base kernels even though we have three different kernels for each feature representation. When combining complex Gaussian kernels, trained MKL algorithms do not improve the classification performance statistically significantly, but they can eliminate some of the kernels.

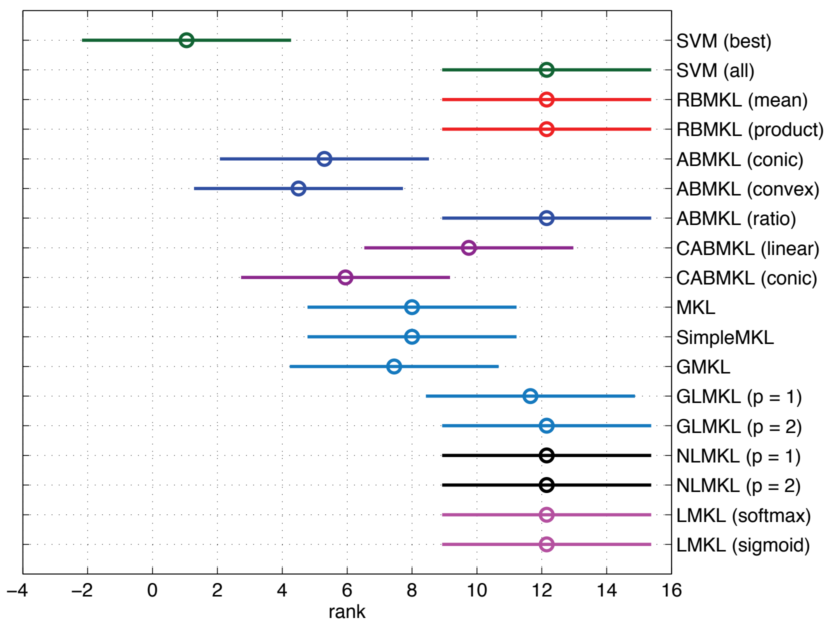


Figure 4: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of active kernel counts using the linear kernel.

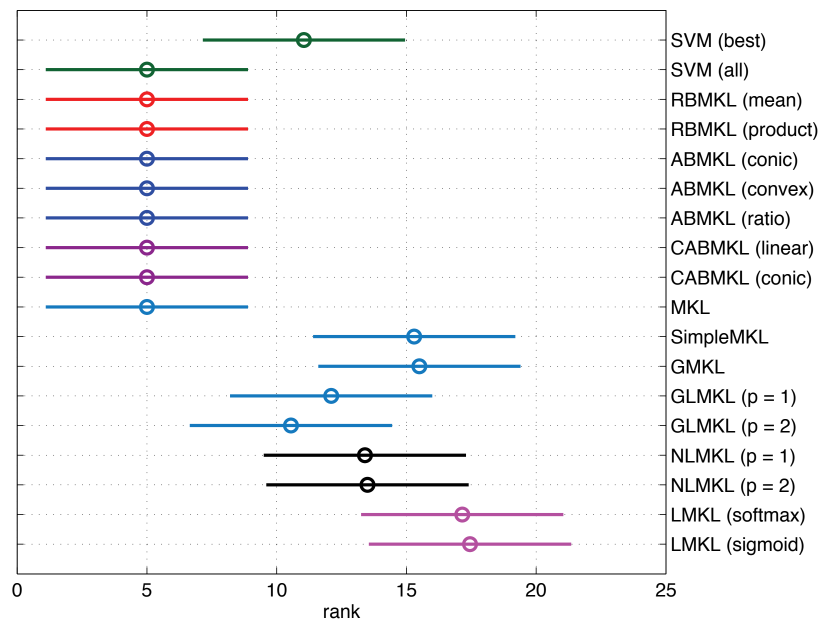


Figure 5: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of optimization toolbox call counts using the linear kernel.

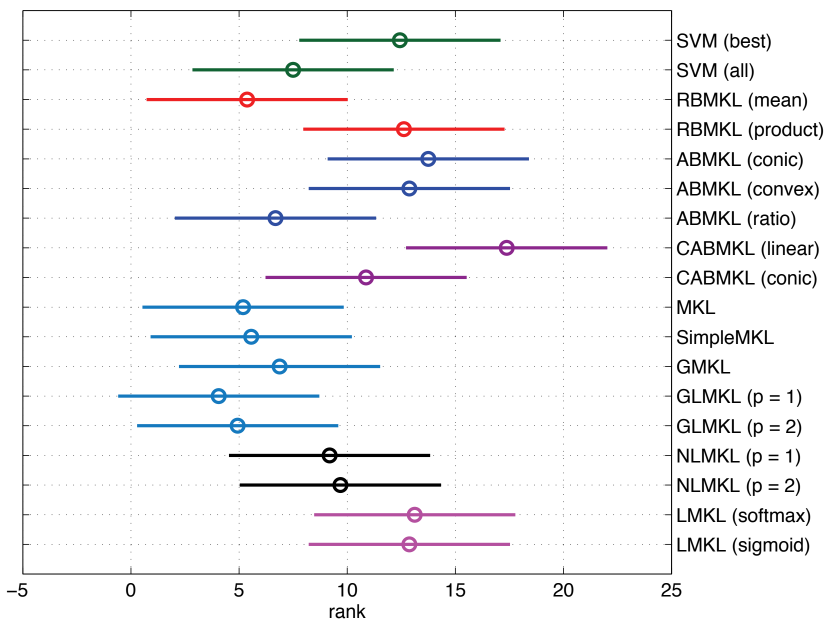


Figure 6: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of misclassification error using the Gaussian kernel.

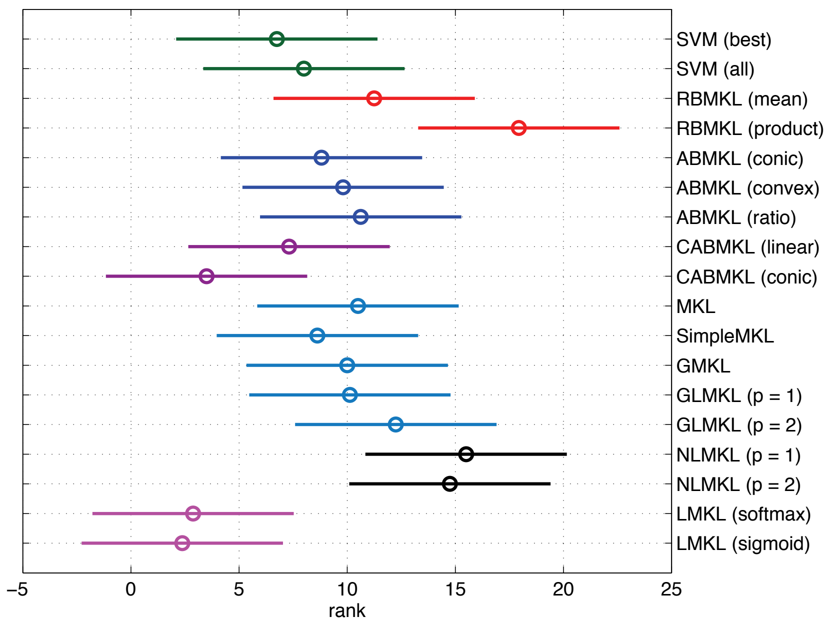


Figure 7: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of support vector percentages using the Gaussian kernel.

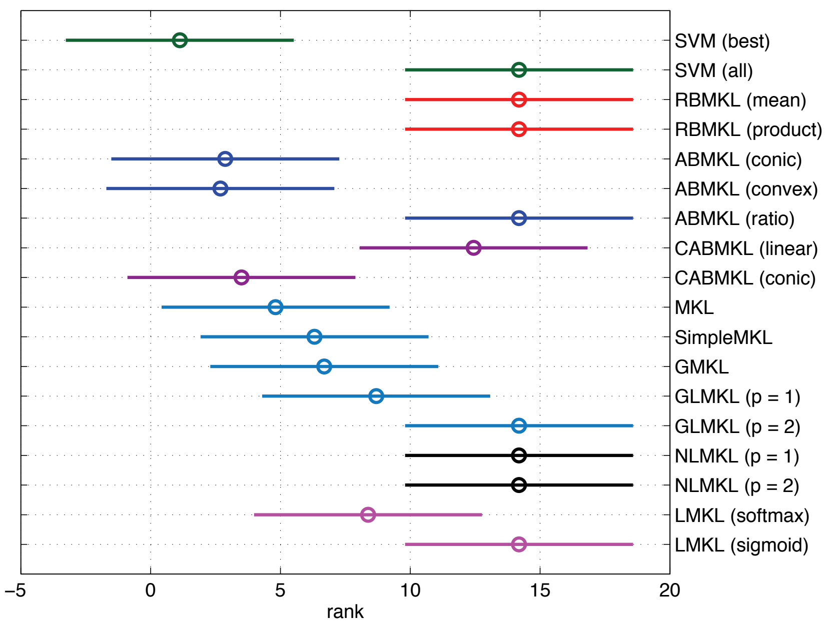


Figure 8: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of active kernel counts using the Gaussian kernel.

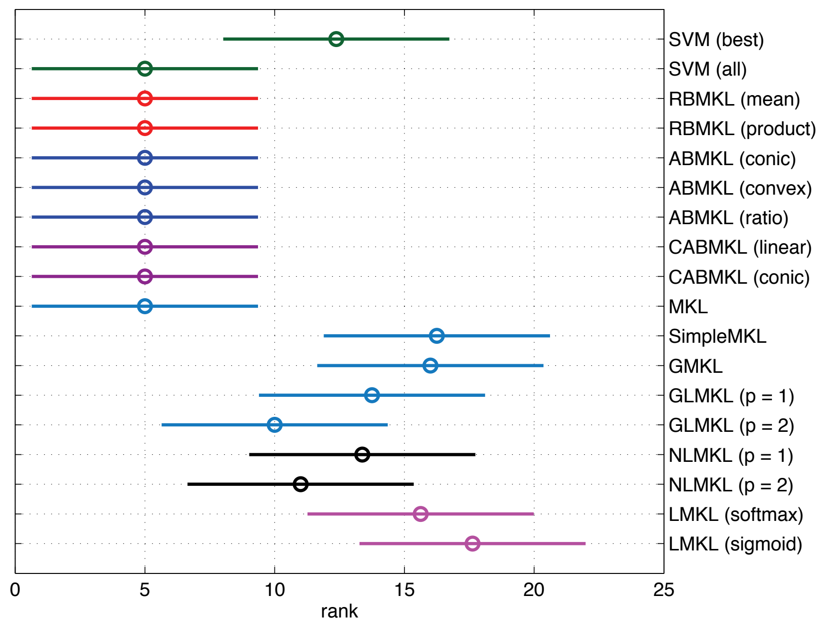


Figure 9: Overall comparison of single-kernel SVM and representative MKL algorithms in terms of optimization toolbox call counts using the Gaussian kernel.

We see that ABMKL (conic), ABMKL (convex), CABMKL (conic), MKL, SimpleMKL, GMKL, GLMKL ($p = 1$), and LMKL (softmax) use fewer kernels (statistically significantly in the case of the first three algorithms) than other combination algorithms.

Figure 9 shows the overall comparison between the algorithms in terms of the optimization toolbox call counts. Similar to the previous results obtained with simple linear kernels, the two-step algorithms need to solve more optimization problems than the other combination algorithms.

5. Conclusions

There is a significant amount of work on multiple kernel learning methods. This is because in many applications, one can come up with many possible kernel functions and instead of choosing one among them, we are interested in an algorithm that can automatically determine which ones are useful, which ones are not and therefore can be pruned, and combine the useful ones. Or, in some applications, we may have different sources of information coming from different modalities or corresponding to results from different experimental methodologies and each has its own (possibly multiple) kernel(s). In such a case, a good procedure for kernel combination implies a good combination of inputs from those multiple sources.

In this paper, we give a taxonomy of multiple kernel learning algorithms to best highlight the similarities and differences among the proposed algorithms in the literature, which we then review in detail. The dimensions we compare the existing MKL algorithms are the learning method, the functional form, the target function, the training method, the base learner, and the computational complexity. Then by looking at these dimensions, we form 12 groups of MKL variants to allow an organized discussion of the literature.

We also perform 10 experiments on four real data sets with simple linear kernels and eight experiments on three real data sets with complex Gaussian kernels comparing 16 MKL algorithms in practice. When combining simple linear kernels, in terms of accuracy, we see that using multiple kernels is better than using a single one but that in combination, trained linear combination is not always better than an untrained, unweighted combination and that nonlinear or data-dependent combination seem more promising. When combining complex Gaussian kernels, trained linear combination is better than nonlinear and data-dependent combinations but not than unweighted combination. Some MKL variants may be preferred because they use fewer support vectors or fewer kernels or need fewer calls to the optimizer during training. The relative importance of these criteria depend on the application at hand.

We conclude that multiple kernel learning is useful in practice and that there is ample evidence that better MKL algorithms can be devised for improved accuracy, decreased complexity and training time.

Acknowledgments

The authors would like to thank the editor and the three anonymous reviewers for their constructive comments, which significantly improved the presentation of the paper. This work was supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program under EA-TÜBA-GEBİP/2001-1-1, Boğaziçi University Scientific Research Project 07HA101 and the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant EEEAG

107E222. The work of M. Gönen was supported by the Ph.D. scholarship (2211) from TÜBİTAK. M. Gönen is currently at the Department of Information and Computer Science, Aalto University School of Science and the Helsinki Institute for Information Technology (HIIT), Finland.

Appendix A. List of Acronyms

GMKL	Generalized Multiple Kernel Learning
GP	Gaussian Process
KFDA	Kernel Fisher Discriminant Analysis
KL	Kullback-Leibler
KRR	Kernel Ridge Regression
LMKL	Localized Multiple Kernel Learning
LP	Linear Programming
MKL	Multiple Kernel Learning
QCQP	Quadratically Constrained Quadratic Programming
QP	Quadratic Programming
RKDA	Regularized Kernel Discriminant Analysis
SDP	Semidefinite Programming
SILP	Semi-infinite Linear Programming
SMKL	Sparse Multiple Kernel Learning
SOCP	Second-order Cone Programming
SVM	Support Vector Machine
SVR	Support Vector Regression

Appendix B. List of Notation

$\mathbb{R}$	Real numbers
$\mathbb{R}_+$	Nonnegative real numbers
$\mathbb{R}_{++}$	Positive real numbers
$\mathbb{R}^N$	Real $N \times 1$ matrices
$\mathbb{R}^{M \times N}$	Real $M \times N$ matrices
$\mathbb{S}^N$	Real symmetric $N \times N$ matrices
$\mathbb{N}$	Natural numbers
$\mathbb{Z}$	Integers
$\mathbb{Z}_+$	Nonnegative integers
$\|\|\mathbf{x}\|\|_p$	The ℓ_p-norm of vector $\mathbf{x}$
$\langle \mathbf{x}, \mathbf{y} \rangle$	Dot product between vectors $\mathbf{x}$ and $\mathbf{y}$
$k(\mathbf{x}, \mathbf{y})$	Kernel function between $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{K}$	Kernel matrix
$\mathbf{X}^\top$	Transpose of matrix $\mathbf{X}$
$\operatorname{tr}(\mathbf{X})$	Trace of matrix $\mathbf{X}$
$\|\|\mathbf{X}\|\|_F$	Frobenius norm of matrix $\mathbf{X}$
$\mathbf{X} \odot \mathbf{Y}$	Element-wise product between matrices $\mathbf{X}$ and $\mathbf{Y}$

References

Ethem Alpaydın. Combined 5×2 cv F test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892, 1999.

Andreas Argyriou, Charles A. Micchelli, and Massimiliano Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Proceeding of the 18th Conference on Learning Theory*, 2005.

Andreas Argyriou, Raphael Hauser, Charles A. Micchelli, and Massimiliano Pontil. A DC-programming algorithm for kernel selection. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Francis R. Bach. Consistency of the group Lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

Francis R. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems 21*, 2009.

Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl 1):i38–46, 2005.

Kristin P. Bennett, Michinari Momma, and Mark J. Embrechts. MARK: A boosting algorithm for heterogeneous kernel models. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

Jinbo Bi, Tong Zhang, and Kristin P. Bennett. Column-generation boosting methods for mixture of kernels. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

Olivier Bousquet and Daniel J. L. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems 15*, 2003.

Olivier Chapelle and Alain Rakotomamonjy. Second order optimization of kernel parameters. In *NIPS Workshop on Automatic Selection of Optimal Kernels*, 2008.

Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002.

Mario Christoudias, Raquel Urtasun, and Trevor Darrell. Bayesian localized multiple kernel learning. Technical Report UCB/EECS-2009-96, University of California at Berkeley, 2009.

Domenico Conforti and Rosita Guido. Kernel based support vector machine via semidefinite programming: Application to medical diagnosis. *Computers and Operations Research*, 37(8):1389–1394, 2010.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L_2 regularization for learning kernels. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

Corinna Cortes, Mehryar Mohri, and Rostamizadeh Afshin. Two-stage learning kernel algorithms. In *Proceedings of the 27th International Conference on Machine Learning*, 2010a.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems 22*, 2010b.

Koby Crammer, Joseph Keshet, and Yoram Singer. Kernel design using boosting. In *Advances in Neural Information Processing Systems 15*, 2003.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

Nello Cristianini, John Shawe-Taylor, Andree Elisseef, and Jaz Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, 2002.

Theodoros Damoulas and Mark A. Girolami. Probabilistic multi-class multi-kernel learning: On protein fold recognition and remote homology detection. *Bioinformatics*, 24(10):1264–1270, 2008.

Theodoros Damoulas and Mark A. Girolami. Combining feature spaces for classification. *Pattern Recognition*, 42(11):2671–2683, 2009a.

Theodoros Damoulas and Mark A. Girolami. Pattern recognition with a Bayesian kernel combination machine. *Pattern Recognition Letters*, 30(1):46–54, 2009b.

Tijl De Bie, Leon-Charles Tranchevent, Liesbeth M. M. van Oeffelen, and Yves Moreau. Kernel-based data fusion for gene prioritization. *Bioinformatics*, 23(13):i125–132, 2007.

Isaac Martín de Diego, Javier M. Moguerza, and Alberto Muñoz. Combining kernel information for support vector classification. In *Proceedings of the 4th International Workshop Multiple Classifier Systems*, 2004.

Isaac Martín de Diego, Alberto Muñoz, and Javier M. Moguerza. Methods for the combination of kernel matrices within a support vector framework. *Machine Learning*, 78(1–2):137–174, 2010a.

Isaac Martín de Diego, Ángel Serrano, Cristina Conde, and Enrique Cabello. Face verification with a kernel fusion method. *Pattern Recognition Letters*, 31:837–844, 2010b.

Réda Dehak, Najim Dehak, Patrick Kenny, and Pierre Dumouchel. Kernel combination for SVM speaker verification. In *Proceedings of the Speaker and Language Recognition Workshop*, 2008.

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

Glenn Fung, Murat Dundar, Jinbo Bi, and Bharat Rao. A fast iterative algorithm for Fisher discriminant using heterogeneous kernels. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Peter Vincent Gehler and Sebastian Nowozin. Infinite kernel learning. Technical report, Max Planck Institute for Biological Cybernetics, 2008.

Mark Girolami and Simon Rogers. Hierarchic Bayesian models for kernel learning. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

Mark Girolami and Mingjun Zhong. Data integration for classification problems employing Gaussian process priors. In *Advances in Neural Processing Systems 19*, 2007.

Mehmet Gönen and Ethem Alpaydın. Localized multiple kernel learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Yves Grandvalet and Stéphane Canu. Adaptive scaling for feature selection in SVMs. In *Advances in Neural Information Processing Systems 15*, 2003.

Junfeng He, Shih-Fu Chang, and Lexing Xie. Fast kernel learning for spatial pyramid matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

Mingqing Hu, Yiqiang Chen, and James Tin-Yau Kwok. Building sparse multiple-kernel SVM classifiers. *IEEE Transactions on Neural Networks*, 20(5):827–839, 2009.

Christian Igel, Tobias Glasmachers, Britta Mersch, Nico Pfeifer, and Peter Meinicke. Gradient-based optimization of kernel-target alignment for sequence kernels applied to bacterial gene start detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):216–226, 2007.

Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. Optimizing kernel alignment over combinations of kernels. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

Seung-Jean Kim, Alessandro Magnani, and Stephen Boyd. Optimal kernel selection in kernel Fisher discriminant analysis. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Marius Kloft, Ulf Brefeld, Sören Sonnenburg, Pavel Laskov, Klaus-Robert Müller, and Alexander Zien. Efficient and accurate ℓ_p-norm multiple kernel learning. In *Advances in Neural Information Processing Systems 22*, 2010a.

Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. Non-sparse regularization and efficient training with multiple kernels. Technical report, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2010b.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004a.

Gert R. G. Lanckriet, Tijl de Bie, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004b.

Gert R. G. Lanckriet, Minghua Deng, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. Kernel-based data fusion and its application to protein function prediction in Yeast. In *Proceedings of the Pacific Symposium on Biocomputing*, 2004c.

Wan-Jui Lee, Sergey Verzakov, and Robert P. W. Duin. Kernel combination versus classifier combination. In *Proceedings of the 7th International Workshop on Multiple Classifier Systems*, 2007.

Darrin P. Lewis, Tony Jebara, and William Stafford Noble. Support vector machine learning from heterogeneous data: An empirical analysis using protein sequence and structure. *Bioinformatics*, 22(22):2753–2760, 2006a.

Darrin P. Lewis, Tony Jebara, and William Stafford Noble. Nonstationary kernel combination. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006b.

Yen-Yu Lin, Tyng-Luh Liu, and Chiou-Shann Fuh. Dimensionality reduction for data in multiple feature representations. In *Advances in Neural Processing Systems 21*, 2009.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

Chris Longworth and Mark J. F. Gales. Multiple kernel learning for speaker verification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.

Chris Longworth and Mark J. F. Gales. Combining derivative and parametric kernels for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):748–757, 2009.

Brian McFee and Gert Lanckriet. Partial order embedding with multiple kernels. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

Javier M. Moguerza, Alberto Muñoz, and Isaac Martín de Diego. Improving support vector classification via the combination of multiple sources of information. In *Proceedings of the Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops*, 2004.

Mosek. *The MOSEK Optimization Tools Manual Version 6.0 (Revision 106)*. MOSEK ApS, Denmark, 2011.

Canh Hao Nguyen and Tu Bao Ho. An efficient kernel matrix evaluation measure. *Pattern Recognition*, 41(11):3366–3372, 2008.

William Stafford Noble. Support vector machine applications in computational biology. In Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert, editors, *Kernel Methods in Computational Biology*, chapter 3. The MIT Press, 2004.

Cheng Soon Ong and Alexander J. Smola. Machine learning using hyperkernels. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Hyperkernels. In *Advances in Neural Information Processing Systems 15*, 2003.

Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

Ayşegül Özen, Mehmet Gönen, Ethem Alpaydın, and Türkan Haliloğlu. Machine learning integration for predicting the effect of single amino acid substitutions on protein stability. *BMC Structural Biology*, 9(1):66, 2009.

Süreyya Özöğür-Akyüz and Gerhard Wilhelm Weber. Learning with infinitely many kernels via semi-infinite programming. In *Proceedings of Euro Mini Conference on Continuous Optimization and Knowledge-Based Technologies*, 2008.

Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the 5th Annual International Conference on Computational Molecular Biology*, 2001.

Shibin Qiu and Terran Lane. Multiple kernel learning for support vector regression. Technical report, Computer Science Department, University of New Mexico, 2005.

Shibin Qiu and Terran Lane. A framework for multiple kernel support vector regression and its applications to siRNA efficacy prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(2):190–199, 2009.

Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

Jagarlapudi Saketha Nath, Govindaraj Dinesh, Sankaran Raman, Chiranjib Bhattacharya, Aharon Ben-Tal, and Kalpathi R. Ramakrishnan. On the algorithmics and applications of a mixed-norm based kernel learning formulation. In *Advances in Neural Information Processing Systems 22*, 2010.

Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert, editors. *Kernel Methods in Computational Biology*. The MIT Press, 2004.

Sören Sonnenburg, Gunnar Rätsch, and Christin Schäfer. A general and efficient multiple kernel learning algorithm. In *Advances in Neural Information Processing Systems 18*, 2006a.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006b.

Niranjan Subrahmanya and Yung C. Shin. Sparse multiple kernel learning for signal processing applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):788–798, 2010.

Marie Szafranski, Yves Grandvalet, and Alain Rakotomamonjy. Composite kernel learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Marie Szafranski, Yves Grandvalet, and Alain Rakotomamonjy. Composite kernel learning. *Machine Learning*, 79(1–2):73–103, 2010.

Ying Tan and Jun Wang. A support vector machine with a hybrid kernel and minimal Vapnik-Chervonenkis dimension. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):385–395, 2004.

Hiroaki Tanabe, Tu Bao Ho, Canh Hao Nguyen, and Saori Kawasaki. Simple but effective methods for combining kernels in computational biology. In *Proceedings of IEEE International Conference on Research, Innovation and Vision for the Future*, 2008.

Ivor Wai-Hung Tsang and James Tin-Yau Kwok. Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks*, 17(1):48–58, 2006.

Koji Tsuda, Shinsuke Uda, Taishin Kin, and Kiyoshi Asai. Minimizing the cross validation error to mix kernel matrices of heterogeneous biological data. *Neural Processing Letters*, 19(1):63–72, 2004.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. John Wiley & Sons, 1998.

Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *Proceedings of the International Conference in Computer Vision*, 2007.

Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for SVMs. In *Advances in Neural Information Processing Systems 13*, 2001.

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

Mingrui Wu, Bernhard Schölkopf, and Gökhan Bakır. A direct method for building sparse kernel learning algorithms. *Journal of Machine Learning Research*, 7:603–624, 2006.

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in Neural Processing Systems 17*, 2005.

Zenglin Xu, Rong Jin, Irwin King, and Michael R. Lyu. An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems 21*, 2009a.

Zenglin Xu, Rong Jin, Jieping Ye, Michael R. Lyu, and Irwin King. Non-monotonic feature selection. In *Proceedings of the 26th International Conference on Machine Learning*, 2009b.

Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group Lasso. In *Proceedings of the 27th International Conference on Machine Learning*, 2010a.

Zenglin Xu, Rong Jin, Shenghuo Zhu, Michael R. Lyu, and Irwin King. Smooth optimization for effective multiple kernel learning. In *Proceedings of the 24th AAAI Conference on Artifical Intelligence*, 2010b.

Yoshihiro Yamanishi, Francis Bach, and Jean-Philippe Vert. Glycan classification with tree kernels. *Bioinformatics*, 23(10):1211–1216, 2007.

Fei Yan, Krystian Mikolajczyk, Josef Kittler, and Muhammad Tahir. A comparison of ℓ_1 norm and ℓ_2 norm multiple kernel SVMs in image and video classification. In *Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing*, 2009.

Jingjing Yang, Yuanning Li, Yonghong Tian, Ling-Yu Duan, and Wen Gao. Group-sensitive multiple kernel learning for object categorization. In *Proceedings of the 12th IEEE International Conference on Computer Vision*, 2009a.

Jingjing Yang, Yuanning Li, Yonghong Tian, Ling-Yu Duan, and Wen Gao. A new multiple kernel approach for visual concept learning. In *Proceedings of the 15th International Multimedia Modeling Conference*, 2009b.

Jingjing Yang, Yuanning Li, Yonghong Tian, Ling-Yu Duan, and Wen Gao. Per-sample multiple kernel approach for visual concept learning. *EURASIP Journal on Image and Video Processing*, 2010.

Jieping Ye, Jianhui Chen, and Shuiwang Ji. Discriminant kernel and regularization parameter learning via semidefinite programming. In *Proceedings of the 24th International Conference on Machine Learning*, 2007a.

Jieping Ye, Shuiwang Ji, and Jianhui Chen. Learning the kernel matrix in discriminant analysis via quadratically constrained quadratic programming. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007b.

Jieping Ye, Shuiwang Ji, and Jianhui Chen. Multi-class discriminant kernel learning via convex programming. *Journal of Machine Learning Research*, 9:719–758, 2008.

Yiming Ying, Kaizhu Huang, and Colin Campbell. Enhanced protein fold recognition through a novel data integration approach. *BMC Bioinformatics*, 10(1):267, 2009.

Bin Zhao, James T. Kwok, and Changshui Zhang. Multiple kernel clustering. In *Proceedings of the 9th SIAM International Conference on Data Mining*, 2009.

Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

Alexander Zien and Cheng Soon Ong. An automated combination of kernels for predicting protein subcellular localization. In *Proceedings of the 8th International Workshop on Algorithms in Bioinformatics*, 2008.

Journal of Machine Learning Research 12 (2011) 2269-2292 Submitted 6/10; Revised 4/11; Published 7/11

Smoothness, Disagreement Coefficient, and the Label Complexity of Agnostic Active Learning

Liwei Wang WANGLW@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception, MOE
School of Electronics Engineering and Computer Science
Peking University
Beijing, 100871, P.R.China

Editor: Rocco Servedio

Abstract

We study pool-based active learning in the presence of noise, that is, the agnostic setting. It is known that the effectiveness of agnostic active learning depends on the learning problem and the hypothesis space. Although there are many cases on which active learning is very useful, it is also easy to construct examples that no active learning algorithm can have an advantage. Previous works have shown that the label complexity of active learning relies on the *disagreement coefficient* which often characterizes the intrinsic difficulty of the learning problem. In this paper, we study the disagreement coefficient of classification problems for which the classification boundary is smooth and the data distribution has a density that can be bounded by a smooth function. We prove upper and lower bounds for the disagreement coefficients of both finitely and infinitely smooth problems. Combining with existing results, it shows that active learning is superior to passive supervised learning for smooth problems.

Keywords: active learning, disagreement coefficient, label complexity, smooth function

1. Introduction

Active learning addresses the problem that the algorithm is given a pool of unlabeled data drawn i.i.d. from some underlying distribution; the algorithm can then pay for the label of any example in the pool. The goal is to learn an accurate classifier by requesting as few labels as possible. This is in contrast with the standard passive supervised learning, where the labeled examples are chosen randomly.

The simplest example that demonstrates the potential of active learning is to learn the optimal threshold on an interval. Suppose the instances are uniformly distributed on $[0,1]$, and there exists a perfect threshold separating the two classes (i.e., there is no noise), then binary search needs $O(\log\frac{1}{\varepsilon})$ labels to learn an ε-accurate classifier, while passive learning requires $O(\frac{1}{\varepsilon})$ labels. Another encouraging example is to learn homogeneous linear separators. If the data are distributed on the unit sphere of $\mathbb{R}^d$, and the distribution has a density function upper and lower bounded by λ and $1/\lambda$ respectively, where λ is some constant, then active learning can still give exponential savings in the label complexity (Dasgupta, 2005).

However, there are also very simple problems that active learning does not help. Suppose again that the instances are uniformly distributed on $[0,1]$. But this time the positive class could be any interval on $[0,1]$. In this case, for any active learning algorithm there exists a distribution (i.e., a target classifier) such that the algorithm needs $\Omega(\frac{1}{\varepsilon})$ label requests to learn an ε-accurate classifier (Dasgupta, 2005). Thus there is no improvement over passive learning in the minimax sense. All above are realizable problems. Of more interest and more realistic is the agnostic setting, where the best classifier in the hypothesis space has a non-zero error ν. For agnostic active learning, there is no active learning algorithm that can always reduce label requests due to a lower bound $\Omega(\frac{\nu^2}{\varepsilon^2})$ for the label complexity (Kääriäinen, 2006).

Previous results have shown that whether active learning helps relies crucially on the *disagreement coefficient* of the learning problem (Hanneke, 2007). The disagreement coefficient depends on the distribution of the instance-label pairs and the hypothesis space and often describes the intrinsic difficulty of the active learning problem. In particular, it has been shown that the label complexity of two important agnostic active learning algorithms A^2 (Balcan et al., 2006) and the one due to Dasgupta et al. (2007) (will be referred to as DHM) are characterized by the disagreement coefficient. If the disagreement coefficient is small, active learning usually has smaller label complexity than passive learning.

In this paper, we study the disagreement coefficient for smooth problems. Specifically we analyze the disagreement coefficient for learning problems whose classification boundaries are smooth. Such problems are often referred to as the boundary fragment class (van der Vaart and Wellner, 1996). Under some mild assumptions on the distribution, we show that the magnitude of the disagreement coefficient depends on the order of smoothness. For finite order smoothness, it is polynomially smaller than the largest possible value, and exponentially smaller for infinite smoothness. Combining with known upper bounds on the label complexity in terms of disagreement coefficient, we give sufficient condition under which active learning is strictly superior to passive learning.

1.1 Related Works

Our work is closely related to Castro and Nowak (2008) which proved label complexity bounds for problems with smooth classification boundary under Tsybakov's noise condition (Tsybakov, 2004). Please see Section 3.3 for a detailed discussion on this work.

Another related work is due to Friedman (2009). He introduced a different notion of smoothness. In particular, he considered smooth problems whose hypothesis space is a finite dimensional parametric space (and therefore has finite VC dimension). He gave conditions under which the disagreement coefficient is always bounded from above by a constant. In contrast, the hypothesis space (the boundary fragment class) studied in our work is a nonparametric class and is more expressive than VC classes.

2. Background

Let $\mathcal{X}$ be an instance space, $\mathcal{D}$ a distribution over $\mathcal{X} \times \{-1,1\}$. Let $\mathcal{H}$ be the hypothesis space, a set of classifiers from $\mathcal{X}$ to $\{-1,1\}$. Denote $\mathcal{D}_X$ the marginal of $\mathcal{D}$ over $\mathcal{X}$. In our active learning model, the algorithm has access to a pool of unlabeled examples from $\mathcal{D}_X$. For any unlabeled point x, the algorithm can ask for its label y, which is generated from the conditional distribution at x. The error of a hypothesis h according to $\mathcal{D}$ is $er_{\mathcal{D}}(h) = \Pr_{(x,y)\sim\mathcal{D}}(h(x) \neq y)$. The empirical error on a sample $\mathcal{S}$ of size n is $er_{\mathcal{S}}(h) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}} \mathbb{I}[h(x) \neq y]$, where $\mathbb{I}$ is the indicator function. We use h^* denote the best classifier in $\mathcal{H}$. That is, $h^* = \arg\min_{h\in\mathcal{H}} er_{\mathcal{D}}(h)$. Let $\nu = er_{\mathcal{D}}(h^*)$. Our goal is to learn a $\hat{h} \in \mathcal{H}$ with error rate at most $\nu + \varepsilon$, where ε is the desired accuracy.

```
Input: unlabeled data pool $(x_1, x_2, \ldots, x_m)$ i.i.d. from $\mathcal{D}_X$, hypothesis space $\mathcal{H}$;
Initially: $V \leftarrow \mathcal{H}, R \leftarrow DIS(V), Q \leftarrow \emptyset$;
for $t = 1, 2, \ldots, m$ do
    if $\Pr(DIS(V)) \le \frac{1}{2}\Pr(R)$ then
        $R \leftarrow DIS(V); Q \leftarrow \emptyset$;
    end
    Find a new data $x_i$ from the data pool with $x_i$ in $R$;
    Request the label $y_i$ of $x_i$, and let $Q \leftarrow Q \cup \{(x_i, y_i)\}$;
    $V \leftarrow \{h \in V : LB(h, Q, \delta/m) \le \min_{h' \in V} UB(h', Q, \delta/m)\}$;
    $h_t \leftarrow \arg\min_{h \in V} er_Q(h)$;
    $\beta_t \leftarrow (UB(h_t, Q, \delta/m) - LB(h_t, Q, \delta/m)) \Pr(R)$ ;
end
Return $\hat{h} = h_j$, where $j = \arg\min_{t \in \{1,2,\ldots,m\}} \beta_t$.
```

Algorithm 1: The A^2 algorithm

A^2 (Balcan et al., 2006) is the first rigorous agnostic active learning algorithm. It can be viewed as a robust version of the active learning algorithm due to Cohn et al. (1994) for the realizable setting. A description of the algorithm is given in Algorithm 1. It was shown that A^2 is never much worse than passive learning in terms of the label complexity. The key observation that A^2 can be superior to passive learning is that, since our goal is to choose an $\hat{h}$ such that $er_{\mathcal{D}}(\hat{h}) \le er_{\mathcal{D}}(h^*) + \varepsilon$, we only need to *compare* the errors of hypotheses. Therefore we can just request labels of those x on which the hypotheses under consideration have disagreement.

To do this, the algorithm keeps track of two spaces. One is the current version space V, consisting of hypotheses that with statistical confidence are not too bad compared to h^*; the other is the region of disagreement $DIS(V)$, which is the set of all $x \in \mathcal{X}$ for which there are hypotheses in V that disagree on x. Formally, for any subset $V \subset \mathcal{H}$,

$$DIS(V) = \{x \in \mathcal{X} : \exists h, h' \in V, \;\; h(x) \ne h'(x)\}.$$

To achieve the statistical guarantee that the version space V contains only good hypotheses, the algorithm must be provided with a uniform convergence bound over the hypothesis space. That is, with probability at least $1 - \delta$ over the draw of sample $\mathcal{S}$ according to $\mathcal{D}$ conditioned on $DIS(V)$ for any version spaces V,

$$LB(\mathcal{S}, h, \delta) \le er_{\mathcal{D}_{|V}}(h) \le UB(\mathcal{S}, h, \delta),$$

hold simultaneously for all $h \in \mathcal{H}$, where the lower bound $LB(\mathcal{S}, h, \delta)$ and upper bound $UB(\mathcal{S}, h, \delta)$ can be computed from the empirical error $er_{\mathcal{S}}(h)$. Here $\mathcal{D}_{|V}$ is the distribution of $\mathcal{D}$ conditioned on $DIS(V)$. If $\mathcal{H}$ has finite VC dimension $VC(\mathcal{H})$, then $er_{\mathcal{S}}(h) \pm O(\frac{VC(\mathcal{H})}{n})^{-1/2}$ are upper and lower bounds of $er_{\mathcal{D}_{|V}}(h)$ respectively.

We will denote the volume of $DIS(V)$ by $\Delta(V) = \Pr_{X \sim \mathcal{D}_X}(X \in DIS(V))$. Requesting labels of the instances from $DIS(V)$ rather than from the whole space $\mathcal{X}$ allows A^2 require fewer labels than passive learning. Hence the key issue is how fast $\Delta(V)$ reduces. This process, and in turn the label complexity of A^2, are nicely characterized by the disagreement coefficient θ introduced in Hanneke (2007).

```
Input: unlabeled data pool $(x_1, x_2, \ldots, x_m)$ i.i.d. from $\mathcal{D}_X$, hypothesis space $\mathcal{H}$;
Initially: $\mathcal{G}_0 \leftarrow \emptyset$, $\mathcal{T}_0 \leftarrow \emptyset$;
for $t = 1, 2, \ldots, m$ do
    For each $\hat{y} \in \{-1, 1\}$, $h_{\hat{y}} \leftarrow \text{LEARN}_{\mathcal{H}}(\mathcal{G}_{t-1} \cup \{(x_t, \hat{y})\}, \mathcal{T}_{t-1})$;
    if $er_{\mathcal{G}_{t-1} \cup \mathcal{T}_{t-1}}(h_{-\hat{y}}) - er_{\mathcal{G}_{t-1} \cup \mathcal{T}_{t-1}}(h_{\hat{y}}) > \Delta_{t-1}$ for some $\hat{y} \in \{-1, 1\}$ then
        $\mathcal{G}_t \leftarrow \mathcal{G}_{t-1} \cup \{(x_t, \hat{y})\}$; $\mathcal{T}_t \leftarrow \mathcal{T}_{t-1}$;
    end
    else
        Request the true label $y_t$ of $x_t$; $\mathcal{G}_t \leftarrow \mathcal{G}_{t-1}$; $\mathcal{T}_t \leftarrow \mathcal{T}_{t-1} \cup \{(x_t, y_t)\}$;
    end
end
Return $h = \text{LEARN}_{\mathcal{H}}(\mathcal{G}_m, \mathcal{T}_m)$.
```

Algorithm 2: The DHM algorithm

Definition 1 *Let $\rho(\cdot, \cdot)$ be the pseudo-metric on a hypothesis space $\mathcal{H}$ induced by $\mathcal{D}_X$. That is, for $h, h' \in \mathcal{H}$, $\rho(h, h') = \Pr_{X \sim \mathcal{D}_X}(h(X) \neq h'(X))$. Let $B(h, r) = \{h' \in \mathcal{H} : \rho(h, h') \leq r\}$. The disagreement coefficient $\theta(\varepsilon)$ is*

$$\theta(\varepsilon) = \sup_{r \geq \varepsilon} \frac{\Delta(B(h^*, r))}{r} = \sup_{r \geq \varepsilon} \frac{\Pr_{X \sim \mathcal{D}_X}(X \in DIS(B(h^*, r)))}{r},$$

where $h^* = \arg\min_{h \in \mathcal{H}} er_{\mathcal{D}}(h)$.

Note that θ depends on $\mathcal{H}$ and $\mathcal{D}$, and $1 \leq \theta(\varepsilon) \leq \frac{1}{\varepsilon}$.[1] The following is an upper bound of the label complexity of A^2 in terms of the disagreement coefficient $\theta(\varepsilon)$ (Hanneke, 2007).

Theorem 2 *Suppose that $\mathcal{H}$ has finite VC dimension $VC(\mathcal{H})$. Then using the definitions given above, the label complexity of A^2 is*

$$O\left((\theta(\nu + \varepsilon))^2 \left(\frac{\nu^2}{\varepsilon^2} + 1\right) \text{polylog}\left(\frac{1}{\varepsilon}\right) \log\left(\frac{1}{\delta}\right) VC(\mathcal{H})\right). \tag{1}$$

In addition, Hanneke (2007) showed that $\tilde{\Omega}(\theta^2 \log \frac{1}{\delta})$ is a lower bound for the A^2 algorithm of any problem with $\nu = 0$, where in $\tilde{\Omega}$ we hide the logrithm terms.

Another important agnostic active learning algorithm is DHM. (Algorithm 2 gives a formal description of the algorithm.) DHM reduces active learning to a series of *constrained* supervised learning. The key idea of the algorithm is that each time we encounter a new unlabeled data x, we test if we can guess the label of x with high confidence, using the information obtained so far. If we can, we put the data and the confidently guessed label $(x, \hat{y})$ into the *guessed* set $\mathcal{G}$; otherwise, we request the true label y of x and put (x, y) into the *true* set $\mathcal{T}$. The criterion of whether we can guess the label of x confidently is as follows. For each $\tilde{y} \in \{-1, +1\}$, we learn a classifier $h_{\tilde{y}} \in \mathcal{H}$ such that $h_{\tilde{y}}(x) = \tilde{y}$, and $h_{\tilde{y}}$ is consistent with all $(x, \hat{y}) \in \mathcal{G}$ and has minimal error on $\mathcal{T}$. (This is the subroutine LEARN in Algorithm 2.) If for some $\tilde{y} \in \{-1, +1\}$ the error rate of $h_{\tilde{y}}$ is smaller than

1. Here we only consider the nontrivial case that $\Delta(B(h^*, r)) \geq r$ for all r. This condition is satisfied by the smooth problems studied in this paper.

that of $h_{-\tilde{y}}$ by a threshold Δ_t (t is the number of unlabeled data encountered so far), then we guess that $\hat{y} = \tilde{y}$ confidently.

The algorithm DHM relies crucially on a good choice of the threshold function Δ_t. If $\mathcal{H}$ has finite VC dimension $VC(\mathcal{H})$, Dasgupta et al. (2007) suggested to choose Δ_t based on the normalized uniform convergence bound on $\mathcal{H}$ (Vapnik, 1998). They also showed that DHM is never much worse than passive learning and it has label complexity[2]

$$\tilde{O}\left(\theta(\nu+\varepsilon)\left(1+\frac{\nu^2}{\varepsilon^2}\right)\text{polylog}\left(\frac{1}{\varepsilon}\right)\log\left(\frac{1}{\delta}\right)VC(\mathcal{H})\right). \tag{2}$$

From (1) and (2) it can be seen that if $\varepsilon > \nu$, the term $\frac{\nu^2}{\varepsilon^2}$ is upper bounded by 1 and the label complexity of the active learning algorithms crucially depends on the disagreement coefficient θ. However, the asymptotic label complexity as ε tends to 0 (assuming $\nu > 0$) can at best only be upper bounded by $O\left(\frac{\nu^2}{\varepsilon^2}\right)$. In fact, this bound cannot be improved: it is known that given some hypothesis space $\mathcal{H}$, for every active learning algorithm A, there is a learning problem (to be concrete, a h^*) such that the label complexity of A is at least $\Omega(\frac{\nu^2}{\varepsilon^2})$ (Kääriäinen, 2006). Thus $\Omega(\frac{\nu^2}{\varepsilon^2})$ is a minimax lower bound of the label complexity of agnostic active learning algorithms.

Although no active learning algorithm is superior to passive learning in all agnostic settings, it turns out that if the disagreement coefficient is small, active learning does always help under a finer parametrization of the noise distribution, known as Tsybakov's noise condition (Tsybakov, 2004).

Definition 3 *Let $\eta(x) = \Pr(Y = 1|X = x)$. We say that the distribution of the learning problem has noise exponent $\kappa = \frac{a+1}{a}$ ($\kappa \geq 1$) if there exists constant $c > 0$ such that*

$$\Pr\left(\left|\eta(X) - \frac{1}{2}\right| \leq t\right) \leq ct^a, \qquad 0 < a \leq +\infty$$

for all $0 < t \leq t_0$ for some constant t_0.

Tsybakov's noise condition characterizes the behavior of $\eta(x)$ when x crosses the class boundary. If $\kappa = 1$, $\eta(x)$ has a jump from $\frac{1}{2} - t_0$ to $\frac{1}{2} + t_0$. The larger the κ, the more "flat" $\eta(x)$ is.

Under Tsybakov's noise condition, Hanneke (2009, 2011) proved that a variant of the DHM algorithm (by choosing the threshold Δ_t based on local Rademacher complexity (Koltchinskii, 2006)) has the following asymptotic label complexity.

Theorem 4 *Suppose that the learning problem satisfies the Tsybakov's noise condition with noise exponent κ. Assume that the hypothesis space $\mathcal{H}$ and the marginal distribution $\mathcal{D}_X$ satisfies that the entropy with bracketing $H_{[\,]}(\varepsilon, \mathcal{H}, L_2(\mathcal{D}_X)) = O\left(\left(\frac{1}{\varepsilon}\right)^{2p}\right)$ for some $0 < p < 1$. If the Bayes classifier $h_B^* \in \mathcal{H}$, then the label complexity of DHM is*

$$O\left(\theta(\varepsilon_0)\left(\frac{1}{\varepsilon}\right)^{2-\frac{2-p}{\kappa}}\left(\log\frac{1}{\varepsilon} + \log\frac{1}{\delta}\right)\right), \tag{3}$$

where ε_0 depends on ε, κ, p, δ and the learning problem. In particular, setting $\varepsilon_0 = \varepsilon^{\frac{1}{\kappa}}$ the theorem holds.

2. Here in $\tilde{O}$ we hide terms like $\log\log(\frac{1}{\varepsilon})$ and $\log\log(\frac{1}{\delta})$.

Inspired by this result, Koltchinskii (2010) further proved that under similar conditions a variant of the A^2 algorithm has label complexity

$$O\left(\theta(\varepsilon^{\frac{1}{\kappa}})\left(\left(\frac{1}{\varepsilon}\right)^{2-\frac{2-p}{\kappa}}+\left(\frac{1}{\varepsilon}\right)^{2-\frac{2}{\kappa}}\left(\log\frac{1}{\delta}+\log\log\frac{1}{\varepsilon}\right)\right)\right). \tag{4}$$

Note that in the last formula $\left(\frac{1}{\varepsilon}\right)^{2-\frac{2-p}{\kappa}}$ dominates over $\left(\frac{1}{\varepsilon}\right)^{2-\frac{2}{\kappa}}$ as $\varepsilon \to 0$ if $p > 0$.

If the hypothesis space $\mathcal{H}$ has finite VC dimension, the entropy with bracketing is

$$H_{[\,]}(\varepsilon,\mathcal{H},L_2(\mathcal{D}_X)) = O\left(\log\frac{1}{\varepsilon}\right),$$

smaller than $\left(\frac{1}{\varepsilon}\right)^{2p}$ for any $p > 0$. In this case, it can be shown that the above label complexity bounds still hold by just putting $p = 0$ into them.

In contrast, the sample complexity for passive learning under the same conditions is known to be (Tsybakov, 2004)

$$O\left(\left(\frac{1}{\varepsilon}\right)^{2-\frac{1-p}{\kappa}}\left(\log\frac{1}{\delta}+\log\log\frac{1}{\varepsilon}\right)\right), \tag{5}$$

and it is also a minimax lower bound. Comparing (3), (4) and (5) one can see that whether active learning is strictly superior to passive learning entirely depends on how small the disagreement coefficient $\theta(\varepsilon)$ is.

One shortcoming of A^2 and DHM is that they are computationally expensive. This is partially because that they need to minimize the 0-1 loss and need to maintain the version space. Beygelzimer et al. (2009) proposed an importance weighting procedure IWAL which, during learning, minimize a convex surrogate loss and therefore avoid 0-1 minimization. Furthermore, Beygelzimer et al. (2010) developed an active learning algorithm which does not need to keep the version space and therefore is computationally efficient. There are also upper bounds on the label complexity of these two algorithms in terms of the disagreement coefficient.

Finally, for a comprehensive survey of the theoretical research on active learning, please see the excellent tutorial (Dasgupta and Langford, 2009).

3. Main Results

As described in the previous section, whether active learning helps largely depends on the disagreement coefficient which often characterizes the intrinsic difficulty of the learning problem using a given hypothesis space. So it is important to understand if the disagreement coefficient is small for learning problems with practical and theoretical interests. In this section we give bounds on the disagreement coefficient for problems that have smooth classification boundaries, under additional assumptions on the distribution. Such smooth problems are often referred to as boundary fragment class and has been extensively studied in passive learning and especially in empirical processes.

In Section 3.1 we give formal definitions of the smooth problems. Section 3.2 contains the main results, where we establish upper and lower bounds for the disagreement coefficient of smooth problems. In Section 3.3 we provide some discussions on some closely related works.

3.1 Smoothness

Let f be a function defined on $\Omega \subset \mathbb{R}^d$ and $\alpha > 0$ be a real number. Let $\underline{\alpha}$ be the largest integer strictly smaller than α. (Hence $\underline{\alpha} = \alpha - 1$ when α is an integer.) For any vector $\mathbf{k} = (k_1, \cdots, k_d)$ of d nonnegative integers, let $|\mathbf{k}| = \sum_{i=1}^d k_i$. Let

$$D^{\mathbf{k}} = \frac{\partial^{|\mathbf{k}|}}{\partial^{k_1} x^1 \cdots \partial^{k_d} x^d},$$

be the differential operator. Define the α-norm as (van der Vaart and Wellner, 1996)

$$\|f\|_\alpha := \max_{|\mathbf{k}| \le \underline{\alpha}} \sup_x |D^{\mathbf{k}} f(x)| + \max_{|\mathbf{k}| = \underline{\alpha}} \sup_{x,x'} \frac{|D^{\mathbf{k}} f(x) - D^{\mathbf{k}} f(x')|}{\|x - x'\|^{\alpha - \underline{\alpha}}},$$

where the suprema are taken over all x, x' over Ω with $x \neq x'$.

Definition 5 *(**Finite Smooth Functions**) A function f is said to be αth order smooth with respect to a constant C, if $\|f\|_\alpha \le C$. The set of αth order smooth functions is defined as*

$$F_C^\alpha := \{f : \|f\|_\alpha \le C\}.$$

Thus αth order smooth functions have uniformly bounded partial derivatives up to order $\underline{\alpha}$, and the $\underline{\alpha}$th order partial derivatives are Hölder continuous. As a special case, note that if f has continuous partial derivatives upper bounded by C up to order m, where m is any positive integer, then $f \in F_C^m$. Also, if $0 < \beta < \alpha$, then $f \in F_C^\alpha$ implies $f \in F_C^\beta$.

Definition 6 *(**Infinitely Smooth Functions**) A function f is said to be infinitely smooth with respect to a constant C, if $\|f\|_\alpha \le C$ for all $\alpha > 0$. The set of infinitely smooth functions is denoted by F_C^∞.*

With the definitions of smoothness, we introduce the hypothesis space we use in active learning algorithms.

Definition 7 *(**Hypotheses with Smooth Classification Boundaries**) A set of hypotheses $\mathcal{H}_C^\alpha$ defined on $\mathcal{X} = [0,1]^{d+1}$ is said to have αth order smooth classification boundaries, if for every $h \in \mathcal{H}_C^\alpha$, the classification boundary is a αth order smooth function on $[0,1]^d$. To be precise, let $\mathbf{x} = (x^1, x^2, \ldots, x^{d+1}) \in [0,1]^{d+1}$. The classification boundary is the graph of function $x^{d+1} = f(x^1, \ldots, x^d)$, where $f \in F_C^\alpha$. Similarly, a hypothesis space $\mathcal{H}_C^\infty$ is said to have infinitely smooth boundaries, if for every $h \in \mathcal{H}_C^\infty$ the classification boundary is the graph an infinitely smooth function on $[0,1]^d$.*

The first thing we need to guarantee is that smooth problems are learnable, both passively and actively. To be concrete, we must show that the entropy with bracketing of smooth problems satisfies

$$H_{[\,]}(\epsilon, \mathcal{H}, L_2(\mathcal{D}_X)) = O\left(\left(\frac{1}{\epsilon}\right)^{2p}\right),$$

for some $p < 1$ (van der Vaart and Wellner, 1996) (see also Theorem 4). For smooth problems, the following proposition is known.

Proposition 8 *(van der Vaart and Wellner, 1996)*

Let the instance space be $[0,1]^{d+1}$ and the hypothesis space be $\mathcal{H}_C^\alpha$. Assume that the marginal distribution $\mathcal{D}_X$ has a density upper bounded by a constant. Then

$$H_{[\,]}(\varepsilon, \mathcal{H}_C^\alpha, L_2(\mathcal{D}_X)) = O\left(\left(\frac{1}{\varepsilon}\right)^{\frac{2d}{\alpha}}\right).$$

The problem is learnable if $\alpha > d$.

In the rest of this paper, we only consider smooth problems such that $\alpha > d$.

3.2 Disagreement Coefficient

The disagreement coefficient θ plays an important role for the label complexity of active learning algorithms. In fact previous negative examples for which active learning does not work are all because of large θ. For instance the interval learning problem, $\theta(\varepsilon) = \frac{1}{\varepsilon}$, which leads to the same label complexity as passive learning. (Recall that $\theta(\varepsilon) \le \frac{1}{\varepsilon}$, so this is the worst case.)

In this section we will show that that the disagreement coefficient $\theta(\varepsilon)$ for smooth problems is small. Especially, we establish both upper bounds (Theorem 9 and Theorem 10) and lower bounds (Theorem 13) for the disagreement coefficient of smooth problems. Finally we will combine our upper bounds on the disagreement coefficient with the label complexity result of Theorem 4 and show that active learning is strictly superior to passive learning for smooth problems.

Theorem 9 *Let the instance space be $\mathcal{X} = [0,1]^{d+1}$. Let the hypothesis space be $\mathcal{H}_C^\alpha$, where $d < \alpha < \infty$. If the marginal distribution $\mathcal{D}_X$ has a density $p(\mathbf{x})$ on $[0,1]^{d+1}$ such that there exists an αth order smooth function $g(\mathbf{x})$ and two constants $0 < a \le b$ such that $ag(\mathbf{x}) \le p(\mathbf{x}) \le bg(\mathbf{x})$ for all $\mathbf{x} \in [0,1]^{d+1}$, then*[3]

$$\theta(\varepsilon) = O\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}}\right).$$

The key points in the theorem are: the classification boundaries are smooth; and the density is bounded from above and below by constants times a smooth function.[4] Note that the density itself is not necessarily smooth. We merely require the density does not change too rapidly.

The intuition behind the theorem above is as follows. Let $f_{h^*}(x)$ and $f_h(x)$ be the classification boundaries of h^* and h, and suppose $\rho(h,h^*)$ is small, where $\rho(h,h^*) = \Pr_{x\sim\mathcal{D}_X}(h(x) \neq h^*(x))$ is the pseudo metric. If the classification boundaries and the density are all smooth, then the two boundaries have to be close to each other everywhere. That is, $|f_h(x) - f_{f^*}(x)|$ is small uniformly for all x. Hence only the points close to the classification boundary of h^* can be in $DIS(B(h^*,\varepsilon))$, which leads to a small disagreement coefficient.

For infinitely smooth problems, we have the following theorem. Note that the requirement on the density is stronger than finite smoothness problems.

3. This upper bound was obtained with the help of Yanqi Dai, Kai Fan, Chicheng Zhang and Ziteng Wang. It improves a previous upper bound $O\left(\left(\frac{1}{\varepsilon}\right)^{1-\frac{\alpha^d}{(1+\alpha)^d}}\right)$, which converges to the current bound as $\frac{\alpha}{d} \to \infty$.

4. These two conditions include a large class of learning problems. For example, the boundary fragment class equipped with most elementary distributions (truncated in $[0,1]^{d+1}$) satisfies these conditions.

Theorem 10 *Let the hypothesis space be $\mathcal{H}_C^\infty$. If the distribution $\mathcal{D}_X$ has a density $p(\mathbf{x})$ such that there exist two constants $0 < a \le b$ such that $a \le p(\mathbf{x}) \le b$ for all $\mathbf{x} \in [0,1]^{d+1}$, then $\theta(\varepsilon) = O(\log^{2d}(\frac{1}{\varepsilon}))$.*

The proofs of Theorem 9 and Theorem 10 rely on the following two lemmas.

Lemma 11 *Let Φ be a function defined on $[0,1]^d$ and is αth order smooth. If*

$$\int_{[0,1]^d} |\Phi(x)| dx \le r,$$

then

$$\|\Phi\|_\infty = O\left(r^{\frac{\alpha}{\alpha+d}}\right) = O\left(r \cdot \left(\frac{1}{r}\right)^{\frac{d}{\alpha+d}}\right),$$

where $\|\Phi\|_\infty = \sup_{x\in[0,1]^d} |\Phi(x)|$.

Lemma 12 *Let Φ be a function defined on $[0,1]^d$ and is infinitely smooth. If*

$$\int_{[0,1]^d} |\Phi(x)| dx \le r,$$

then

$$\|\Phi\|_\infty = O\left(r \cdot \left(\log \frac{1}{r}\right)^{2d}\right).$$

Proof of Theorem 9 First of all, since we focus on binary classification, $DIS(B(h^*,r))$ can be written equivalently as

$$DIS(B(h^*,r)) = \{x \in \mathcal{X},\ \exists h \in B(h^*,r),\ s.t.\ h(x) \neq h^*(x)\}.$$

Consider any $h \in B(h^*,r)$. Let $f_h, f_{h^*} \in F_C^\alpha$ be the corresponding classification boundaries of h and h^* respectively. If r is sufficiently small, we must have

$$\rho(h,h^*) = \Pr_{X\sim\mathcal{D}_X}(h(X) \neq h^*(X)) = \int\limits_{[0,1]^d} dx^1 \dots dx^d \left| \int_{f_{h^*}(x^1,\dots,x^d)}^{f_h(x^1,\dots,x^d)} p(x^1,\dots,x^{d+1}) dx^{d+1} \right|.$$

Denote

$$\Phi_h(x^1,\dots,x^d) = \int_{f_{h^*}(x^1,\dots,x^d)}^{f_h(x^1,\dots,x^d)} p(x^1,\dots,x^{d+1}) dx^{d+1}.$$

We assert that there is a αth order smooth function $\tilde{\Phi}_h(x^1,\dots,x^d)$ and two constants $0 < a \le b$ such that $a|\tilde{\Phi}_h| \le |\Phi_h| \le b|\tilde{\Phi}_h|$. To see this, remember that f_h and f_{h^*} are αth order smooth functions; and the density p is upper and lower bounded by constants times a αth order smooth function $g(x^1,\dots,x^{d+1})$. Also note that if we define

$$\tilde{\Phi}_h(x^1,\dots,x^d) = \int_{f_{h^*}(x^1,\dots,x^d)}^{f_h(x^1,\dots,x^d)} g(x^1,\dots,x^{d+1}) dx^{d+1},$$

$\tilde{\Phi}_h$ is a αth order smooth function, which is easy to check by taking derivatives. Now

$$\int_{[0,1]^d} |\tilde{\Phi}_h(x)|dx \le \int_{[0,1]^d} \frac{1}{a}|\Phi_h(x)|dx \le \frac{r}{a}.$$

According to Lemma 11, we have $\|\tilde{\Phi}_h\|_\infty = O(r^{\frac{\alpha}{\alpha+d}})$. Thus $\|\Phi_h\|_\infty \le b\|\tilde{\Phi}_h\|_\infty = O(r^{\frac{\alpha}{\alpha+d}})$. Because this holds for all $h \in B(h^*, r)$, we have

$$\sup_{h\in B(h^*,r)} \|\Phi_h\|_\infty = O\left(r^{\frac{\alpha}{\alpha+d}}\right).$$

Now consider the region of disagreement of $B(h^*, r)$. Note that

$$DIS(B(h^*,r)) = \cup_{h\in B(h^*,r)}\{x : h(x) \ne h^*(x)\}.$$

Hence

$$\begin{aligned}\Pr_{X\sim\mathcal{D}_X}(x \in DIS(B(h^*,r))) &= \Pr_{X\sim\mathcal{D}_X}\left(x \in \cup_{h\in B(h^*,r)}\{x : h(x) \ne h^*(x)\}\right) \\ &\le 2\int_{[0,1]^d} \sup_{h\in B(h^*,r)} \|\Phi_h\|_\infty dx^1 \dots dx^d = O\left(r^{\frac{\alpha}{\alpha+d}}\right) = O\left(r \cdot \left(\frac{1}{r}\right)^{\frac{d}{\alpha+d}}\right).\end{aligned}$$

The theorem follows by the definition of $\theta(\varepsilon)$. ■

Theorem 10 can be proved similarly by using Lemma 12.

In the next theorem, we give lower bounds on the disagreement coefficient for finite smooth problems under the condition that the marginal distribution $\mathcal{D}_X$ is the uniform distribution.[5] Note that in this case the lower bound matches the upper bound in Theorem 9. Thus in general Theorem 9 cannot be improved.

Theorem 13 *Let the hypothesis space be $\mathcal{H}_C^\alpha$ where $\alpha < \infty$. Assume that the marginal distribution $\mathcal{D}_X$ is uniform on $[0,1]^{d+1}$. Then the disagreement coefficient has the following lower bound*[6]

$$\theta(\varepsilon) = \Omega\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}}\right).$$

Proof Without loss of generality, we assume that the classification boundary of the optimal classifier h^* is the graph of function $x^{d+1} = f(x^1, x^2, \dots, x^d) \equiv 1/2$. That is, the classification boundary of h^* is a hyperplane orthogonal to the $d+1$th axis. We will show that for most points $(x^1, x^2, \dots, x^{d+1}) \in [0,1]^{d+1}$ that are $\varepsilon \cdot \left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}}$-close to the classification boundary of h^*, that is, $|x^{d+1} - \frac{1}{2}| \le \varepsilon \cdot \left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}}$, there is a $h_f \in \mathcal{H}_C^\alpha$ satisfying

$$h_f(x^1, x^2, \dots, x^{d+1}) \ne h^*(x^1, x^2, \dots, x^{d+1}),$$

5. The condition can be relaxed to that $\mathcal{D}_X$ is bounded from above and below by positive constants.

6. The bound was obtained with the help of Yanqi Dai, Kai Fan, Chicheng Zhang and Ziteng Wang. It improves a previous lower bound $\Omega\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d}{2\alpha+d}}\right)$.

and at the same time

$$\rho(h_f, h^*) = \Pr(h_f(X) \neq h^*(X)) \leq \varepsilon,$$

and therefore $h_f \in B(h^*, \varepsilon)$. Thus the volume of $DIS(B(h^*, \varepsilon))$ is $\Omega\left(\varepsilon \cdot \left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}}\right)$; and consequently

$$\theta(\varepsilon) \geq \frac{\Pr(DIS(B(h^*, \varepsilon)))}{\varepsilon} = \Omega\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}}\right).$$

For this purpose, fixing $(x^1, x^2, \ldots, x^{d+1}) \in [0,1]^{d+1}$ with

$$0 \leq x^{d+1} - \frac{1}{2} \leq c\varepsilon\left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}},$$

for some constant c. We only consider point $(x^1, x^2, \ldots, x^{d+1}) \in [0,1]^{d+1}$ that are not too close to the "boundary" of $[0,1]^{d+1}$. (e.g., $0.1 \leq x^i \leq 0.9$ for all $1 \leq i \leq d$.) We construct h_f whose classification boundary is the graph of the following function f. For convenience, we shift $(x^1, x^2, \ldots, x^d, \frac{1}{2})$ to the origin. Let f be defined on $[0,1]^d$ as

$$f(u_1, u_2, \ldots, u_d) = \begin{cases} \xi^{-\alpha}\left(\xi^2 - \sum_{i=1}^d u_i^2\right)^{\alpha} & \text{if } \sum_{i=1}^d u_i^2 \leq \xi^2, \\ 0 & \text{otherwise,} \end{cases}$$

where ξ is determined by

$$\int_\Omega |f| d\omega = \varepsilon,$$

that is, $\rho(h_f, h^*) = \varepsilon$, and Ω is the region obtained from $[0,1]^d$ after shifting $(x^1, x^2, \ldots, x^d, \frac{1}{2})$ to the origin.

First, it is not hard to check by calculus that f is αth order smooth. Next, since $\int_\Omega |f| d\omega = \varepsilon$, it is not difficult to calculate that $\xi = c'\varepsilon^{\frac{1}{\alpha+d}}$, for some constant c'. Thus

$$\|f\|_\infty = f(0, 0, \ldots, 0) = c'\varepsilon^{\frac{\alpha}{\alpha+d}} = c'\varepsilon\left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}}.$$

So we have $h_f(0, 0, \ldots, 0) \neq h^*(0, 0, \ldots, 0)$ and $\rho(h_f, h^*) = \varepsilon$. This completes the proof. ■

For infinite smoothness, we do not know any lower bound for the disagreement coefficient larger than the trivial $\Omega(1)$.

3.2.1 LABEL COMPLEXITY FOR SMOOTH PROBLEMS

Now we combine our results (Theorem 9 and Theorem 10) with the label complexity bounds for active learning (Theorem 4 and (4)) and show that active learning is strictly superior to passive learning for smooth problems.

Remember that under Tsybakov's noise conditions the label complexity of active learning is (see Theorem 4 and (4))

$$O\left(\theta(\varepsilon^{\frac{1}{\kappa}})\left(\frac{1}{\varepsilon}\right)^{2-\frac{2-p}{\kappa}}\left(\log\frac{1}{\varepsilon} + \log\frac{1}{\delta}\right)\right).$$

While for passive learning it is (see (5))

$$O\left(\left(\frac{1}{\varepsilon}\right)^{2-\frac{1-p}{\kappa}}\left(\log\frac{1}{\delta}+\log\log\frac{1}{\varepsilon}\right)\right).$$

We see that if

$$\theta(\varepsilon^{\frac{1}{\kappa}})=o\left(\left(\frac{1}{\varepsilon}\right)^{\frac{1}{\kappa}}\right),$$

then active learning requires strictly fewer labels than passive learning.

By Theorem 9 and remember that $\alpha > d$ (see Proposition 8) we obtain

$$\theta(\varepsilon^{\frac{1}{\kappa}})=O\left(\frac{1}{\varepsilon}\right)^{\frac{d}{\kappa(\alpha+d)}}=o\left(\left(\frac{1}{\varepsilon}\right)^{\frac{1}{2\kappa}}\right)=o\left(\left(\frac{1}{\varepsilon}\right)^{\frac{1}{\kappa}}\right).$$

So we have the following conclusion.

Theorem 14 *Assume that the Tsybakov noise exponent* κ *is finite. Then active learning algorithms* A^2 *and DHM have label complexity strictly smaller than passive learning for* α*th order smooth problems whenever* $\alpha > d$.

3.3 Discussion

In this section we discuss and compare our results to a closely related work due to Castro and Nowak (2008), which also studied the label complexity of smooth problems under Tsybakov's noise condition. Castro and Nowak's work is heavily based on their detailed analysis of actively learning a threshold on $[0,1]$ described below.

Consider the learning problem in which the instance space $\mathcal{X}=[0,1]$; the hypothesis space $\mathcal{H}$ contains all threshold functions, that is, $\mathcal{H}=\{\mathbb{I}(x\geq t): t\in[0,1]\}\cup\{\mathbb{I}(x<t): t\in[0,1]\}$, where $\mathbb{I}$ is indicator function; and the marginal distribution $\mathcal{D}_X$ is the uniform distribution on $[0,1]$. Suppose that the Bayes classifier $h_B^*\in\mathcal{H}$. Assume that the learning problem satisfies the "geometric" Tsybakov's noise condition

$$\left|\eta(x)-\frac{1}{2}\right|\geq b\,|x-x_B^*|^{\kappa-1}, \tag{6}$$

for some constant $b>0$ and for all x such that $|\eta(x)-\frac{1}{2}|\leq\tau_0$ with the constant $\tau_0>0$. Here x_B^* is the threshold of the Bayes classifier. (One can verity that (6) implies the ordinary Tsybakov's condition with noise exponent κ when $\mathcal{D}_X$ is uniform on [0,1].) In addition, assume that the learning problem satisfies a reverse-sided Tsybakov's condition

$$\left|\eta(x)-\frac{1}{2}\right|\leq B\,|x-x_B^*|^{\kappa-1},$$

for some constant $B>0$.

Under these assumptions, Castro and Nowak showed that an active learning algorithm they attributed to Burnashev and Zigangirov (1974) (will be referred to as BZ), which is essentially a Bayesian binary search algorithm,[7] has label complexity $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{2-\frac{2}{\kappa}}\right)$. Moreover, due to the

7. Note that this BZ algorithm can choose any point x from the instance space, not necessarily from the given pool of unlabeled data. This model is called membership query, making stronger assumptions than the pool-based active learning model.

reverse-sided Tsybakov's condition, one can show that with high probability that the threshold $\hat{x}$ returned by the active learning algorithm converges to the Bayes threshold x_B^* exponentially fast with respect to the number of label requests.[8]

Castro and Nowak then generalized this result to smooth problems similar to what we studied in this paper. Let the hypothesis space be $\mathcal{H}_C^\alpha$. Suppose that the Bayes classifier $h_B^* \in \mathcal{H}_C^\alpha$. Assume that on every vertical line segment in $[0,1]^{d+1}$, (that is, for every $\{(x^1,x^2,\dots,x^d,x^{d+1}) : x^{d+1} \in [0,1]\}$, $(x^1,x^2,\dots,x^d) \in [0,1]^d$,) the one dimensional distribution $\eta_{x^1,\dots,x^d}(x^{d+1})$ satisfies the two-sided geometric Tsybakov's condition with noise exponent κ. Based on these assumptions, they proposed the following active learning algorithm: Choosing M vertical line segments in $[0,1]^{d+1}$. Performing one dimensional threshold learning on each line segment as in the one-dimensional case described above. After obtaining the threshold for each line, doing a piecewise polynomial interpolation on these thresholds and return the interpolation function as the classification boundary. They showed that this algorithm has label complexity $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{2-\frac{2-\frac{d}{\alpha}}{\kappa}}\right)$.

In sum, their algorithm makes the following main assumptions:

(A1) On every vertical line, the conditional distribution is two-sided Tsybakov. Thus the distribution $\mathcal{D}_{XY}$ has a uniform "one-dimensional" behavior along the $(d+1)$th axis.

(A2) The algorithm can choose any point from $\mathcal{X} = [0,1]^{d+1}$ and ask for its label.

Comparing the label complexity of this algorithm

$$\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{2-\frac{2-\frac{d}{\alpha}}{\kappa}}\right)$$

and that obtained from Theorem 4 and Proposition 8

$$\tilde{O}\left(\theta(\varepsilon_0)\left(\frac{1}{\varepsilon}\right)^{2-\frac{2-\frac{d}{\alpha}}{\kappa}}\right)$$

one sees that their label complexity is smaller by $\theta(\varepsilon_0)$. It seems that the disagreement coefficient of smooth problem does not play a role in their label complexity formula. The reason is that the assumption (A1) in their model assumes that the distribution of the problem has a uniform one-dimensional behavior: on each line segment parallel to the $d+1$th axis, the conditional distribution $\eta_{x^1,\dots,x^d}(x^{d+1})$ satisfies the two-sided Tsybakov's condition with equal noise exponent κ. Therefore the algorithm can assign equal label budget to each line segment for one-dimensional learning. Recall that the disagreement coefficient of the one-dimensional threshold learning problem is at most 2 for all $\varepsilon > 0$, so there seems no $\theta(\varepsilon)$ term in the final label complexity formula. If, instead of assumption (A1), we assume the ordinary Tsybakov's noise condition, the algorithm has to assign

8. It needs to be pointed out that the BZ algorithm requires that the noise exponent κ is known and the algorithm takes it as imput. But by Threorem 4 and (4) we know that both A^2 and DHM (with slight modifications) have the same label complexity and the convergence property, since the disagreement coefficient for this threshold problem is $\theta(\varepsilon) = 2$ for all $\varepsilon > 0$.

label budgets according to the "worst" noise on all the line segments, that is, the largest κ of the conditional distributions $\eta_{x^1,\dots,x^d}(x^{d+1})$ over all $(x^1,\dots,x^d)$. But under the ordinary Tsybakov's condition, the largest κ on lines can be arbitrarily large, resulting a label complexity equal to that of passive learning.

4. Proof of Lemma 11 and 12—Some Generalizations of the Landau-Kolmogorov Type Inequalities

In this section we give proofs of Lemma 11 and Lemma 12. The two lemmas are closely related to the Landau-Kolmogorov type inequalities (Landau, 1913; Kolmogorov, 1938; Schoenberg, 1973) (see also Mitrinović et al., 1991 Chapter I for a comprehensive survey), and specifically the following result due to Gorny (1939).

Theorem 15 *Let $f(x)$ be a function defined on $[0,1]$ and has derivatives up to the nth order. Let $M_k = \|f^{(k)}\|_\infty$, $k = 0,1,\dots,n$. Then*

$$M_k \le 4\left(\frac{n}{k}\right)^k e^k M_0^{1-\frac{k}{n}} M_n'^{\frac{k}{n}},$$

where $M_n' = \max(M_0 n!, M_n)$.

Roughly, for function f defined on a finite interval, the above theorem bounds the ∞-norm of the kth order derivative of f by the ∞-norm of f and its nth derivative.

In order to prove Lemma 11, we give the following generalization of Theorem 15 (in the direction of dimensionality and non-integer smoothness). Our proof is elementary and is simpler than Gorny's proof of Theorem 15. But note that the definition of M'_α in Theorem 16 is different to that of Theorem 15.

Theorem 16 *Let f be a function defined on $[0,1]^d$. Let $\alpha > 1$ be a real number. Assume that f has partial derivatives up to order $\underline{\alpha}$. For all $1 \le t \le \alpha$, define*

$$M_t = \max_{|\mathbf{k}|=\underline{t}} \sup_{x,x'} \frac{|D^{\mathbf{k}} f(x) - D^{\mathbf{k}} f(x')|}{\|x-x'\|^{t-\underline{t}}},$$

where $D^{\mathbf{k}}$ is the differential operator defined in Section 3.1 and $x,x' \in [0,1]^d$. Also define $M_0 = \sup_{x\in[0,1]^d} f(x)$. Then

$$M_k \le C M_0^{1-\frac{k}{\alpha}} M_\alpha'^{\frac{k}{\alpha}}, \tag{7}$$

where $M'_\alpha = \max(M_0, M_\alpha)$, $k = 1,2,\dots,\underline{\alpha}$, and the constant C depends on k and α but does not depend on M_0 and M_α.

Lemma 11 can be derived from Theorem 16.

Proof of Lemma 11 The proof has two steps. First, we construct a function $\overline{f}$ by scaling Φ and redefine the domain of the function, so that a) the integral of $\overline{f}$ over the unit hypercube is at most 1; b) the $\underline{\alpha}$th order derivatives of $\overline{f}$ is Hölder continuous with the same constant as Φ, and c) $\|\overline{f}\|_\infty = \left(\frac{1}{r}\right)^{\frac{\alpha}{\alpha+d}} \|\Phi\|_\infty$. Next, we use Theorem16 to show that $\|\overline{f}\|_\infty$ can be bounded from above by a constant depending only on α, d, C (recall that $\Phi \in F_C^{\alpha}$) but independent of r. Combining the two steps concludes the theorem.

Now, for the first step assume

$$\int_{[0,1]^d} |\Phi(x)|dx = t \leq r.$$

Let

$$f(x^1,x^2,\ldots,x^d) = \left(\frac{1}{t}\right)^{\frac{\alpha}{\alpha+d}} \Phi(t^{\frac{1}{\alpha+d}}x^1, t^{\frac{1}{\alpha+d}}x^2,\ldots,t^{\frac{1}{\alpha+d}}x^d),$$

where now the domain of f is $(x^1,\ldots,x^d) \in [0,\frac{1}{t}^{\frac{1}{\alpha+d}}]^d$.

First, it is easy to check that

$$\int_{[0,\frac{1}{t}^{\frac{1}{\alpha+d}}]^d} |f(x)|dx = 1;$$

and the $\underline{\alpha}$th order derivatives of f is Hölder continuous with the same constant C as Φ. That is,

$$\max_{|\mathbf{k}|=\underline{\alpha}} \sup_{x,x'\in[0,\frac{1}{t}^{\frac{1}{\alpha+d}}]^d} \frac{|D^{\mathbf{k}}f(x)-D^{\mathbf{k}}f(x')|}{\|x-x'\|^{\alpha-\underline{\alpha}}} = \max_{|\mathbf{k}|=\underline{\alpha}} \sup_{x,x'\in[0,1]^d} \frac{|D^{\mathbf{k}}\Phi(x)-D^{\mathbf{k}}\Phi(x')|}{\|x-x'\|^{\alpha-\underline{\alpha}}} \leq C.$$

In addition, clearly we have

$$\|\Phi\|_\infty = t^{\frac{\alpha}{\alpha+d}}\|f\|_\infty \leq r^{\frac{\alpha}{\alpha+d}}\|f\|_\infty.$$

Thus in order to prove the lemma, we only need to show $\|f\|_\infty$ is bounded from above by a universal constant independent of r.

Note that the domain of f is $[0,\frac{1}{t}^{\frac{1}{\alpha+d}}]^d$, larger than $[0,1]^d$. Assume f achieves its maximum at $(a_1,a_2,\ldots,a_d) \in [0,\frac{1}{t}^{\frac{1}{\alpha+d}}]^d$. Now we truncate the domain of f to a d-dimensional hypercube $[z_1,z_1+1]\otimes[z_2,z_2+1]\otimes\ldots,\otimes[z_d,z_d+1]$ so that $(a_1,a_2,\ldots,a_d) \in [z_1,z_1+1]\otimes[z_2,z_2+1]\otimes\ldots,\otimes[z_d,z_d+1]$. Let $\overline{f}$ be the function by restricting f on this hypercube $[z_1,z_1+1]\otimes[z_2,z_2+1]\otimes\ldots,\otimes[z_d,z_d+1]$. Clearly, we have

$$\|\overline{f}\|_\infty = \|f\|_\infty,$$

where $\|\overline{f}\|_\infty$ is the maximum over the hypercube $[z_1,z_1+1]\otimes[z_2,z_2+1]\otimes\ldots,\otimes[z_d,z_d+1]$. Thus we just need to show $\|\overline{f}\|_\infty$ has a universal upper bound.

Now we begin the second step of the proof, where our goal is to show $\overline{f}$ has an upper bound independent of r. Assume $z_i = 0$ for $i=1,\ldots,d$ by shifting if necessary.

Let

$$g_d(x^1,\ldots,x^d) = \int_0^{x^d} \overline{f}(x^1,\ldots,x^{d-1},u_d)du_d.$$

For any fixed $x^1,\ldots,x^{d-1}$, consider g_d as a function of the single variable x^d. Since $\overline{f}$ is the first order derivative of g_d, it is easy to check that g_d has derivatives up to order $\underline{\alpha}+1$ with respect to x^d and its $\underline{\alpha}+1$ order derivative is Hölder continuous with constant C. Thus according to Theorem16, we have

$$\|\overline{f}\|_\infty \leq \|g_d\|_\infty^{\frac{\alpha}{\alpha+1}} C^{\frac{1}{\alpha+1}}. \tag{8}$$

Similarly, let

$$g_i(x^1,\dots,x^d)=\int_0^{x^i} g_{i+1}(x^1,\dots,x^{i-1},u_i,x^{i+1},\dots,x^d)du_i, \quad i=1,2,\dots,d-1.$$

For each g_i use the above argument and observe that the $\underline{\alpha}+1$th order derivative of each g_i is bounded from above by C, then it is easy to obtain that

$$\|g_i\|_\infty \le \|g_{i+1}\|_\infty^{\frac{\alpha}{\alpha+1}} C^{\frac{1}{\alpha+1}}. \tag{9}$$

Combining (8) and (9) for all $i=1,2,\dots,d$ we have

$$\|\overline{f}\|_\infty \le \|g_1\|_\infty^{\left(\frac{\alpha}{\alpha+1}\right)^d} C',$$

where C' is a constant depending on $C,\ \alpha,\ d$. This completes the proof since

$$g_1(x^1,\dots,x^d)=\int_0^{x^1}\dots\int_0^{x^d}\overline{f}(u_1,\dots,u_d)du_1,\dots,du_d \le 1.$$

■

Proof of Theorem 16 The structure of the proof is as follows: we first deal with the case of $d=1$ and then generalize to $d>1$. For the case of $d=1$, we first show the case $1<\alpha\le 2$, and then prove general α by induction.

Now assume $d=1$ and $1<\alpha\le 2$. Our goal is to show

$$M_1 \le C M_0^{1-\frac{1}{\alpha}} M_\alpha'^{\frac{1}{\alpha}}.$$

For any fixed $x\in[0,1]$, there must be a $y\in[0,1]$ such that $|y-x|=1/2$. We thus have

$$\frac{f(y)-f(x)}{y-x}=f'(x+u), \tag{10}$$

where $|u|\le 1/2$. Since $1<\alpha\le 2$, we know $\underline{\alpha}=1$. By the definition of M_α we have

$$|f'(x+u)-f'(x)|\le M_\alpha |u|^{\alpha-1}. \tag{11}$$

Combining (10) and (11) and recall $|y-x|=1/2$, we obtain

$$\begin{aligned} |f'(x)| &\le \left|\frac{f(y)-f(x)}{y-x}\right| + M_\alpha |u|^{\alpha-1} \\ &\le 4M_0 + \left(\frac{1}{2}\right)^{\alpha-1} M_\alpha \\ &\le 4M_0 + M_\alpha. \end{aligned} \tag{12}$$

Let $g(x)=f(ax+r)$, where $0<a\le 1$, $r\in[0,1-a]$ and $x\in[0,1]$. Let

$$M_0^g=\sup_{x\in[0,1]}|g(x)|, \qquad M_\alpha^g=\sup_{x,x'\in[0,1]}\frac{|g'(x)-g'(x')|}{|x-x'|^{\alpha-1}}.$$

It is easy to check that $M_0^g \leq M_0$ and $M_\alpha^g \leq a^\alpha M_\alpha$. Applying (12) to $g(x)$, which is defined on $[0,1]$, we obtain that for every $a \in (0,1]$

$$|f'(ax+r)| = \frac{1}{a}|g'(x)| \leq \frac{4M_0^g}{a} + \frac{M_\alpha^g}{a} \leq \frac{4M_0}{a} + a^{\alpha-1}M_\alpha.$$

Taking

$$a = \min\left(\left(\frac{M_0}{M_\alpha}\right)^{\frac{1}{\alpha}}, 1\right),$$

we obtain for all $x \in [r, a+r]$

$$|f'(x)| \leq 5M_0^{1-\frac{1}{\alpha}} M_\alpha'^{\frac{1}{\alpha}},$$

where $M_\alpha' = \max(M_0, M_\alpha)$. Since $r \in [0, 1-a]$ is arbitrary, we have that

$$M_1 = \sup_{x\in[0,1]} |f'(x)| \leq 5M_0^{1-\frac{1}{\alpha}} M_\alpha'^{\frac{1}{\alpha}}. \tag{13}$$

Note that this implies that for all nonnegative integer m and $1 < \alpha \leq 2$, we have

$$M_{m+1} \leq 5M_m^{1-\frac{1}{\alpha}} M_{m+\alpha}'^{\frac{1}{\alpha}}.$$

We next prove the general $\alpha > 1$ case. Let n be a positive integer. By induction, assume for all $1 < \alpha \leq n$ we already have, for $k = 1, 2, \ldots, \underline{\alpha}$

$$M_k \leq CM_0^{1-\frac{k}{\alpha}} \left(\max(M_0, M_\alpha)\right)^{\frac{k}{\alpha}}, \tag{14}$$

where the constant C depends on k and α but does not depend on M_0 and M_α. (In the following the constant C my be different from line to line and even in the same line.) We will prove that (14) is true for $\alpha \in (n, n+1]$. Here we will treat the two cases $1 \leq k < n$ and $k = n$ separately.

For the case $1 \leq k < n$, since $\alpha - k \leq n$, by the assumption of the induction we have

$$M_n \leq CM_k^{1-\frac{n-k}{\alpha-k}} \left(\max(M_k, M_\alpha)\right)^{\frac{n-k}{\alpha-k}}. \tag{15}$$

Combining (14) and (15), and setting $\alpha = n$ in (14). We distinguish three cases.

Case I: $M_0 > M_n$
We have

$$\begin{aligned} M_k &\leq CM_0^{1-\frac{k}{n}} \left(\max(M_0, M_n)\right)^{\frac{k}{n}} \\ &= CM_0 \\ &\leq CM_0^{1-\frac{k}{\alpha}} \left(\max(M_0, M_\alpha)\right)^{\frac{k}{\alpha}}. \end{aligned}$$

Case II: $M_0 \leq M_n$ and $M_k > M_\alpha$
We have

$$M_k \leq CM_0^{1-\frac{k}{n}} M_n^{\frac{k}{n}} \leq CM_0^{1-\frac{k}{n}} M_k^{\frac{k}{n}}.$$

Thus

$$M_k \le CM_0 \le CM_0^{1-\frac{k}{\alpha}} \left(\max(M_0, M_\alpha)\right)^{\frac{k}{\alpha}}.$$

Case III: $M_0 \le M_n$ and $M_k \le M_\alpha$
We have

$$\begin{aligned} M_k &\le CM_0^{1-\frac{k}{n}} M_n^{\frac{k}{n}} \\ &\le CM_0^{1-\frac{k}{n}} \left(M_k^{1-\frac{n-k}{\alpha-k}} M_\alpha^{\frac{n-k}{\alpha-k}} \right)^{\frac{k}{n}} \\ &= CM_0^{1-\frac{k}{n}} M_k^{\frac{k(\alpha-n)}{n(\alpha-k)}} M_\alpha^{\frac{k(n-k)}{n(\alpha-k)}}. \end{aligned}$$

We obtain after some simple calculations that

$$M_k \le CM_0^{1-\frac{k}{\alpha}} M_\alpha^{\frac{k}{\alpha}}.$$

This completes the proof for $1 \le k < n$.

For the case $k = n$, note that

$$M_n \le CM_1^{1-\frac{n-1}{\alpha-1}} \max(M_1, M_\alpha)^{\frac{n-1}{\alpha-1}}, \tag{16}$$

and

$$M_1 \le CM_0^{1-\frac{1}{n}} \max(M_0, M_n)^{\frac{1}{n}}. \tag{17}$$

We need to distinguish four cases.
Case I: $M_1 > M_\alpha$ and $M_0 > M_n$
Combining (16) and (17), we have

$$M_n \le CM_1 \le CM_0 \le CM_0^{1-\frac{n}{\alpha}} \left(\max(M_0, M_\alpha)\right)^{\frac{n}{\alpha}}.$$

Case II: $M_1 > M_\alpha$ and $M_0 \le M_n$
We have

$$M_n \le CM_1 \le CM_0^{1-\frac{1}{n}} M_n^{\frac{1}{n}}.$$

Thus

$$M_n \le CM_0 \le CM_0^{1-\frac{n}{\alpha}} \left(\max(M_0, M_\alpha)\right)^{\frac{n}{\alpha}}.$$

Case III: $M_1 \le M_\alpha$ and $M_0 > M_n$
We have

$$\begin{aligned} M_n &\le CM_1^{1-\frac{n-1}{\alpha-1}} M_\alpha^{\frac{n-1}{\alpha-1}} \\ &\le CM_0^{1-\frac{n-1}{\alpha-1}} M_\alpha^{\frac{n-1}{\alpha-1}}. \end{aligned} \tag{18}$$

If $M_0 \le M_\alpha$, then from (18) we obtain

$$\begin{aligned} M_n &\le CM_0^{1-\frac{n}{\alpha}} M_\alpha^{\frac{n}{\alpha}} \left(\frac{M_0}{M_\alpha} \right)^{\frac{n}{\alpha} - \frac{n-1}{\alpha-1}} \\ &\le CM_0^{1-\frac{n}{\alpha}} \left(\max(M_0, M_\alpha)\right)^{\frac{n}{\alpha}}. \end{aligned}$$

Otherwise, $M_0 > M_\alpha$, by (18)

$$M_n \leq C M_0.$$

Case IV: $M_1 \leq M_\alpha$ and $M_0 \leq M_n$
We have

$$\begin{aligned} M_n &\leq C\left(M_0^{1-\frac{1}{n}} M_n^{\frac{1}{n}}\right)^{1-\frac{n-1}{\alpha-1}} M_\alpha^{\frac{n-1}{\alpha-1}} \\ &= C M_n^{\frac{(n-1)(\alpha-n)}{n(\alpha-1)}} M_\alpha^{\frac{n-1}{\alpha-1}} M_n^{\frac{\alpha-n}{n(\alpha-1)}}. \end{aligned}$$

After some simple calculation, this yields

$$M_n \leq C M_0^{1-\frac{n}{\alpha}} M_\alpha^{\frac{n}{\alpha}}.$$

This completes the proof of the $k = n$ case, and we finished the discussion of the $d = 1$ case.

The above arguments are easy to generalize to the $d \geq 2$ case. Consider the function $f(x^1, \dots, x^d)$. Again we first look at $1 < \alpha \leq 2$. Assume M_1 is achieved at the jth partial derivative of f. That is,

$$\left\| \frac{\partial f}{\partial x^j} \right\|_\infty = M_1.$$

Fixing $x^1, \dots, x^{j-1}, x^{j+1}, \dots, x^d$, consider f as a function of a single variable x^j. By the argument for the $d = 1$ case, we know that

$$M_1 \leq 5 M_0^{1-\frac{1}{\alpha}} \tilde{M}^{\frac{1}{\alpha}},$$

where

$$\tilde{M} = \max\left(M_0, \sup_{x,x'} \frac{\left|\frac{\partial f}{\partial x^j}\big|_x - \frac{\partial f}{\partial x^j}\big|_{x'}\right|}{\|x - x'\|^{\alpha-1}}\right).$$

Clearly, $\tilde{M} \leq \max(M_0, M_\alpha) = M'_\alpha$. Hence for $1 < \alpha \leq 2$, we have $M_1 \leq 5 M_0^{1-\frac{1}{\alpha}} M_\alpha'^{\frac{1}{\alpha}}$. Finally, using the previous induction argument and noting that it does not depend on the dimensionality d, we obtain the desired result for all $\alpha > 1$. ■

To prove Lemma 12 however, Theorem 16 is not a suitable tool. Note that the M'_α in (7) is $\max(M_0, M_\alpha)$, while in Theorem 15 $M'_\alpha = \max(n! M_0, M_\alpha)$. Therefore the constant in Theorem 16 grows exponentially regarding to α. In the following we give another generalization of Gorny's inequality which will be used to prove Lemma 12. The price however is that it cannot handle the non-integer smoothness.

Theorem 17 *Let $f(x)$ be defined on $[0,1]^d$ and have uniformly bounded partial derivatives up to order n. Let*

$$M_k = \sup_{|\mathbf{k}|=k} \|D^{\mathbf{k}} f\|_\infty, \quad k = 0, 1, 2, \dots, n.$$

Then

$$M_k \leq C n^k M_0^{1-\frac{k}{n}} M_n'^{\frac{k}{n}},$$

where $M'_n = \max(n! M_0, M_n)$, and C is a constant depending on k but does not depend on M_0, M_n and n.

This theorem is a straightforward generalization of Gorny's theorem to multidimension. Now we can use this theorem to prove Lemma 12.

Proof of Lemma 12 Similar to the proof of Lemma 11, for $(x^1,\dots,x^d)\in[0,1]^d$, let

$$f(x^1,\dots,x^d)=\int_0^{x^1}\int_0^{x^2}\cdots\int_0^{x^d}\Phi(u_1,\dots,u_d)du_1,\dots,du_d.$$

It is easy to check that f is infinitely smooth. Let M_t be defined as in Theorem 15 for f. Clearly $M_0\le r$ and $\|\Phi\|_\infty\le M_d$. Since f is infinitely smooth, there is a constant C such that $M_n\le C$ for $n=d+1,d+2,\dots$.

Now for r sufficiently small, take $n=\frac{\log\frac{1}{r}}{\log\log\frac{1}{r}}$. Let's first look at $n!M_0$. Note that

$$n^n=\left(\frac{\log\frac{1}{r}}{\log\log\frac{1}{r}}\right)^{\frac{\log\frac{1}{r}}{\log\log\frac{1}{r}}}\le\left(\log\frac{1}{r}\right)^{\frac{\log\frac{1}{r}}{\log\log\frac{1}{r}}}=\frac{1}{r}.$$

We have

$$\begin{aligned}M_0n! &\le r\sqrt{2\pi n}n^ne^{-n}\le\sqrt{2\pi\frac{\log\frac{1}{r}}{\log\log\frac{1}{r}}}(r)^{\frac{1}{\log\log\frac{1}{r}}}\\ &\le \sqrt{2\pi}\exp\left(\frac{\log\log\frac{1}{r}-\log\log\log\frac{1}{r}}{2}-\frac{\log\frac{1}{r}}{\log\log\frac{1}{r}}\right),\end{aligned}$$

which tends to zero as $r\to0$ and therefore

$$M'_n=\max(M_0n!,M_n))\le C.$$

Thus we have, by Theorem 17

$$\begin{aligned}\|\Phi\|_\infty &\le M_d\\ &\le Cn^dM_0^{1-\frac{d}{n}}M_n'^{\frac{d}{n}}\\ &\le Cn^dM_0^{1-\frac{d}{n}}\\ &\le C\left(\frac{\log\frac{1}{r}}{\log\log\frac{1}{r}}\right)^d r\left(\frac{1}{r}\right)^{\frac{d\log\log\frac{1}{r}}{\log\frac{1}{r}}}\\ &\le Cr\left(\log\frac{1}{r}\right)^{2d}.\end{aligned}$$

■

Proof of Theorem 17 We know that the theorem is valid when $d=1$. Now assume $d\ge2$. Using the same argument in the proof of Theorem 16, we have that for all positive integers n,

$$M_1\le CnM_0^{1-\frac{1}{n}}M_n'^{\frac{1}{n}},$$

and in general

$$M_{m+1} \le CnM_m^{1-\frac{1}{n}} M'^{\frac{1}{n}}_{m+n}, \tag{19}$$

for any nonnegative integer m.

Now we prove the theorem by induction on k. Assume we have already shown

$$M_{k-1} \le Cn^{k-1} M_0^{1-\frac{k-1}{n}} \left(\max(M_0 n!, M_n)\right)^{\frac{k-1}{n}}. \tag{20}$$

By (19) we have

$$M_k \le C(n-k+1) M_{k-1}^{1-\frac{1}{n-k+1}} \left(\max\left((n-k+1)! M_{k-1}, M_n\right)\right)^{\frac{1}{n-k+1}}. \tag{21}$$

We consider the following four cases separately. Note that below we will frequently use the fact that for $m = 1, 2, \ldots$

$$\frac{1}{\sqrt{2\pi}} (m!)^{\frac{1}{m}} e \le m \le (m!)^{\frac{1}{m}} e^{1+\frac{1}{12}}.$$

To see this, just note that

$$\sqrt{2\pi m} m^m e^{-(m+\frac{1}{12m})} \le m! \le \sqrt{2\pi m} m^m e^{-m},$$

and

$$1 \le \left(\sqrt{2\pi m}\right)^{\frac{1}{m}} \le \sqrt{2\pi}.$$

Case I: $(n-k+1)! M_{k-1} > M_n$ and $n! M_0 \le M_n$

From (21) we have

$$M_k \le C(n-k+1) M_{k-1} \left((n-k+1)!\right)^{\frac{1}{n-k+1}} \le C(n-k+1)^2 M_{k-1} \le Cn^2 M_{k-1}.$$

Taking into consideration of (20), we have

$$\begin{aligned} M_k &\le Cn^{k+1} M_0^{1-\frac{k-1}{n}} M'^{\frac{k-1}{n}}_n \\ &\le Cn^k M_0^{1-\frac{k}{n}} M'^{\frac{k}{n}}_n \left(n M_0^{\frac{1}{n}} M'^{-\frac{1}{n}}_n\right) \\ &\le Cn^k M_0^{1-\frac{k}{n}} M'^{\frac{k}{n}}_n \left(\frac{n! M_0}{M'_n}\right)^{\frac{1}{n}} \\ &\le Cn^k M_0^{1-\frac{k}{n}} M'^{\frac{k}{n}}_n. \end{aligned}$$

Case II: $(n-k+1)! M_{k-1} > M_n$ and $n! M_0 > M_n$

By the similar argument as in Case I, we have

$$\begin{aligned} M_k &\le Cn^{k+1} M_0^{1-\frac{k-1}{n}} M'^{\frac{k-1}{n}}_n \\ &= Cn^{k+1} M_0 (n!)^{\frac{k-1}{n}} \\ &\le Cn^k M_0 (n!)^{\frac{k}{n}} \\ &= Cn^k M_0^{1-\frac{k}{n}} (n! M_0)^{\frac{k}{n}}. \end{aligned}$$

Case III: $(n-k+1)!M_{k-1} \le M_n$ and $n!M_0 > M_n$

Combining (20) and (21),

$$\begin{aligned}
M_k &\le C(n-k+1)\left(n^{k-1}M_0(n!)^{\frac{k-1}{n}}\right)^{1-\frac{1}{n-k+1}} M_n^{\frac{1}{n-k+1}}\\
&\le Cn^k M_0^{1-\frac{1}{n-k+1}}(n!)^{(\frac{k-1}{n})(1-\frac{1}{n-k+1})}(n!M_0)^{\frac{1}{n-k+1}}\\
&\le Cn^k M_0(n!)^{\frac{k}{n}}\\
&\le Cn^k M_0^{1-\frac{k}{n}}(n!M_0)^{\frac{k}{n}}.
\end{aligned}$$

Case IV: $(n-k+1)!M_{k-1} \le M_n$ and $n!M_0 \le M_n$ Combining (20) and (21), we obtain

$$\begin{aligned}
M_k &\le C(n-k+1)M_{k-1}^{1-\frac{1}{n-k+1}} M_n^{\frac{1}{n-k+1}}\\
&\le Cn\left(n^{k-1}M_0^{1-\frac{k-1}{n}} M_n^{\frac{k-1}{n}}\right)^{1-\frac{1}{n-k+1}} M_n^{\frac{1}{n-k+1}}\\
&\le Cn^k M_0^{1-\frac{k}{n}} M_n^{\frac{k}{n}}.
\end{aligned}$$

This completes the proof. ■

5. Conclusion

This paper studies the disagreement coefficient of smooth problems and extends our previous results (Wang, 2009). Comparing to the worst case $\theta(\varepsilon) = \frac{1}{\varepsilon}$ for which active learning has the same label complexity as passive learning, the disagreement coefficient is $\theta(\varepsilon) = O\left(\left(\frac{1}{\varepsilon}\right)^{\frac{d}{\alpha+d}}\right)$ for αth ($\alpha < \infty$) order smooth problems, and is $\theta(\varepsilon) = O\left(\log^{2d}\left(\frac{1}{\varepsilon}\right)\right)$ for infinite order smooth problems. Combining with the bounds on the label complexity in terms of disagreement coefficient, we give sufficient conditions for which active learning algorithm A^2 and DHM are superior to passive learning under Tsybakov's noise condition.

Although we assume that the classification boundary is the graph of a function, our results can be generalized to the case that the boundaries are a finite number of functions. To be precise, consider N (N is even) functions $f_1(\mathbf{x}) \le \cdots \le f_N(\mathbf{x})$, for all $\mathbf{x} \in [0,1]^d$. Let $f_0(\mathbf{x}) \equiv 0$, $f_{N+1}(\mathbf{x}) \equiv 1$. The positive (or negative) set defined by these functions is $\{(\mathbf{x}, x^{d+1}): \ f_{2i}(\mathbf{x}) \le x^{d+1} \le f_{2i+1}(\mathbf{x}), \ i = 0, 1, \ldots, \frac{N}{2}\}$. It is easy to show that our main theorems still hold in this case. Moreover, using the techniques in Dudley (1999, page 259), our results may generalize to the case that the classification boundaries are intrinsically smooth, and not necessarily graphs of smooth functions. This would include a substantially richer class of problems which can be benefit from active learning.

There is an open problems worthy of further study. For infinitely smooth problems we proved that the disagreement coefficient can be upper and lower bounded by $O\left(\log^{2d}\left(\frac{1}{\varepsilon}\right)\right)$ and $\Omega(1)$ respectively. Improving the upper bound and (or) the lower bound would be interesting.

Acknowledgments

The author thank Yanqi Dai, Kai Fan, Chicheng Zhang and Ziteng Wang for very helpful discussions. In particular, they helped to improve Theorem 9 and Theorem 13 over previous versions. The author is grateful to an anonymous reviewer who gave many constructive suggestions. It greatly improves the quality of the paper. This work was supported by NSFC (61075003) and a grant from MOE-Microsoft Key Laboratory of Statistics and Information Technology of Peking University.

References

M.-F. Balcan, A.Beygelzimer, and J. Langford. Agnostic active learning. In *23th International Conference on Machine Learning*, 2006.

A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *26th International Conference on Machine Learning*, 2009.

A. Beygelzimer, D. Hsu, J. Langford, and T. Zhang. Agnostic active learning without constraints. In *Advances in Neural Information Processing Systems*, 2010.

M. Burnashev and K. Zigangirov. An interval estimation problem for controlled problem. *Problems of Information Transmission*, 10:223–231, 1974.

R. Castro and R. Nowak. Minimax bounds for active learning. *IEEE Transactions on Information Theory*, 54:2339–2353, 2008.

D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994.

S. Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems*, 2005.

S. Dasgupta and J. Langford. A tutorial on active learning, 2009.

S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Advances in Neural Information Processing Systems*, 2007.

R.M. Dudley. *Uniform Central Limit Theorems*. Cambridge University Press, 1999.

E. Friedman. Active learning for smooth problems. In *22th Annual Conference on Learning Theory*, 2009.

A. Gorny. Contribution a l'etude des fonctions dérivables d'une variable réelle. *Acta Mathematica*, 13:317–358, 1939.

S. Hanneke. A bound on the label complexity of agnostic active learning. In *24th International Conference on Machine Learning*, 2007.

S. Hanneke. Adaptive rates of convergence in active learning. In *22th Annual Conference on Learning Theory*, 2009.

S. Hanneke. Rates of convergence in active learning. *Annals of Statistics*, 39:333–361, 2011.

M. Kääriäinen. Active learning in the non-realizable case. In *17th International Conference on Algorithmic Learning Theory*, 2006.

A. Kolmogorov. Une généralisation de l'inégalite de J. Hadamard entre les bornes supérieurs des dérivés successives d'une fonction. *C. R. Académie des Sciences, Paris*, 207:763–765, 1938.

V. Koltchinskii. Local rademacher complexities and oracle inequalities in risk minimization. *The Annals of Statistics*, 34:2593–2656, 2006.

V. Koltchinskii. Rademacher complexities and bounding the excess risk in active learning. *Journal of Machine Learning Research*, 11:2457–2485, 2010.

E. Landau. Einige ungleichungen für zweimal differentiierbare funktionen. *Proceedings of the London Mathematical Society*, 13:43–49, 1913.

D.S. Mitrinović, J.E. Pečarié, and A.M. Fink. *Inequalities Involving Functions and Their Integrals and Derivatives*. Kluwer Academic Publishers, 1991.

I.J. Schoenberg. The elementary cases of landau's problem of inequlities between derivatives. *The American Mathematical Monthly*, 80:121–158, 1973.

A. Tsybakov. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32: 135–166, 2004.

A. van der Vaart and J. Wellner. *Weak Convergence and Empirical Processes with Application to Statistics*. Springer Verlag, 1996.

V. Vapnik. *Statistical Learning Theory*. John Wiely and Sons, 1998.

L. Wang. Sufficient conditions for agnostic active learnable. In *Advances in Neural Information Processing Systems*, 2009.

Journal of Machine Learning Research 12 (2011) 2293-2296 Submitted 5/10; Revised 3/11; Published 7/11

MSVMpack: A Multi-Class Support Vector Machine Package

Fabien Lauer FABIEN.LAUER@LORIA.FR
Yann Guermeur YANN.GUERMEUR@LORIA.FR
LORIA – Equipe ABC
Campus Scientifique, BP 239
54506 Vandœuvre-lès-Nancy cedex, France

Editor: Mikio Braun

Abstract

This paper describes MSVMpack, an open source software package dedicated to our generic model of *multi-class* support vector machine. All four multi-class support vector machines (M-SVMs) proposed so far in the literature appear as instances of this model. MSVMpack provides for them the first unified implementation and offers a convenient basis to develop other instances. This is also the first parallel implementation for M-SVMs. The package consists in a set of command-line tools with a callable library. The documentation includes a tutorial, a user's guide and a developer's guide.

Keywords: multi-class support vector machines, open source, C

1. Introduction

In the framework of polytomy computation, a multi-class support vector machine (M-SVM) is a support vector machine (SVM) dealing with all the categories simultaneously. Four M-SVMs can be found in the literature: the models of Weston and Watkins (1998), Crammer and Singer (2001), Lee et al. (2004), and the M-SVM^2 of Guermeur and Monfrini (2011). The proposed software implements them all in a single package named MSVMpack. Its design paves the way for the implementation of our generic model of M-SVM and the integration of additional functionalities such as model selection algorithms. The current version offers a parallel implementation with the possibility to use custom kernels. This software package is available for Linux under the terms of the GPL at `http://www.loria.fr/~lauer/MSVMpack/` and provides two command-line tools with a C application programming interface without dependencies beside a linear programming solver.

2. Multi-Class Support Vector Machines

We consider Q-category classification problems where $\mathcal{X}$ is the description space and the set $\mathcal{Y}$ of the categories can be identified with $[\![1, Q]\!]$. Let κ be a real-valued positive type function (Berlinet and Thomas-Agnan, 2004) on $\mathcal{X}^2$ and let $(\mathbf{H}_\kappa, \langle \cdot, \cdot \rangle_{\mathbf{H}_\kappa})$ be the corresponding reproducing kernel Hilbert space. Let $\mathcal{H} = \mathbf{H}_\kappa^Q$ and $\mathcal{H} = (\mathbf{H}_\kappa + \{1\})^Q$. $\mathcal{H}$ is the class of functions $h = (h_k)_{1 \leqslant k \leqslant Q}$ from $\mathcal{X}$ to $\mathbb{R}^Q$ that can be written as $h(\cdot) = h(\cdot) + b = \left(h_k(\cdot) + b_k\right)_{1 \leqslant k \leqslant Q}$, where $h = \left(h_k\right)_{1 \leqslant k \leqslant Q} \in \mathcal{H}$ and $b = (b_k)_{1 \leqslant k \leqslant Q} \in \mathbb{R}^Q$. A function h assigns the category y to x if and only if $y = \operatorname{argmax}_{1 \leqslant k \leqslant Q} h_k(x)$ (cases of ex æquo are dealt with by introducing a dummy category). $\mathcal{H}$ is endowed with the norm

Reference	M-SVM type	M	p	K_1	K_2	K_3
Weston and Watkins (1998)	WW	I_{Qm}	1	1	1	0
Crammer and Singer (2001)	CS	$\frac{1}{Q-1}I_{Qm}$	1	1	1	1
Lee et al. (2004)	LLW	I_{Qm}	1	0	$\frac{1}{Q-1}$	0
Guermeur and Monfrini (2011)	MSVM2	$M^{(2)}$	2	0	$\frac{1}{Q-1}$	0

Table 1: Specifications of the M-SVMs with their type as used by the MSVMpack interface.

$\|\cdot\|_{\mathcal{H}}$ given by:

$$\forall h \in \mathcal{H}, \ \|h\|_{\mathcal{H}} = \sqrt{\sum_{k=1}^{Q} \langle h_k, h_k \rangle_{\mathbf{H}_\kappa}} = \left\| \left(\|h_k\|_{\mathbf{H}_\kappa} \right)_{1 \leqslant k \leqslant Q} \right\|_2 .$$

With these definitions at hand, our generic definition of a Q-category M-SVM is:

Definition 1 (Generic model of M-SVM, Definition 4 in Guermeur, forthcoming) *Let* $((x_i, y_i))_{1 \leqslant i \leqslant m} \in (\mathcal{X} \times [\![1, Q]\!])^m$ *and* $\lambda \in \mathbb{R}_+^*$. *Let* $\xi \in \mathbb{R}^{Qm}$ *be a vector such that for* $(i,k) \in [\![1,m]\!] \times [\![1,Q]\!]$, ξ_{ik} *is its component of index* $(i-1)Q+k$, *with* $(\xi_{iy_i})_{1 \leqslant i \leqslant m} = 0_m$. *A* Q-category M-SVM *is a classifier obtained by solving a convex quadratic programming (QP) problem of the form*

$$\min_{h,\xi} J(h,\xi) = \|M\xi\|_p^p + \lambda \|h\|_{\mathcal{H}}^2$$

$$s.t. \begin{cases} \forall i \in [\![1,m]\!], \ \forall k \in [\![1,Q]\!] \setminus \{y_i\}, \ K_1 h_{y_i}(x_i) - h_k(x_i) \geqslant K_2 - \xi_{ik} \\ \forall i \in [\![1,m]\!], \ \forall (k,l) \in ([\![1,Q]\!] \setminus \{y_i\})^2, \ K_3 (\xi_{ik} - \xi_{il}) = 0 \\ \forall i \in [\![1,m]\!], \ \forall k \in [\![1,Q]\!] \setminus \{y_i\}, \ (2-p)\xi_{ik} \geqslant 0 \\ (1-K_1) \sum_{k=1}^{Q} h_k = 0, \end{cases}$$

where $p \in \{1,2\}$, $(K_1, K_3) \in \{0,1\}^2$, $K_2 \in \mathbb{R}_+^*$ *and the matrix* M *is such that* $\|M\xi\|_p$ *is a norm of* ξ.

Extending to matrices the notation used to designate the components of ξ and using δ to denote the Kronecker symbol, let us define the general term of $M^{(2)} \in \mathcal{M}_{Qm,Qm}(\mathbb{R})$ as:

$$m^{(2)}_{ik,jl} = (1 - \delta_{y_i,k}) \left(1 - \delta_{y_j,l}\right) \delta_{i,j} \left(\delta_{k,l} + \frac{\sqrt{Q}-1}{Q-1} \right).$$

This allows us to summarize the characteristics of the four M-SVMs in Table 1. The potential of the generic model is discussed in Guermeur (forthcoming).

3. The Software Package

MSVMpack includes a C application programming interface (API) and two command-line tools: one for training an M-SVM and one for making predictions with a trained M-SVM. The following discusses some algorithmic issues before presenting these tools and the API.

3.1 Training Algorithm

As in the bi-class case, an M-SVM is trained by solving the Wolfe dual of its instantiation of the QP problem in Definition 1. The corresponding dual variables α_{ik} are the Lagrange multipliers of the constraints of correct classification. The implemented QP algorithm is based on the Frank-Wolfe method (Frank and Wolfe, 1956), in which each step of the descent is obtained as the solution of a linear program (LP). The LP solver included in MSVMpack is `lp_solve` (Berkelaar et al., 2009). In order to make it possible to process large data sets, a decomposition method is applied and only a small subset of the data is considered in each iteration.

Let J_d be the dual objective function and let $\alpha = (\alpha_{ik})$ be a (feasible) solution of the dual problem obtained at some point of the training procedure. The quality of α is measured thanks to the computation of an upper bound $U(\alpha)$ on the optimum $J(h^*, \xi^*) = J_d(\alpha^*)$ that goes to this optimum. The stopping criterion is defined as a large enough value for $J_d(\alpha)/U(\alpha)$. In MSVMpack, the bound $U(\alpha)$ is obtained by solving the primal problem with h being the function associated with the current α. This partial optimization requires little computation except in the case of the M-SVM^2, for which another QP problem has to be solved. However, the computational burden may increase for a large number of classes (e.g., $Q > 20$).

3.2 Practical Use and Experiments

In its most simple form, the command line '`trainmsvm trainingdata -m WW`' is used to train an M-SVM, where the `-m` flag allows one to choose the type of M-SVM model according to Table 1. Then, this model can be applied to a test set by using '`predmsvm testdata`'. The complete list of options and parameters for these command-line tools can be found in the documentation or simply obtained by calling them without argument.

Table 2 shows a comparison of MSVMpack with other implementations of M-SVMs on a subset of the USPS database with 500 instances from 10 classes and the whole CB513 data set with 84119 instances from 3 classes. For the latter, the numbers reflect the average error and total times over a 5-fold cross validation, and the implementations that failed due to a lack of memory are not included in the Table. We refer the reader to the documentation for the details of the experimental setup and additional comparisons on other data sets.

3.3 Calling the Library from Other Programs

The "Developer's guide" section of the documentation presents the API reference and an example program including MSVMpack functionalities through this API. The library defines specific data structures for M-SVM models and data sets. It also provides wrapper functions, which act according to the M-SVM model type, for example, call the corresponding training function. The standard workflow for a train-and-test sequence is: call `MSVM_make_model()` to initialize the model; call `MSVM_make_dataset()` for each data set to load; call `MSVM_train()` to train the model; and call `MSVM_classify_set()` to test the trained classifier.

4. Ongoing and Future Developments

MSVMpack implements the four M-SVMs proposed in the literature. Current work focuses on the explicit implementation of our generic model of M-SVM, which will make it possible to study new machines thanks to a simple choice of the values of the hyperparameters M, p, and $(K_t)_{1 \leqslant t \leqslant 3}$. Future

Data set	M-SVM	Software	Test error	Training time	Testing time
USPS_500	WW	Spider (Matlab)	10.20 %	4m 19s	0.2s
$Q = 10$		BSVM (C++)	10.00 %	0.2s	0.1s
$m = 500$		MSVMpack (C)	10.40 %	2.5s	0.1s
$\mathcal{X} \subset \mathbb{R}^{256}$	CS	MCSVM (C)	9.80 %	0.5s	0.3s
test set:		MSVMpack	9.80 %	30s	0.1s
$m = 500$	LLW	SMSVM (R)	12.00 %	5m 58s	0.1s
		MSVMpack	11.40 %	1m 22s	0.1s
	MSVM^2	MSVMpack	12.00 %	22s	0.1s
CB513	WW	BSVM	23.96 %	9h 48m 40s	46m 29s
$Q = 3$		MSVMpack	23.72 %	1h 05m 11s	1m 51s
$m = 84119$	CS	MCSVM	23.55 %	22h 52m 10s	2h 08m 33s
$\mathcal{X} \subset \mathbb{Z}^{260}$		MSVMpack	23.63 %	1h 00m 36s	2m 06s
test:	LLW	MSVMpack	25.65 %	1h 14m 21s	2m 33s
5-fold CV	MSVM^2	MSVMpack	23.47 %	6h 44m 50s	2m 49s

Table 2: Relative performance of different M-SVM implementations on two data sets.

work will consider including automatic tuning procedures for the regularization parameter λ, and relaxing the hypothesis on the norm of the penalizer.

References

M. Berkelaar, K. Eikland, and P. Notebaert. *An Open Source (Mixed-Integer) Linear Programming System*, 2009. Software available at `http://lpsolve.sourceforge.net/`.

A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, Boston, 2004.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1–2):95–110, 1956.

Y. Guermeur. A generic model of multi-class support vector machine. *International Journal of Intelligent Information and Database Systems*, forthcoming.

Y. Guermeur and E. Monfrini. A quadratic loss multi-class SVM for which a radius-margin bound applies. *Informatica*, 22(1):73–96, 2011.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.

Journal of Machine Learning Research 12 (2011) 2297-2334 Submitted 9/10; Revised 3/11; Published 7/11

Proximal Methods for Hierarchical Sparse Coding

Rodolphe Jenatton[∗ †] RODOLPHE.JENATTON@INRIA.FR
Julien Mairal[∗†] JULIEN.MAIRAL@INRIA.FR
Guillaume Obozinski[†] GUILLAUME.OBOZINSKI@INRIA.FR
Francis Bach[†] FRANCIS.BACH@INRIA.FR
INRIA - WILLOW Project-Team
Laboratoire d'Informatique de l'Ecole Normale Supérieure (INRIA/ENS/CNRS UMR 8548)
23, avenue d'Italie
75214 Paris CEDEX 13, France

Editor: Tong Zhang

Abstract

Sparse coding consists in representing signals as sparse linear combinations of atoms selected from a dictionary. We consider an extension of this framework where the atoms are further assumed to be embedded in a tree. This is achieved using a recently introduced tree-structured sparse regularization norm, which has proven useful in several applications. This norm leads to regularized problems that are difficult to optimize, and in this paper, we propose efficient algorithms for solving them. More precisely, we show that the proximal operator associated with this norm is computable exactly via a dual approach that can be viewed as the composition of elementary proximal operators. Our procedure has a complexity linear, or close to linear, in the number of atoms, and allows the use of accelerated gradient techniques to solve the tree-structured sparse approximation problem at the same computational cost as traditional ones using the ℓ_1-norm. Our method is efficient and scales gracefully to millions of variables, which we illustrate in two types of applications: first, we consider *fixed* hierarchical dictionaries of wavelets to denoise natural images. Then, we apply our optimization tools in the context of *dictionary learning*, where learned dictionary elements naturally self-organize in a prespecified arborescent structure, leading to better performance in reconstruction of natural image patches. When applied to text documents, our method learns hierarchies of topics, thus providing a competitive alternative to probabilistic topic models.

Keywords: Proximal methods, dictionary learning, structured sparsity, matrix factorization

1. Introduction

Modeling signals as sparse linear combinations of atoms selected from a dictionary has become a popular paradigm in many fields, including signal processing, statistics, and machine learning. This line of research, also known as *sparse coding*, has witnessed the development of several well-founded theoretical frameworks (Tibshirani, 1996; Chen et al., 1998; Mallat, 1999; Tropp, 2004, 2006; Wainwright, 2009; Bickel et al., 2009) and the emergence of many efficient algorithmic tools

∗. These authors contributed equally.
†. Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach are now affiliated to INRIA - Sierra Project-Team. Julien Mairal is now with the Statistics Department of the University of California at Berkeley. When this work was performed all authors were affiliated to INRIA - Willow Project-Team.

(Efron et al., 2004; Nesterov, 2007; Beck and Teboulle, 2009; Wright et al., 2009; Needell and Tropp, 2009; Yuan et al., 2010).

In many applied settings, the structure of the problem at hand, such as, for example, the spatial arrangement of the pixels in an image, or the presence of variables corresponding to several levels of a given factor, induces relationships between dictionary elements. It is appealing to use this a priori knowledge about the problem *directly* to constrain the possible sparsity patterns. For instance, when the dictionary elements are partitioned into predefined groups corresponding to different types of features, one can enforce a similar block structure in the sparsity pattern—that is, allow only that either all elements of a group are part of the signal decomposition or that all are dismissed simultaneously (see Yuan and Lin, 2006; Stojnic et al., 2009).

This example can be viewed as a particular instance of *structured sparsity*, which has been lately the focus of a large amount of research (Baraniuk et al., 2010; Zhao et al., 2009; Huang et al., 2009; Jacob et al., 2009; Jenatton et al., 2009; Micchelli et al., 2010). In this paper, we concentrate on a specific form of structured sparsity, which we call *hierarchical sparse coding*: the dictionary elements are assumed to be embedded in a directed tree $\mathcal{T}$, and the sparsity patterns are constrained to form a *connected and rooted subtree* of $\mathcal{T}$ (Donoho, 1997; Baraniuk, 1999; Baraniuk et al., 2002, 2010; Zhao et al., 2009; Huang et al., 2009). This setting extends more generally to a forest of directed trees.[1]

In fact, such a hierarchical structure arises in many applications. Wavelet decompositions lend themselves well to this tree organization because of their multiscale structure, and benefit from it for image compression and denoising (Shapiro, 1993; Crouse et al., 1998; Baraniuk, 1999; Baraniuk et al., 2002, 2010; He and Carin, 2009; Zhao et al., 2009; Huang et al., 2009). In the same vein, edge filters of natural image patches can be represented in an arborescent fashion (Zoran and Weiss, 2009). Imposing these sparsity patterns has further proven useful in the context of hierarchical variable selection, for example, when applied to kernel methods (Bach, 2008), to log-linear models for the selection of potential orders (Schmidt and Murphy, 2010), and to bioinformatics, to exploit the tree structure of gene networks for multi-task regression (Kim and Xing, 2010). Hierarchies of latent variables, typically used in neural networks and deep learning architectures (see Bengio, 2009, and references therein) have also emerged as a natural structure in several applications, notably to model text documents. In particular, in the context of *topic models* (Blei et al., 2003), a hierarchical model of latent variables based on Bayesian non-parametric methods has been proposed by Blei et al. (2010) to model hierarchies of topics.

To perform hierarchical sparse coding, our work builds upon the approach of Zhao et al. (2009) who first introduced a sparsity-inducing norm Ω leading to this type of tree-structured sparsity pattern. We tackle the resulting nonsmooth convex optimization problem with proximal methods (e.g., Nesterov, 2007; Beck and Teboulle, 2009; Wright et al., 2009; Combettes and Pesquet, 2010) and we show in this paper that its key step, the computation of the *proximal operator*, can be solved exactly with a complexity linear, or close to linear, in the number of dictionary elements—that is, with the same complexity as for classical ℓ_1-sparse decomposition problems (Tibshirani, 1996; Chen et al., 1998). Concretely, given an m-dimensional signal $\mathbf{x}$ along with a dictionary $\mathbf{D} = [\mathbf{d}^1, \ldots, \mathbf{d}^p] \in \mathbb{R}^{m \times p}$ composed of p atoms, the optimization problem at the core of our paper can be written as

$$\min_{\alpha \in \mathbb{R}^p} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\alpha\|_2^2 + \lambda\Omega(\alpha), \text{ with } \lambda \geq 0.$$

1. A tree is defined as a connected graph that contains no cycle (see Ahuja et al., 1993).

In this formulation, the sparsity-inducing norm Ω encodes a hierarchical structure among the atoms of $\mathbf{D}$, where this structure is assumed to be known beforehand. The precise meaning of *hierarchical structure* and the definition of Ω will be made more formal in the next sections. A particular instance of this problem—known as the *proximal problem*—is central to our analysis and concentrates on the case where the dictionary $\mathbf{D}$ is orthogonal.

In addition to a speed benchmark that evaluates the performance of our proposed approach in comparison with other convex optimization techniques, two types of applications and experiments are considered. First, we consider settings where the dictionary is fixed and given a priori, corresponding for instance to a basis of wavelets for the denoising of natural images. Second, we show how one can take advantage of this hierarchical sparse coding in the context of dictionary learning (Olshausen and Field, 1997; Aharon et al., 2006; Mairal et al., 2010a), where the dictionary is learned to adapt to the predefined tree structure. This extension of dictionary learning is notably shown to share interesting connections with hierarchical probabilistic topic models.

To summarize, the contributions of this paper are threefold:

- We show that the proximal operator for a tree-structured sparse regularization can be computed exactly in a finite number of operations using a dual approach. Our approach is equivalent to computing a particular sequence of elementary proximal operators, and has a complexity linear, or close to linear, in the number of variables. Accelerated gradient methods (e.g., Nesterov, 2007; Beck and Teboulle, 2009; Combettes and Pesquet, 2010) can then be applied to solve large-scale tree-structured sparse decomposition problems at the same computational cost as traditional ones using the ℓ_1-norm.

- We propose to use this regularization scheme to learn dictionaries embedded in a tree, which, to the best of our knowledge, has not been done before in the context of structured sparsity.

- Our method establishes a bridge between hierarchical dictionary learning and hierarchical topic models (Blei et al., 2010), which builds upon the interpretation of topic models as multinomial PCA (Buntine, 2002), and can learn similar hierarchies of topics. This point is discussed in Sections 5.5 and 6.

Note that this paper extends a shorter version published in the proceedings of the international conference of machine learning (Jenatton et al., 2010).

1.1 Notation

Vectors are denoted by bold lower case letters and matrices by upper case ones. We define for $q \geq 1$ the ℓ_q-norm of a vector $\mathbf{x}$ in $\mathbb{R}^m$ as $\|\mathbf{x}\|_q \triangleq (\sum_{i=1}^m |\mathbf{x}_i|^q)^{1/q}$, where $\mathbf{x}_i$ denotes the i-th coordinate of $\mathbf{x}$, and $\|\mathbf{x}\|_\infty \triangleq \max_{i=1,\dots,m} |\mathbf{x}_i| = \lim_{q\to\infty} \|\mathbf{x}\|_q$. We also define the ℓ_0-pseudo-norm as the number of nonzero elements in a vector:[2] $\|\mathbf{x}\|_0 \triangleq \#\{i \text{ s.t. } \mathbf{x}_i \neq 0\} = \lim_{q\to 0^+}(\sum_{i=1}^m |\mathbf{x}_i|^q)$. We consider the Frobenius norm of a matrix $\mathbf{X}$ in $\mathbb{R}^{m\times n}$: $\|\mathbf{X}\|_\text{F} \triangleq (\sum_{i=1}^m \sum_{j=1}^n \mathbf{X}_{ij}^2)^{1/2}$, where $\mathbf{X}_{ij}$ denotes the entry of $\mathbf{X}$ at row i and column j. Finally, for a scalar y, we denote $(y)_+ \triangleq \max(y,0)$.

The rest of this paper is organized as follows: Section 2 presents related work and the problem we consider. Section 3 is devoted to the algorithm we propose, and Section 4 introduces the

2. Note that it would be more proper to write $\|\mathbf{x}\|_0^0$ instead of $\|\mathbf{x}\|_0$ to be consistent with the traditional notation $\|\mathbf{x}\|_q$. However, for the sake of simplicity, we will keep this notation unchanged in the rest of the paper.

dictionary learning framework and shows how it can be used with tree-structured norms. Section 5 presents several experiments demonstrating the effectiveness of our approach and Section 6 concludes the paper.

2. Problem Statement and Related Work

Let us consider an input signal of dimension m, typically an image described by its m pixels, which we represent by a vector $\mathbf{x}$ in $\mathbb{R}^m$. In traditional sparse coding, we seek to approximate this signal by a sparse linear combination of atoms, or dictionary elements, represented here by the columns of a matrix $\mathbf{D} \triangleq [\mathbf{d}^1,\dots,\mathbf{d}^p]$ in $\mathbb{R}^{m\times p}$. This can equivalently be expressed as $\mathbf{x} \approx \mathbf{D}\alpha$ for some sparse vector α in $\mathbb{R}^p$, that is, such that the number of nonzero coefficients $\|\alpha\|_0$ is small compared to p. The vector α is referred to as the code, or decomposition, of the signal $\mathbf{x}$.

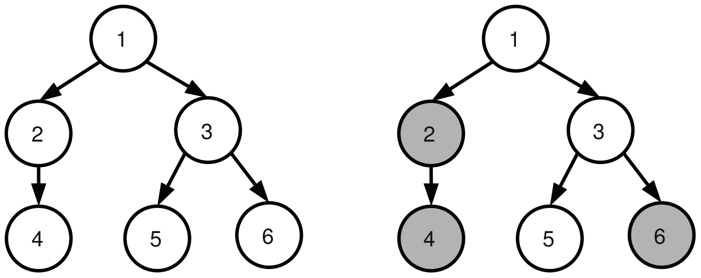


Figure 1: Example of a tree $\mathcal{T}$ when $p = 6$. With the rule we consider for the nonzero patterns, if we have $\alpha_5 \neq 0$, we must also have $\alpha_k \neq 0$ for k in ancestors$(5) = \{1,3,5\}$.

In the rest of the paper, we focus on specific sets of nonzero coefficients—or simply, nonzero patterns—for the decomposition vector α. In particular, we assume that we are given a tree[3] $\mathcal{T}$ whose p nodes are indexed by j in $\{1,\dots,p\}$. We want the nonzero patterns of α to form a *connected and rooted subtree* of $\mathcal{T}$; in other words, if ancestors$(j) \subseteq \{1,\dots,p\}$ denotes the set of indices corresponding to the ancestors[4] of the node j in $\mathcal{T}$ (see Figure 1), the vector α obeys the following rule

$$\alpha_j \neq 0 \Rightarrow [\alpha_k \neq 0 \text{ for all } k \text{ in ancestors}(j)]. \tag{1}$$

Informally, we want to exploit the structure of $\mathcal{T}$ in the following sense: the decomposition of any signal $\mathbf{x}$ can involve a dictionary element $\mathbf{d}^j$ *only if the ancestors of* $\mathbf{d}^j$ *in the tree* $\mathcal{T}$ *are themselves part of the decomposition*.

We now review previous work that has considered the sparse approximation problem with tree-structured constraints (1). Similarly to traditional sparse coding, there are basically two lines of research, that either (A) deal with nonconvex and combinatorial formulations that are in general computationally intractable and addressed with greedy algorithms, or (B) concentrate on convex relaxations solved with convex programming methods.

3. Our analysis straightforwardly extends to the case of a forest of trees; for simplicity, we consider a single tree $\mathcal{T}$.
4. We consider that the set of ancestors of a node also contains the node itself.

2.1 Nonconvex Approaches

For a given sparsity level $s \geq 0$ (number of nonzero coefficients), the following nonconvex problem

$$\min_{\substack{\alpha \in \mathbb{R}^p \\ \|\alpha\|_0 \leq s}} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\alpha\|_2^2 \quad \text{such that condition (1) is respected,} \tag{2}$$

has been tackled by Baraniuk (1999); Baraniuk et al. (2002) in the context of wavelet approximations with a greedy procedure. A penalized version of problem (2) (that adds $\lambda\|\alpha\|_0$ to the objective function in place of the constraint $\|\alpha\|_0 \leq s$) has been considered by Donoho (1997), while studying the more general problem of best approximation from dyadic partitions (see Section 6 in Donoho, 1997). Interestingly, the algorithm we introduce in Section 3 shares conceptual links with the dynamic-programming approach of Donoho (1997), which was also used by Baraniuk et al. (2010), in the sense that the same order of traversal of the tree is used in both procedures. We investigate more thoroughly the relations between our algorithm and this approach in Appendix A.

Problem (2) has been further studied for structured compressive sensing (Baraniuk et al., 2010), with a greedy algorithm that builds upon Needell and Tropp (2009). Finally, Huang et al. (2009) have proposed a formulation related to (2), with a nonconvex penalty based on an information-theoretic criterion.

2.2 Convex Approach

We now turn to a convex reformulation of the constraint (1), which is the starting point for the convex optimization tools we develop in Section 3.

2.2.1 HIERARCHICAL SPARSITY-INDUCING NORMS

Condition (1) can be equivalently expressed by its contrapositive, thus leading to an intuitive way of penalizing the vector α to obtain tree-structured nonzero patterns. More precisely, defining descendants$(j) \subseteq \{1,\dots,p\}$ analogously to ancestors(j) for j in $\{1,\dots,p\}$, condition (1) amounts to saying that *if a dictionary element is not used in the decomposition, its descendants in the tree should not be used either*. Formally, this can be formulated as:

$$\alpha_j = 0 \Rightarrow [\alpha_k = 0 \text{ for all } k \text{ in descendants}(j)]. \tag{3}$$

From now on, we denote by $\mathcal{G}$ the set defined by $\mathcal{G} \triangleq \{\text{descendants}(j); j \in \{1,\dots,p\}\}$, and refer to each member g of $\mathcal{G}$ as a *group* (Figure 2). To obtain a decomposition with the desired property (3), one can naturally penalize the number of groups g in $\mathcal{G}$ that are "involved" in the decomposition of $\mathbf{x}$, that is, that record at least one nonzero coefficient of α:

$$\sum_{g \in \mathcal{G}} \delta^g, \text{ with } \delta^g \triangleq \begin{cases} 1 & \text{if there exists } j \in g \text{ such that } \alpha_j \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

While this intuitive penalization is nonconvex (and not even continuous), a convex proxy has been introduced by Zhao et al. (2009). It was further considered by Bach (2008); Kim and Xing (2010); Schmidt and Murphy (2010) in several different contexts. For any vector $\alpha \in \mathbb{R}^p$, let us define

$$\Omega(\alpha) \triangleq \sum_{g \in \mathcal{G}} \omega_g \|\alpha_{|g}\|,$$

where $\alpha_{|g}$ is the vector of size p whose coordinates are equal to those of α for indices in the set g, and to 0 otherwise.[5] The notation $\|.\|$ stands in practice either for the ℓ_2- or ℓ_∞-norm, and $(\omega_g)_{g\in\mathcal{G}}$ denotes some positive weights.[6] As analyzed by Zhao et al. (2009) and Jenatton et al. (2009), when penalizing by Ω, some of the vectors $\alpha_{|g}$ are set to zero for some $g \in \mathcal{G}$.[7] Therefore, the components of α corresponding to some complete subtrees of $\mathcal{T}$ are set to zero, which exactly matches condition (3), as illustrated in Figure 2.

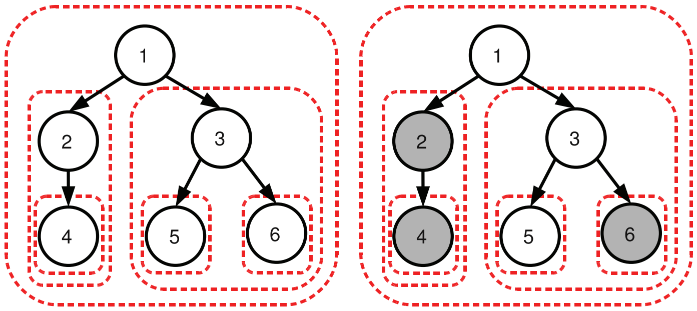


Figure 2: Left: example of a tree-structured set of groups $\mathcal{G}$ (dashed contours in red), corresponding to a tree $\mathcal{T}$ with $p = 6$ nodes represented by black circles. Right: example of a sparsity pattern induced by the tree-structured norm corresponding to $\mathcal{G}$: the groups $\{2,4\}, \{4\}$ and $\{6\}$ are set to zero, so that the corresponding nodes (in gray) that form subtrees of $\mathcal{T}$ are removed. The remaining nonzero variables $\{1,3,5\}$ form a rooted and connected subtree of $\mathcal{T}$. This sparsity pattern obeys the following equivalent rules: (i) if a node is selected, the same goes for all its ancestors. (ii) if a node is not selected, then its descendant are not selected.

Note that although we presented for simplicity this hierarchical norm in the context of a single tree with a single element at each node, it can easily be extended to the case of forests of trees, and/or trees containing arbitrary numbers of dictionary elements at each node (with nodes possibly containing no dictionary element). More broadly, this formulation can be extended with the notion of *tree-structured* groups, which we now present:

Definition 1 (Tree-structured set of groups.)
A set of groups $\mathcal{G} \triangleq \{g\}_{g\in\mathcal{G}}$ is said to be tree-structured in $\{1,\dots,p\}$, if $\bigcup_{g\in\mathcal{G}} g = \{1,\dots,p\}$ and if for all $g,h \in \mathcal{G}$, $(g\cap h \neq \emptyset) \Rightarrow (g \subseteq h$ or $h \subseteq g)$. For such a set of groups, there exists a (non-unique) total order relation $\preceq$ such that:

$$g \preceq h \;\Rightarrow\; \big\{ g \subseteq h \quad \text{or} \quad g \cap h = \emptyset \big\}.$$

Given such a tree-structured set of groups $\mathcal{G}$ and its associated norm Ω, we are interested throughout the paper in the following hierarchical sparse coding problem,

$$\min_{\alpha\in\mathbb{R}^p} f(\alpha) + \lambda\Omega(\alpha), \tag{5}$$

5. Note the difference with the notation α_g, which is often used in the literature on structured sparsity, where α_g is a vector of size $|g|$.
6. For a complete definition of Ω for any ℓ_q-norm, a discussion of the choice of q, and a strategy for choosing the weights ω_g (see Zhao et al., 2009; Kim and Xing, 2010).
7. It has been further shown by Bach (2010) that the convex envelope of the nonconvex function of Equation (4) is in fact Ω with $\|.\|$ being the ℓ_∞-norm.

where Ω is the tree-structured norm we have previously introduced, the non-negative scalar λ is a regularization parameter controlling the sparsity of the solutions of (5), and f a smooth convex loss function (see Section 3 for more details about the smoothness assumptions on f). In the rest of the paper, we will mostly use the square loss $f(\alpha) = \frac{1}{2}\|\mathbf{x} - \mathbf{D}\alpha\|_2^2$, with a dictionary $\mathbf{D}$ in $\mathbb{R}^{m\times p}$, but the formulation of Equation (5) extends beyond this context. In particular one can choose f to be the logistic loss, which is commonly used for classification problems (e.g., see Hastie et al., 2009).

Before turning to optimization methods for the hierarchical sparse coding problem, we consider a particular instance. The *sparse group Lasso* was recently considered by Sprechmann et al. (2010) and Friedman et al. (2010) as an extension of the group Lasso of Yuan and Lin (2006). To induce sparsity both groupwise and within groups, Sprechmann et al. (2010) and Friedman et al. (2010) add an ℓ_1 term to the regularization of the group Lasso, which given a partition $\mathcal{P}$ of $\{1,\dots,p\}$ in disjoint groups yields a regularized problem of the form

$$\min_{\alpha\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\alpha\|_2^2 + \lambda \sum_{g\in\mathcal{P}} \|\alpha_{|g}\|_2 + \lambda'\|\alpha\|_1.$$

Since $\mathcal{P}$ is a partition, the set of groups in $\mathcal{P}$ and the singletons form together a tree-structured set of groups according to definition 1 and the algorithm we will develop is therefore applicable to this problem.

2.2.2 OPTIMIZATION FOR HIERARCHICAL SPARSITY-INDUCING NORMS

While generic approaches like interior-point methods (Boyd and Vandenberghe, 2004) and subgradient descent schemes (Bertsekas, 1999) might be used to deal with the nonsmooth norm Ω, several dedicated procedures have been proposed.

In Zhao et al. (2009), a boosting-like technique is used, with a path-following strategy in the specific case where $\|.\|$ is the ℓ_∞-norm. Based on the variational equality

$$\|\mathbf{u}\|_1 = \min_{\mathbf{z}\in\mathbb{R}_+^p} \frac{1}{2}\Big[\sum_{j=1}^{p} \frac{\mathbf{u}_j^2}{\mathbf{z}_j} + \mathbf{z}_j\Big], \tag{6}$$

Kim and Xing (2010) follow a reweighted least-square scheme that is well adapted to the square loss function. To the best of our knowledge, a formulation of this type is however not available when $\|.\|$ is the ℓ_∞-norm. In addition it requires an appropriate smoothing to become provably convergent. The same approach is considered by Bach (2008), but built upon an active-set strategy. Other proposed methods consist of a projected gradient descent with approximate projections onto the ball $\{\mathbf{u}\in\mathbb{R}^p; \Omega(\mathbf{u}) \le \lambda\}$ (Schmidt and Murphy, 2010), and an augmented-Lagrangian based technique (Sprechmann et al., 2010) for solving a particular case with two-level hierarchies.

While the previously listed first-order approaches are (1) loss-function dependent, and/or (2) not guaranteed to achieve optimal convergence rates, and/or (3) not able to yield sparse solutions without a somewhat arbitrary post-processing step, we propose to resort to proximal methods[8] that do not suffer from any of these drawbacks.

8. Note that the authors of Chen et al. (2010) have considered proximal methods for general group structure $\mathcal{G}$ when $\|.\|$ is the ℓ_2-norm; due to a smoothing of the regularization term, the convergence rate they obtained is suboptimal.

3. Optimization

We begin with a brief introduction to proximal methods, necessary to present our contributions. From now on, we assume that f is convex and continuously differentiable with Lipschitz-continuous gradient. It is worth mentioning that there exist various proximal schemes in the literature that differ in their settings (e.g., batch versus stochastic) and/or the assumptions made on f. For instance, the material we develop in this paper could also be applied to online/stochastic frameworks (Duchi and Singer, 2009; Hu et al., 2009; Xiao, 2010) and to possibly nonsmooth functions f (e.g., Duchi and Singer, 2009; Xiao, 2010; Combettes and Pesquet, 2010, and references therein). Finally, most of the technical proofs of this section are presented in Appendix B for readability.

3.1 Proximal Operator for the Norm Ω

Proximal methods have drawn increasing attention in the signal processing (e.g., Becker et al., 2009; Wright et al., 2009; Combettes and Pesquet, 2010, and numerous references therein) and the machine learning communities (e.g., Bach et al., 2011, and references therein), especially because of their convergence rates (optimal for the class of first-order techniques) and their ability to deal with large nonsmooth convex problems (e.g., Nesterov, 2007; Beck and Teboulle, 2009). In a nutshell, these methods can be seen as a natural extension of gradient-based techniques when the objective function to minimize has a nonsmooth part. Proximal methods are iterative procedures. The simplest version of this class of methods linearizes at each iteration the function f around the current estimate $\hat{\alpha}$, and this estimate is updated as the (unique by strong convexity) solution of the *proximal problem*, defined as follows:

$$\min_{\alpha\in\mathbb{R}^p} f(\hat{\alpha}) + (\alpha-\hat{\alpha})^\top \nabla f(\hat{\alpha}) + \lambda\Omega(\alpha) + \frac{L}{2}\|\alpha-\hat{\alpha}\|_2^2.$$

The quadratic term keeps the update in a neighborhood where f is close to its linear approximation, and $L>0$ is a parameter which is an upper bound on the Lipschitz constant of ∇f. This problem can be equivalently rewritten as:

$$\min_{\alpha\in\mathbb{R}^p} \frac{1}{2}\Big\|\alpha - \big(\hat{\alpha} - \frac{1}{L}\nabla f(\hat{\alpha})\big)\Big\|_2^2 + \frac{\lambda}{L}\Omega(\alpha).$$

Solving *efficiently* and *exactly* this problem is crucial to enjoy the fast convergence rates of proximal methods. In addition, when the nonsmooth term Ω is not present, the previous proximal problem exactly leads to the standard gradient update rule. More generally, we define the *proximal operator*:

Definition 2 (Proximal Operator)
The proximal operator associated with our regularization term $\lambda\Omega$, which we denote by $Prox_{\lambda\Omega}$, is the function that maps a vector $\mathbf{u}\in\mathbb{R}^p$ to the unique solution of

$$\min_{\mathbf{v}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \lambda\Omega(\mathbf{v}). \tag{7}$$

This operator was initially introduced by Moreau (1962) to generalize the projection operator onto a convex set. What makes proximal methods appealing for solving sparse decomposition problems is that this operator can be often computed in closed-form. For instance,

- When Ω is the ℓ_1-norm—that is, $\Omega(\mathbf{u}) = \|\mathbf{u}\|_1$, the proximal operator is the well-known elementwise soft-thresholding operator,

$$\forall j \in \{1,\dots,p\},\ \ \mathbf{u}_j \mapsto \operatorname{sign}(\mathbf{u}_j)(|\mathbf{u}_j| - \lambda)_+ \ = \begin{cases} 0 & \text{if } |\mathbf{u}_j| \leq \lambda \\ \operatorname{sign}(\mathbf{u}_j)(|\mathbf{u}_j| - \lambda) & \text{otherwise.} \end{cases}$$

- When Ω is a group-Lasso penalty with ℓ_2-norms—that is, $\Omega(\mathbf{u}) = \sum_{g\in\mathcal{G}} \|\mathbf{u}_{|g}\|_2$, with $\mathcal{G}$ being a partition of $\{1,\dots,p\}$, the proximal problem is *separable* in every group, and the solution is a generalization of the soft-thresholding operator to groups of variables:

$$\forall g \in \mathcal{G}\ \ , \mathbf{u}_{|g} \mapsto \mathbf{u}_{|g} - \Pi_{\|.\|_2\leq\lambda}[\mathbf{u}_{|g}] = \begin{cases} 0 & \text{if } \|\mathbf{u}_{|g}\|_2 \leq \lambda \\ \frac{\|\mathbf{u}_{|g}\|_2-\lambda}{\|\mathbf{u}_{|g}\|_2}\mathbf{u}_{|g} & \text{otherwise,} \end{cases}$$

  where $\Pi_{\|.\|_2\leq\lambda}$ denotes the orthogonal projection onto the ball of the ℓ_2-norm of radius λ.

- When Ω is a group-Lasso penalty with ℓ_∞-norms—that is, $\Omega(\mathbf{u}) = \sum_{g\in\mathcal{G}} \|\mathbf{u}_{|g}\|_\infty$, the solution is also a group-thresholding operator:

$$\forall g \in \mathcal{G},\ \ \mathbf{u}_{|g} \mapsto \mathbf{u}_{|g} - \Pi_{\|.\|_1\leq\lambda}[\mathbf{u}_{|g}],$$

  where $\Pi_{\|.\|_1\leq\lambda}$ denotes the orthogonal projection onto the ℓ_1-ball of radius λ, which can be solved in $O(p)$ operations (Brucker, 1984; Maculan and Galdino de Paula, 1989). Note that when $\|\mathbf{u}_{|g}\|_1 \leq \lambda$, we have a group-thresholding effect, with $\mathbf{u}_{|g} - \Pi_{\|.\|_1\leq\lambda}[\mathbf{u}_{|g}] = 0$.

More generally, a classical result (see, e.g., Combettes and Pesquet, 2010; Wright et al., 2009) says that the proximal operator for a norm $\|.\|$ can be computed as the residual of the projection of a vector onto a ball of the dual-norm denoted by $\|.\|_*$, and defined for any vector κ in $\mathbb{R}^p$ by $\|\kappa\|_* \triangleq \max_{\|\mathbf{z}\|\leq 1} \mathbf{z}^\top\kappa$.[9] This is a classical duality result for proximal operators leading to the different closed forms we have just presented. We have indeed that $\text{Prox}_{\lambda\|.\|_2} = \text{Id} - \Pi_{\|.\|_2\leq\lambda}$ and $\text{Prox}_{\lambda\|.\|_\infty} = \text{Id} - \Pi_{\|.\|_1\leq\lambda}$, where Id stands for the identity operator. Obtaining closed forms is, however, not possible anymore as soon as some groups in $\mathcal{G}$ overlap, which is always the case in our hierarchical setting with tree-structured groups.

3.2 A Dual Formulation of the Proximal Problem

We now show that Equation (7) can be solved using a dual approach, as described in the following lemma. The result relies on conic duality (Boyd and Vandenberghe, 2004), and does not make any assumption on the choice of the norm $\|.\|$:

Lemma 3 (Dual of the proximal problem)
Let $\mathbf{u} \in \mathbb{R}^p$ *and let us consider the problem*

$$\max_{\xi\in\mathbb{R}^{p\times|\mathcal{G}|}} -\frac{1}{2}\Big[\Big\|\mathbf{u} - \sum_{g\in\mathcal{G}} \xi^g\Big\|_2^2 - \|\mathbf{u}\|_2^2\Big] \qquad (8)$$
$$s.t.\ \forall g \in \mathcal{G},\ \|\xi^g\|_* \leq \lambda\omega_g \text{ and } \xi^g_j = 0 \text{ if } j \notin g,$$

9. It is easy to show that the dual norm of the ℓ_2-norm is the ℓ_2-norm itself. The dual norm of the ℓ_∞ is the ℓ_1-norm.

where $\xi = (\xi^g)_{g\in\mathcal{G}}$ and ξ^g_j denotes the j-th coordinate of the vector ξ^g in $\mathbb{R}^p$. Then, problems (7) and (8) are dual to each other and strong duality holds. In addition, the pair of primal-dual variables $\{\mathbf{v},\xi\}$ is optimal if and only if ξ is a feasible point of the optimization problem (8), and

$$\mathbf{v} = \mathbf{u} - \textstyle\sum_{g\in\mathcal{G}} \xi^g \quad \text{and} \quad \forall g \in \mathcal{G},\ \xi^g = \Pi_{\|.\|_* \le \lambda\omega_g}(\mathbf{v}_{|g} + \xi^g), \tag{9}$$

where we denote by $\Pi_{\|.\|_* \le \lambda\omega_g}$ the orthogonal projection onto the ball $\{\kappa \in \mathbb{R}^p;\ \|\kappa\|_* \le \lambda\omega_g\}$.

Note that we focus here on specific tree-structured groups, but the previous lemma is valid regardless of the nature of $\mathcal{G}$. The rationale of introducing such a dual formulation is to consider an equivalent problem to (7) that removes the issue of overlapping groups at the cost of a larger number of variables. In Equation (7), one is indeed looking for a vector $\mathbf{v}$ of size p, whereas one is considering a matrix ξ in $\mathbb{R}^{p\times|\mathcal{G}|}$ in Equation (8) with $\sum_{g\in\mathcal{G}}|g|$ nonzero entries, but with separable (convex) constraints for each of its columns.

This specific structure makes it possible to use block coordinate ascent (Bertsekas, 1999). Such a procedure is presented in Algorithm 1. It optimizes sequentially Equation (8) with respect to the variable ξ^g, while keeping fixed the other variables ξ^h, for $h \neq g$. It is easy to see from Equation (8) that such an update of a column ξ^g, for a group g in $\mathcal{G}$, amounts to computing the orthogonal projection of the vector $\mathbf{u}_{|g} - \sum_{h\neq g}\xi^h_{|g}$ onto the ball of radius $\lambda\omega_g$ of the dual norm $\|.\|_*$.

Algorithm 1 Block coordinate ascent in the dual

```
Inputs: $\mathbf{u} \in \mathbb{R}^p$ and set of groups $\mathcal{G}$.
Outputs: $(\mathbf{v}, \xi)$ (primal-dual solutions).
Initialization: $\xi = \mathbf{0}$.
while ( maximum number of iterations not reached ) do
  for $g \in \mathcal{G}$ do
    $\xi^g \leftarrow \Pi_{\|.\|_* \le \lambda\omega_g}\big(\big[\mathbf{u} - \sum_{h\neq g}\xi^h\big]_{|g}\big)$.
  end for
end while
$\mathbf{v} \leftarrow \mathbf{u} - \sum_{g\in\mathcal{G}}\xi^g$.
```

3.3 Convergence in One Pass

In general, Algorithm 1 is not guaranteed to solve exactly Equation (7) in a finite number of iterations. However, when $\|.\|$ is the ℓ_2- or ℓ_∞-norm, and provided that the groups in $\mathcal{G}$ are appropriately ordered, we now prove that only *one pass* of Algorithm 1, that is, only one iteration over all groups, is sufficient to obtain the exact solution of Equation (7). This result constitutes the main technical contribution of the paper and is the key for the efficiency of our procedure.

Before stating this result, we need to introduce a lemma showing that, given two nested groups g,h such that $g \subseteq h \subseteq \{1,\dots,p\}$, if ξ^g is updated before ξ^h in Algorithm 1, then the optimality condition for ξ^g is not perturbed by the update of ξ^h.

Lemma 4 (Projections with nested groups)
Let $\|.\|$ denote either the ℓ_2- or ℓ_∞-norm, and g and h be two nested groups—that is, $g \subseteq h \subseteq \{1,\dots,p\}$. Let $\mathbf{u}$ be a vector in $\mathbb{R}^p$, and let us consider the successive projections

$$\xi^g \triangleq \Pi_{\|.\|_* \le t_g}(\mathbf{u}_{|g}) \ \text{ and } \ \xi^h \triangleq \Pi_{\|.\|_* \le t_h}(\mathbf{u}_{|h} - \xi^g),$$

with $t_g, t_h > 0$. Let us introduce $\mathbf{v} = \mathbf{u} - \xi^g - \xi^h$. The following relationships hold

$$\xi^g = \Pi_{\|.\|_* \leq t_g}(\mathbf{v}_{|g} + \xi^g) \quad \textit{and} \quad \xi^h = \Pi_{\|.\|_* \leq t_h}(\mathbf{v}_{|h} + \xi^h).$$

The previous lemma establishes the convergence in one pass of Algorithm 1 in the case where $\mathcal{G}$ only contains two nested groups $g \subseteq h$, provided that ξ^g is computed before ξ^h. Let us illustrate this fact more concretely. After initializing ξ^g and ξ^h to zero, Algorithm 1 first updates ξ^g with the formula $\xi^g \leftarrow \Pi_{\|.\|_* \leq \lambda\omega_g}(\mathbf{u}_{|g})$, and then performs the following update: $\xi^h \leftarrow \Pi_{\|.\|_* \leq \lambda\omega_h}(\mathbf{u}_{|h} - \xi^g)$ (where we have used that $\xi^g = \xi^g_{|h}$ since $g \subseteq h$). We are now in position to apply Lemma 4 which states that the primal/dual variables $\{\mathbf{v}, \xi^g, \xi^h\}$ satisfy the optimality conditions (9), as described in Lemma 3. In only one pass over the groups $\{g, h\}$, we have in fact reached a solution of the dual formulation presented in Equation (8), and in particular, the solution of the proximal problem (7).

In the following proposition, this lemma is extended to general tree-structured sets of groups $\mathcal{G}$:

Proposition 5 (Convergence in one pass)
Suppose that the groups in $\mathcal{G}$ are ordered according to the total order relation $\preceq$ of Definition 1, and that the norm $\|.\|$ is either the ℓ_2- or ℓ_∞-norm. Then, after initializing ξ to $\mathbf{0}$, a single pass of Algorithm 1 over $\mathcal{G}$ with the order $\preceq$ yields the solution of the proximal problem (7).

Proof The proof largely relies on Lemma 4 and proceeds by induction. By definition of Algorithm 1, the feasibility of ξ is always guaranteed. We consider the following induction hypothesis

$$\mathcal{H}(h) \triangleq \big\{\forall g \preceq h, \text{ it holds that } \xi^g = \Pi_{\|.\|_* \leq \lambda\omega_g}([\mathbf{u} - \textstyle\sum_{g' \preceq h} \xi^{g'}]_{|g} + \xi^g)\big\}.$$

Since the dual variables ξ are initially equal to zero, the summation over $g' \preceq h$, $g' \neq g$ is equivalent to a summation over $g' \neq g$. We initialize the induction with the first group in $\mathcal{G}$, that, by definition of $\preceq$, does not contain any other group. The first step of Algorithm 1 easily shows that the induction hypothesis $\mathcal{H}$ is satisfied for this first group.

We now assume that $\mathcal{H}(h)$ is true and consider the next group h', $h \preceq h'$, in order to prove that $\mathcal{H}(h')$ is also satisfied. We have for each group $g \subseteq h$,

$$\xi^g = \Pi_{\|.\|_* \leq \lambda\omega_g}([\mathbf{u} - \textstyle\sum_{g' \preceq h} \xi^{g'}]_{|g} + \xi^g) = \Pi_{\|.\|_* \leq \lambda\omega_g}([\mathbf{u} - \sum_{g' \preceq h} \xi^{g'} + \xi^g]_{|g}).$$

Since $\xi^g_{|h'} = \xi^g$ for $g \subseteq h'$, we have

$$[\mathbf{u} - \textstyle\sum_{g' \preceq h} \xi^{g'}]_{|h'} = [\mathbf{u} - \sum_{g' \preceq h} \xi^{g'}]_{|h'} + \xi^g - \xi^g = [\mathbf{u} - \sum_{g' \preceq h} \xi^{g'} + \xi^g]_{|h'} - \xi^g,$$

and following the update rule for the group h',

$$\xi^{h'} = \Pi_{\|.\|_* \leq \lambda\omega_{h'}}([\mathbf{u} - \textstyle\sum_{g' \preceq h} \xi^{g'}]_{|h'}) = \Pi_{\|.\|_* \leq \lambda\omega_{h'}}([\mathbf{u} - \sum_{g' \preceq h} \xi^{g'} + \xi^g]_{|h'} - \xi^g).$$

At this point, we can apply Lemma 4 for each group $g \subseteq h$, which proves that the induction hypothesis $\mathcal{H}(h')$ is true. Let us introduce $\mathbf{v} \triangleq \mathbf{u} - \sum_{g \in \mathcal{G}} \xi^g$. We have shown that for all g in $\mathcal{G}$, $\xi^g = \Pi_{\|.\|_* \leq \lambda\omega_g}(\mathbf{v}_{|g} + \xi^g)$. As a result, the pair $\{\mathbf{v}, \xi\}$ satisfies the optimality conditions (9) of problem (8). Therefore, after one complete pass over $g \in \mathcal{G}$, the primal/dual pair $\{\mathbf{v}, \xi\}$ is optimal, and in particular, $\mathbf{v}$ is the solution of problem (7). ■

Using conic duality, we have derived a dual formulation of the proximal operator, leading to Algorithm 1 which is generic and works for any norm $\|.\|$, as long as one is able to perform projections onto balls of the dual norm $\|.\|_*$. We have further shown that when $\|.\|$ is the ℓ_2- or the ℓ_∞-norm, a single pass provides the exact solution when the groups $\mathcal{G}$ are correctly ordered. We show however in Appendix C, that, perhaps surprisingly, the conclusions of Proposition 5 do not hold for general ℓ_q-norms, if $q \notin \{1,2,\infty\}$. Next, we give another interpretation of this result.

3.4 Interpretation in Terms of Composition of Proximal Operators

In Algorithm 1, since all the vectors ξ^g are initialized to $\mathbf{0}$, when the group g is considered, we have by induction $\mathbf{u} - \sum_{h \neq g} \xi^h = \mathbf{u} - \sum_{h \preceq g} \xi^h$. Thus, to maintain at each iteration of the inner loop $\mathbf{v} = \mathbf{u} - \sum_{h \neq g} \xi^h$ one can instead update $\mathbf{v}$ after updating ξ^g according to $\mathbf{v} \leftarrow \mathbf{v} - \xi^g$. Moreover, since ξ^g is no longer needed in the algorithm, and since only the entries of $\mathbf{v}$ indexed by g are updated, we can combine the two updates into $\mathbf{v}_{|g} \leftarrow \mathbf{v}_{|g} - \Pi_{\|.\|_* \leq \lambda\omega_g}(\mathbf{v}_{|g})$, leading to a simplified Algorithm 2 equivalent to Algorithm 1.

Algorithm 2 Practical Computation of the Proximal Operator for ℓ_2- or ℓ_∞-norms.

```
Inputs: $\mathbf{u} \in \mathbb{R}^p$ and an ordered tree-structured set of groups $\mathcal{G}$.
Outputs: $\mathbf{v}$ (primal solution).
Initialization: $\mathbf{v} = \mathbf{u}$.
for $g \in \mathcal{G}$, following the order $\preceq$, do
    $\mathbf{v}_{|g} \leftarrow \mathbf{v}_{|g} - \Pi_{\|.\|_* \leq \lambda\omega_g}(\mathbf{v}_{|g})$.
end for
```

Actually, in light of the classical relationship between proximal operator and projection (as discussed in Section 3.1), it is easy to show that each update $\mathbf{v}_{|g} \leftarrow \mathbf{v}_{|g} - \Pi_{\|.\|_* \leq \lambda\omega_g}(\mathbf{v}_{|g})$ is equivalent to $\mathbf{v}_{|g} \leftarrow \text{Prox}_{\lambda\omega_g\|.\|}[\mathbf{v}_{|g}]$. To simplify the notations, we define the proximal operator for a group g in $\mathcal{G}$ as $\text{Prox}^g(\mathbf{u}) \triangleq \text{Prox}_{\lambda\omega_g\|.\|}(\mathbf{u}_{|g})$ for every vector $\mathbf{u}$ in $\mathbb{R}^p$.

Thus, Algorithm 2 in fact performs a sequence of $|\mathcal{G}|$ proximal operators, and we have shown the following corollary of Proposition 5:

Corollary 6 (Composition of Proximal Operators)
Let $g_1 \preccurlyeq \ldots \preccurlyeq g_m$ such that $\mathcal{G} = \{g_1, \ldots, g_m\}$. The proximal operator $Prox_{\lambda\Omega}$ associated with the norm Ω can be written as the composition of elementary operators:

$$Prox_{\lambda\Omega} = Prox^{g_m} \circ \ldots \circ Prox^{g_1}.$$

3.5 Efficient Implementation and Complexity

Since Algorithm 2 involves $|\mathcal{G}|$ projections on the dual balls (respectively the ℓ_2- and the ℓ_1-balls for the ℓ_2- and ℓ_∞-norms) of vectors in $\mathbb{R}^p$, in a first approximation, its complexity is at most $O(p^2)$, because each of these projections can be computed in $O(p)$ operations (Brucker, 1984; Maculan and Galdino de Paula, 1989). But in fact, the algorithm performs one projection for each group g involving $|g|$ variables, and the total complexity is therefore $O\Big(\sum_{g \in \mathcal{G}} |g|\Big)$. By noticing that if g and h are two groups with the same depth in the tree, then $g \cap h = \emptyset$, it is easy to show that the number of variables involved in all the projections is less than or equal to dp, where d is the depth of the tree:

Algorithm 3 Fast computation of the Proximal operator for ℓ_2-norm case.

```
Require: $\mathbf{u} \in \mathbb{R}^p$ (input vector), set of groups $\mathcal{G}$, $(\omega_g)_{g\in\mathcal{G}}$ (positive weights), and $g_0$ (root of the tree).
1: Variables: $\rho = (\rho_g)_{g\in\mathcal{G}}$ in $\mathbb{R}^{|\mathcal{G}|}$ (scaling factors); $\mathbf{v}$ in $\mathbb{R}^p$ (output, primal variable).
2: computeSqNorm($g_0$).
3: recursiveScaling($g_0$,1).
4: Return $\mathbf{v}$ (primal solution).

Procedure computeSqNorm($g$)
1: Compute the squared norm of the group: $\eta_g \leftarrow \|\mathbf{u}_{\text{root}(g)}\|_2^2 + \sum_{h\in\text{children}(g)}$ computeSqNorm($h$).
2: Compute the scaling factor of the group: $\rho_g \leftarrow \big(1 - \lambda\omega_g/\sqrt{\eta_g}\big)_+$.
3: Return $\eta_g\rho_g^2$.

Procedure recursiveScaling($g$,$t$)
1: $\rho_g \leftarrow t\rho_g$.
2: $\mathbf{v}_{\text{root}(g)} \leftarrow \rho_g \mathbf{u}_{\text{root}(g)}$.
3: for $h \in$ children($g$) do
4:   recursiveScaling($h$,$\rho_g$).
5: end for
```

Lemma 7 (Complexity of Algorithm 2)
Algorithm 2 gives the solution of the primal problem Equation (7) in $O(pd)$ operations, where d is the depth of the tree.

Lemma 7 should not suggest that the complexity is linear in p, since d could depend of p as well, and in the worst case the hierarchy is a chain, yielding $d = p-1$. However, in a balanced tree, $d = O(\log(p))$. In practice, the structures we have considered experimentally are relatively flat, with a depth not exceeding $d = 5$, and the complexity is therefore almost linear.

Moreover, in the case of the ℓ_2-norm, it is actually possible to propose an algorithm with complexity $O(p)$. Indeed, in that case each of the proximal operators Prox^g is a scaling operation: $\mathbf{v}_{|g} \leftarrow \big(1 - \lambda\omega_g/\|\mathbf{v}_{|g}\|_2\big)_+ \mathbf{v}_{|g}$. The composition of these operators in Algorithm 1 thus corresponds to performing sequences of scaling operations. The idea behind Algorithm 3 is that the corresponding scaling factors depend only on the norms of the successive residuals of the projections and that these norms can be computed recursively in one pass through all nodes in $O(p)$ operations; finally, computing and applying all scalings to each entry takes then again $O(p)$ operations.

To formulate the algorithm, two new notations are used: for a group g in $\mathcal{G}$, we denote by root(g) the indices of the variables that are at the root of the subtree corresponding to g,[10] and by children(g) the set of groups that are the children of root(g) in the tree. For example, in the tree presented in Figure 2, $\text{root}(\{3,5,6\}) = \{3\}$, $\text{root}(\{1,2,3,4,5,6\}) = \{1\}$, $\text{children}(\{3,5,6\}) = \{\{5\},\{6\}\}$, and $\text{children}(\{1,2,3,4,5,6\}) = \{\{2,4\},\{3,5,6\}\}$. Note that all the groups of children(g) are necessarily included in g. The next lemma is proved in Appendix B.

Lemma 8 (Correctness and complexity of Algorithm 3)
When $\|.\|$ is chosen to be the ℓ_2-norm, Algorithm 3 gives the solution of the primal problem Equation (7) in $O(p)$ operations.

10. As a reminder, root(g) is not a singleton when several dictionary elements are considered per node.

So far the dictionary $\mathbf{D}$ was fixed to be for example a wavelet basis. In the next section, we apply the tools we developed for solving efficiently problem (5) to learn a dictionary $\mathbf{D}$ adapted to our hierarchical sparse coding formulation.

4. Application to Dictionary Learning

We start by briefly describing dictionary learning.

4.1 The Dictionary Learning Framework

Let us consider a set $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ in $\mathbb{R}^{m\times n}$ of n signals of dimension m. Dictionary learning is a matrix factorization problem which aims at representing these signals as linear combinations of the dictionary elements, that are the columns of a matrix $\mathbf{D} = [\mathbf{d}^1, \ldots, \mathbf{d}^p]$ in $\mathbb{R}^{m\times p}$. More precisely, the dictionary $\mathbf{D}$ is *learned* along with a matrix of decomposition coefficients $\mathbf{A} = [\alpha^1, \ldots, \alpha^n]$ in $\mathbb{R}^{p\times n}$, so that $\mathbf{x}^i \approx \mathbf{D}\alpha^i$ for every signal $\mathbf{x}^i$.

While learning simultaneously $\mathbf{D}$ and $\mathbf{A}$, one may want to encode specific prior knowledge about the problem at hand, such as, for example, the positivity of the decomposition (Lee and Seung, 1999), or the sparsity of $\mathbf{A}$ (Olshausen and Field, 1997; Aharon et al., 2006; Lee et al., 2007; Mairal et al., 2010a). This leads to penalizing or constraining $(\mathbf{D}, \mathbf{A})$ and results in the following formulation:

$$\min_{\mathbf{D}\in\mathcal{D}, \mathbf{A}\in\mathcal{A}} \frac{1}{n}\sum_{i=1}^{n}\Big[\frac{1}{2}\|\mathbf{x}^i - \mathbf{D}\alpha^i\|_2^2 + \lambda\Psi(\alpha^i)\Big], \tag{10}$$

where $\mathcal{A}$ and $\mathcal{D}$ denote two convex sets and Ψ is a regularization term, usually a norm or a squared norm, whose effect is controlled by the regularization parameter $\lambda > 0$. Note that $\mathcal{D}$ is assumed to be bounded to avoid any degenerate solutions of Problem (10). For instance, the standard sparse coding formulation takes Ψ to be the ℓ_1-norm, $\mathcal{D}$ to be the set of matrices in $\mathbb{R}^{m\times p}$ whose columns have unit ℓ_2-norm, with $\mathcal{A} = \mathbb{R}^{p\times n}$ (Olshausen and Field, 1997; Lee et al., 2007; Mairal et al., 2010a).

However, this classical setting treats each dictionary element independently from the others, and does not exploit possible relationships between them. To embed the dictionary in a tree structure, we therefore replace the ℓ_1-norm by our hierarchical norm and set $\Psi = \Omega$ in Equation (10).

A question of interest is whether hierarchical priors are more appropriate in supervised settings or in the matrix-factorization context in which we use it. It is not so common in the supervised setting to have strong prior information that allows us to organize the features in a hierarchy. On the contrary, in the case of dictionary learning, since the atoms are learned, one can argue that the dictionary elements learned will *have to* match well the hierarchical prior that is imposed by the regularization. In other words, combining structured regularization with dictionary learning has precisely the advantage that the dictionary elements will *self-organize* to match the prior.

4.2 Learning the Dictionary

Optimization for dictionary learning has already been intensively studied. We choose in this paper a typical alternating scheme, which optimizes in turn $\mathbf{D}$ and $\mathbf{A} = [\alpha^1, \ldots, \alpha^n]$ while keeping the other variable fixed (Aharon et al., 2006; Lee et al., 2007; Mairal et al., 2010a).[11] Of course, the convex optimization tools we develop in this paper do not change the intrinsic non-convex nature of the

11. Note that although we use this classical scheme for simplicity, it would also be possible to use the stochastic approach proposed by Mairal et al. (2010a).

dictionary learning problem. However, they solve the underlying convex subproblems efficiently, which is crucial to yield good results in practice. In the next section, we report good performance on some applied problems, and we show empirically that our algorithm is stable and does not seem to get trapped in bad local minima. The main difficulty of our problem lies in the optimization of the vectors α^i, i in $\{1,\dots,n\}$, for the dictionary $\mathbf{D}$ kept fixed. Because of Ω, the corresponding convex subproblem is nonsmooth and has to be solved for each of the n signals considered. The optimization of the dictionary $\mathbf{D}$ (for $\mathbf{A}$ fixed), which we discuss first, is in general easier.

4.2.1 UPDATING THE DICTIONARY D

We follow the matrix-inversion free procedure of Mairal et al. (2010a) to update the dictionary. This method consists in iterating block-coordinate descent over the columns of $\mathbf{D}$. Specifically, we assume that the domain set $\mathcal{D}$ has the form

$$\mathcal{D}_\mu \triangleq \{\mathbf{D} \in \mathbb{R}^{m\times p},\ \mu\|\mathbf{d}^j\|_1 + (1-\mu)\|\mathbf{d}^j\|_2^2 \leq 1, \text{ for all } j \in \{1,\dots,p\}\}, \tag{11}$$

or $\mathcal{D}_\mu^+ \triangleq \mathcal{D}_\mu \cap \mathbb{R}_+^{m\times p}$, with $\mu \in [0,1]$. The choice for these particular domain sets is motivated by the experiments of Section 5. For natural image patches, the dictionary elements are usually constrained to be in the unit ℓ_2-norm ball (i.e., $\mathcal{D} = \mathcal{D}_0$), while for topic modeling, the dictionary elements are distributions of words and therefore belong to the simplex (i.e., $\mathcal{D} = \mathcal{D}_1^+$). The update of each dictionary element amounts to performing a Euclidean projection, which can be computed efficiently (Mairal et al., 2010a). Concerning the stopping criterion, we follow the strategy from the same authors and go over the columns of $\mathbf{D}$ only a few times, typically 5 times in our experiments. Although we have not explored locality constraints on the dictionary elements, these have been shown to be particularly relevant to some applications such as patch-based image classification (Yu et al., 2009). Combining tree structure and locality constraints is an interesting future research.

4.2.2 UPDATING THE VECTORS α^i

The procedure for updating the columns of $\mathbf{A}$ is based on the results derived in Section 3.3. Furthermore, positivity constraints can be added on the domain of $\mathbf{A}$, by noticing that for our norm Ω and any vector $\mathbf{u}$ in $\mathbb{R}^p$, adding these constraints when computing the proximal operator is equivalent to solving $\min_{\mathbf{v}\in\mathbb{R}^p} \frac{1}{2}\|[\mathbf{u}]_+ - \mathbf{v}\|_2^2 + \lambda\Omega(\mathbf{v})$. This equivalence is proved in Appendix B.6. We will indeed use positive decompositions to model text corpora in Section 5. Note that by constraining the decompositions α^i to be nonnegative, some entries α^i_j may be set to zero in addition to those already zeroed out by the norm Ω. As a result, the sparsity patterns obtained in this way might not satisfy the tree-structured condition (1) anymore.

5. Experiments

We next turn to the experimental validation of our hierarchical sparse coding.

5.1 Implementation Details

In Section 3.3, we have shown that the proximal operator associated to Ω can be computed exactly and efficiently. The problem is therefore amenable to fast proximal algorithms that are well suited to nonsmooth convex optimization. Specifically, we tried the accelerated scheme from both Nesterov

(2007) and Beck and Teboulle (2009), and finally opted for the latter since, for a comparable level of precision, fewer calls of the proximal operator are required. The basic proximal scheme presented in Section 3.1 is formalized by Beck and Teboulle (2009) as an algorithm called ISTA; the same authors propose moreover an accelerated variant, FISTA, which is a similar procedure, except that the operator is not directly applied on the current estimate, but on an auxiliary sequence of points that are linear combinations of past estimates. This latter algorithm has an optimal convergence rate in the class of first-order techniques, and also allows for warm restarts, which is crucial in the alternating scheme of dictionary learning.[12]

Finally, we monitor the convergence of the algorithm by checking the relative decrease in the cost function.[13] Unless otherwise specified, all the algorithms used in the following experiments are implemented in `C/C++`, with a `Matlab` interface. Our implementation is freely available at `http://www.di.ens.fr/willow/SPAMS/`.

5.2 Speed Benchmark

To begin with, we conduct speed comparisons between our approach and other convex programming methods, in the setting where Ω is chosen to be a linear combination of ℓ_2-norms. The algorithms that take part in the following benchmark are:

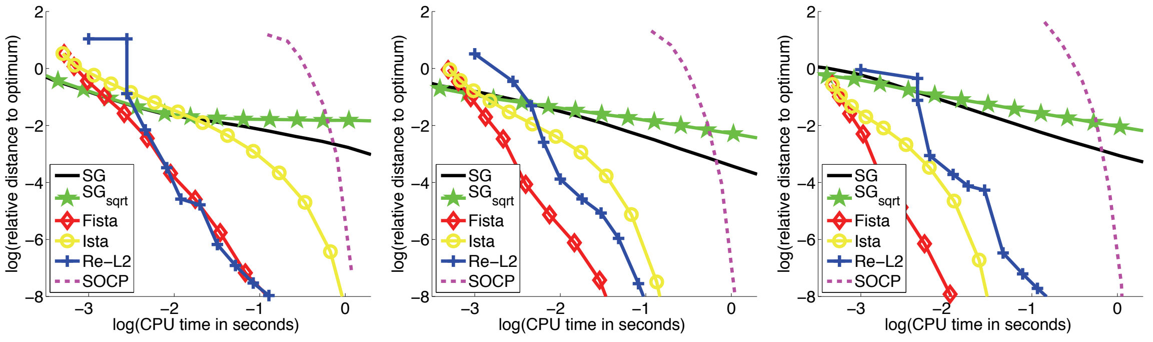
- Proximal methods, with ISTA and the accelerated FISTA methods (Beck and Teboulle, 2009).
- A reweighted-least-square scheme (Re-ℓ_2), as described by Jenatton et al. (2009); Kim and Xing (2010). This approach is adapted to the square loss, since closed-form updates can be used.[14]
- Subgradient descent, whose step size is taken to be equal either to $a/(k+b)$ or $a/(\sqrt{k}+b)$ (respectively referred to as SG and SG_{sqrt}), where k is the iteration number, and (a,b) are the best[15] parameters selected on the logarithmic grid $(a,b) \in \{10^{-4},\dots,10^{3}\} \times \{10^{-2},\dots,10^{5}\}$.
- A commercial software (`Mosek`, available at `http://www.mosek.com/`) for second-order cone programming (SOCP).

Moreover, the experiments we carry out cover various settings, with notably different sparsity regimes, that is, low, medium and high, respectively corresponding to about $50\%, 10\%$ and 1% of the total number of dictionary elements. Eventually, all reported results are obtained on a single core of a 3.07Ghz CPU with 8GB of memory.

5.2.1 HIERARCHICAL DICTIONARY OF NATURAL IMAGE PATCHES

In this first benchmark, we consider a least-squares regression problem regularized by Ω that arises in the context of denoising of natural image patches, as further exposed in Section 5.4. In particular, based on a hierarchical dictionary, we seek to reconstruct noisy 16×16-patches. The dictionary we use is represented on Figure 7. Although the problem involves a small number of variables, that is, $p = 151$ dictionary elements, it has to be solved repeatedly for tens of thousands of patches, at moderate precision. It is therefore crucial to be able to solve this problem quickly and efficiently.

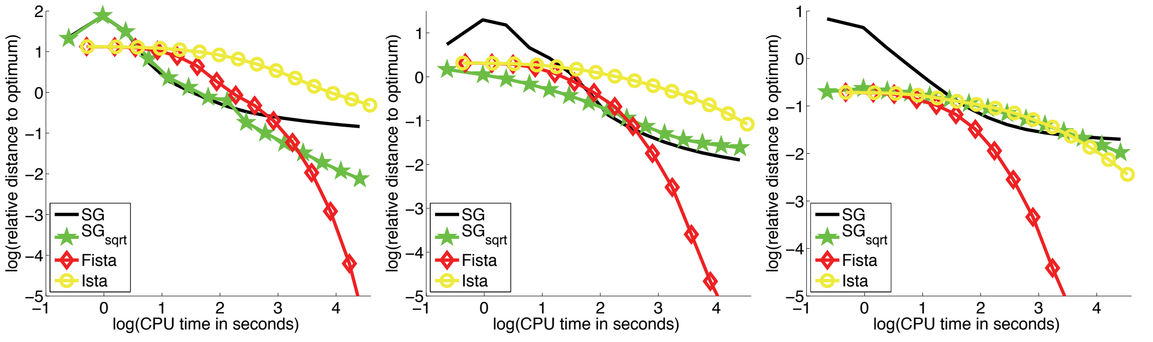
12. Unless otherwise specified, the initial stepsize in ISTA/FISTA is chosen as the maximum eigenvalue of the sampling covariance matrix divided by 100, while the growth factor in the line search is set to 1.5.
13. We are currently investigating algorithms for computing duality gaps based on network flow optimization tools (Mairal et al., 2010b).
14. The computation of the updates related to the variational formulation (6) also benefits from the hierarchical structure of $\mathcal{G}$, and can be performed in $O(p)$ operations.
15. "The best step size" is understood as being the step size leading to the smallest cost function after 500 iterations.



(a) scale: small, regul.: low (b) scale: small, regul.: medium (c) scale: small, regul.: high

Figure 3: Benchmark for solving a least-squares regression problem regularized by the hierarchical norm Ω. The experiment is small scale, $m = 256, p = 151$, and shows the performances of six optimization methods (see main text for details) for three levels of regularization. The curves represent the relative value of the objective to the optimal value as a function of the computational time in second on a $\log_{10} / \log_{10}$ scale. All reported results are obtained by averaging 5 runs.

We can draw several conclusions from the results of the simulations reported in Figure 3. First, we observe that in most cases, the accelerated proximal scheme performs better than the other approaches. In addition, unlike FISTA, ISTA seems to suffer in non-sparse scenarios. In the least sparse setting, the reweighted-ℓ_2 scheme is the only method that competes with FISTA. It is however not able to yield truly sparse solutions, and would therefore need a subsequent (somewhat arbitrary) thresholding operation. As expected, the generic techniques such as SG and SOCP do not compete with dedicated algorithms.



(a) scale: large, regul.: low (b) scale: large, regul.: medium (c) scale: large, regul.: high

Figure 4: Benchmark for solving a large-scale multi-class classification problem for four optimization methods (see details about the data sets and the methods in the main text). Three levels of regularization are considered. The curves represent the relative value of the objective to the optimal value as a function of the computational time in second on a $\log_{10} / \log_{10}$ scale. In the highly regularized setting, tuning the step-size for the subgradient turned out to be difficult, which explains the behavior of SG in the first iterations.

5.2.2 MULTI-CLASS CLASSIFICATION OF CANCER DIAGNOSIS

The second benchmark explores a different supervised learning setting, where f is no longer the square loss function. The goal is to demonstrate that our optimization tools apply in various scenarios, beyond traditional sparse approximation problems. To this end, we consider a gene expression data set[16] in the context of cancer diagnosis. More precisely, we focus on a multi-class classification problem where the number m of samples to be classified is small compared to the number p of gene expressions that characterize these samples. Each atom thus corresponds to a gene expression across the m samples, whose class labels are recorded in the vector $\mathbf{x}$ in $\mathbb{R}^m$.

The data set contains $m = 308$ samples, $p = 30017$ variables and 26 classes. In addition, the data exhibit highly-correlated dictionary elements. Inspired by Kim and Xing (2010), we build the tree-structured set of groups $\mathcal{G}$ using Ward's hierarchical clustering (Johnson, 1967) on the gene expressions. The norm Ω built in this way aims at capturing the hierarchical structure of gene expression networks (Kim and Xing, 2010).

Instead of the square loss function, we consider the multinomial logistic loss function that is better suited to deal with multi-class classification problems (see, e.g., Hastie et al., 2009). As a direct consequence, algorithms whose applicability crucially depends on the choice of the loss function f are removed from the benchmark. This is the case with reweighted-ℓ_2 schemes that do not have closed-form updates anymore. Importantly, the choice of the multinomial logistic loss function leads to an optimization problem over a matrix with dimensions p times the number of classes (i.e., a total of $30017 \times 26 \approx 780000$ variables). Also, due to scalability issues, generic interior point solvers could not be considered here.

The results in Figure 4 highlight that the accelerated proximal scheme performs overall better that the two other methods. Again, it is important to note that both proximal algorithms yield sparse solutions, which is not the case for SG.

5.3 Denoising with Tree-Structured Wavelets

We demonstrate in this section how a tree-structured sparse regularization can improve classical wavelet representation, and how our method can be used to efficiently solve the corresponding large-scale optimization problems. We consider two wavelet orthonormal bases, Haar and Daubechies3 (see Mallat, 1999), and choose a classical quad-tree structure on the coefficients, which has notably proven to be useful for image compression problems (Baraniuk, 1999). This experiment follows the approach of Zhao et al. (2009) who used the same tree-structured regularization in the case of small one-dimensional signals, and the approach of Baraniuk et al. (2010) and Huang et al. (2009) images where images were reconstructed from compressed sensing measurements with a hierarchical nonconvex penalty.

We compare the performance for image denoising of both nonconvex and convex approaches. Specifically, we consider the following formulation

$$\min_{\alpha\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\alpha\|_2^2 + \lambda\psi(\alpha) = \min_{\alpha\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{D}^\top\mathbf{x}-\alpha\|_2^2 + \lambda\psi(\alpha),$$

where $\mathbf{D}$ is one of the orthonormal wavelet basis mentioned above, $\mathbf{x}$ is the input noisy image, $\mathbf{D}\alpha$ is the estimate of the denoised image, and ψ is a sparsity-inducing regularization. Note that in this case, $m = p$. We first consider classical settings where ψ is either the ℓ_1-norm— this leads to the

16. The data set we use is *14_Tumors*, which is freely available at `http://www.gems-system.org/`.

wavelet soft-thresholding method of Donoho and Johnstone (1995)—or the ℓ_0-pseudo-norm, whose solution can be obtained by hard-thresholding (see Mallat, 1999). Then, we consider the convex tree-structured regularization Ω defined as a sum of ℓ_2-norms (ℓ_∞-norms), which we denote by Ω_{ℓ_2} (respectively Ω_{ℓ_∞}). Since the basis is here orthonormal, solving the corresponding decomposition problems amounts to computing a single instance of the proximal operator. As a result, when ψ is Ω_{ℓ_2}, we use Algorithm 3 and for Ω_{ℓ_∞}, Algorithm 2 is applied. Finally, we consider the nonconvex tree-structured regularization used by Baraniuk et al. (2010) denoted here by ℓ_0^{tree}, which we have presented in Equation (4); the implementation details for ℓ_0^{tree} can be found in Appendix A.

		Haar				
	σ	ℓ_0 [0.0012]	ℓ_0^{tree} [0.0098]	ℓ_1 [0.0016]	Ω_{ℓ_2} [0.0125]	Ω_{ℓ_∞} [0.0221]
PSNR	5	34.48	34.78	35.52	**35.89**	35.79
	10	29.63	30.24	30.74	**31.40**	31.23
	25	24.44	25.27	25.30	**26.41**	26.14
	50	21.53	22.37	20.42	**23.41**	23.05
	100	19.27	20.09	19.43	**20.97**	20.58
IPSNR	5	-	$.30 \pm .23$	$1.04 \pm .31$	$\mathbf{1.41 \pm .45}$	$1.31 \pm .41$
	10	-	$.60 \pm .24$	$1.10 \pm .22$	$\mathbf{1.76 \pm .26}$	$1.59 \pm .22$
	25	-	$.83 \pm .13$	$.86 \pm .35$	$\mathbf{1.96 \pm .22}$	$1.69 \pm .21$
	50	-	$.84 \pm .18$	$.46 \pm .28$	$\mathbf{1.87 \pm .20}$	$1.51 \pm .20$
	100	-	$.82 \pm .14$	$.15 \pm .23$	$\mathbf{1.69 \pm .19}$	$1.30 \pm .19$

		Daub3				
	σ	ℓ_0 [0.0013]	ℓ_0^{tree} [0.0099]	ℓ_1 [0.0017]	Ω_{ℓ_2} [0.0129]	Ω_{ℓ_∞} [0.0204]
PSNR	5	34.64	34.95	35.74	**36.14**	36.00
	10	30.03	30.63	31.10	**31.79**	31.56
	25	25.04	25.84	25.76	**26.90**	26.54
	50	22.09	22.90	22.42	**23.90**	23.41
	100	19.56	20.45	19.67	**21.40**	20.87
IPSNR	5	-	$.31 \pm .21$	$1.10 \pm .23$	$\mathbf{1.49 \pm .34}$	$1.36 \pm .31$
	10	-	$.60 \pm .16$	$1.06 \pm .25$	$\mathbf{1.76 \pm .19}$	$1.53 \pm .17$
	25	-	$.80 \pm .10$	$.71 \pm .28$	$\mathbf{1.85 \pm .17}$	$1.50 \pm .18$
	50	-	$.81 \pm .15$	$.33 \pm .24$	$\mathbf{1.80 \pm .11}$	$1.33 \pm .12$
	100	-	$.89 \pm .13$	$0.11 \pm .24$	$\mathbf{1.82 \pm .24}$	$1.30 \pm .17$

Table 1: Top part of the tables: Average PSNR measured for the denoising of 12 standard images, when the wavelets are Haar or Daubechies3 wavelets (see Mallat, 1999), for two nonconvex approaches (ℓ_0 and ℓ_0^{tree}) and three different convex regularizations—that is, the ℓ_1-norm, the tree-structured sum of ℓ_2-norms (Ω_{ℓ_2}), and the tree-structured sum of ℓ_∞-norms (Ω_{ℓ_∞}). Best results for each level of noise and each wavelet type are in bold. Bottom part of the tables: Average improvement in PSNR with respect to the ℓ_0 nonconvex method (the standard deviations are computed over the 12 images). CPU times (in second) averaged over all images and noise realizations are reported in brackets next to the names of the methods they correspond to.

Compared to Zhao et al. (2009), the novelty of our approach is essentially to be able to solve efficiently and exactly large-scale instances of this problem. We use 12 classical standard test images,[17] and generate noisy versions of them corrupted by a white Gaussian noise of variance σ. For each image, we test several values of $\lambda = 2^{\frac{i}{4}}\sigma\sqrt{\log m}$, with i taken in a specific range.[18] We then keep the parameter λ giving the best reconstruction error. The factor $\sigma\sqrt{\log m}$ is a classical heuristic for choosing a reasonable regularization parameter (see Mallat, 1999). We provide reconstruction results in terms of PSNR in Table 1.[19] We report in this table the results when Ω is chosen to be a sum of ℓ_2-norms or ℓ_∞-norms with weights ω_g all equal to one. Each experiment was run 5 times with different noise realizations. In every setting, we observe that the tree-structured norm significantly outperforms the ℓ_1-norm and the nonconvex approaches. We also present a visual comparison on two images on Figure 5, showing that the tree-structured norm reduces visual artefacts (these artefacts are better seen by zooming on a computer screen). The wavelet transforms in our experiments are computed with the matlabPyrTools software.[20]

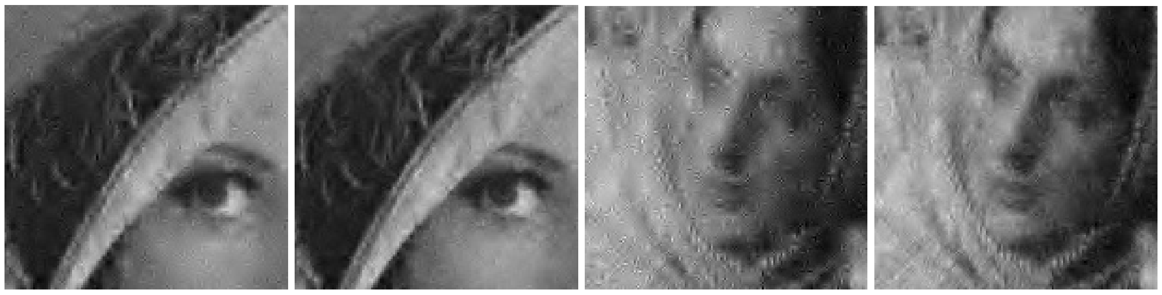

(a) Lena, $\sigma = 25, \ell_1$ (b) Lena, $\sigma = 25, \Omega_{\ell_2}$ (c) Barb., $\sigma = 50, \ell_1$ (d) Barb., $\sigma = 50, \Omega_{\ell_2}$

Figure 5: Visual comparison between the wavelet shrinkage model with the ℓ_1-norm and the tree-structured model, on cropped versions of the images Lena and Barb.. Haar wavelets are used.

This experiment does of course not provide state-of-the-art results for image denoising (see Mairal et al., 2009b, and references therein), but shows that the tree-structured regularization significantly improves the reconstruction quality for wavelets. In this experiment the convex setting Ω_{ℓ_2} and Ω_{ℓ_∞} also outperforms the nonconvex one ℓ_0^{tree}.[21] We also note that the speed of our approach makes it scalable to real-time applications. Solving the proximal problem for an image with $m = 512 \times 512 = 262\,144$ pixels takes approximately 0.013 seconds on a single core of a 3.07GHz CPU if Ω is a sum of ℓ_2-norms, and 0.02 seconds when it is a sum of ℓ_∞-norms. By contrast, unstructured approaches have a speed-up factor of about 7-8 with respect to the tree-structured methods.

17. These images are used in classical image denoising benchmarks. See Mairal et al. (2009b).
18. For the convex formulations, i ranges in $\{-15,-14,\dots,15\}$, while in the nonconvex case i ranges in $\{-24,\dots,48\}$.
19. Denoting by MSE the mean-squared-error for images whose intensities are between 0 and 255, the PSNR is defined as $\text{PSNR} = 10\log_{10}(255^2/\text{MSE})$ and is measured in dB. A gain of 1dB reduces the MSE by approximately 20%.
20. Software available at `http://www.cns.nyu.edu/~eero/steerpyr/`.
21. It is worth mentioning that comparing convex and nonconvex approaches for sparse regularization is a bit difficult. This conclusion holds for the classical formulation we have used, but might not hold in other settings such as Coifman and Donoho (1995).

5.4 Dictionaries of Natural Image Patches

This experiment studies whether a hierarchical structure can help dictionaries for denoising natural image patches, and in which noise regime the potential gain is significant. We aim at reconstructing *corrupted* patches from a test set, after having learned dictionaries on a training set of *non-corrupted* patches. Though not typical in machine learning, this setting is reasonable in the context of images, where lots of non-corrupted patches are easily available.[22]

noise	50 %	60 %	70 %	80 %	90 %
flat	19.3 ± 0.1	26.8 ± 0.1	36.7 ± 0.1	50.6 ± 0.0	72.1 ± 0.0
tree	18.6 ± 0.1	25.7 ± 0.1	35.0 ± 0.1	48.0 ± 0.0	65.9 ± 0.3

Table 2: Quantitative results of the reconstruction task on natural image patches. First row: percentage of missing pixels. Second and third rows: mean square error multiplied by 100, respectively for classical sparse coding, and tree-structured sparse coding.

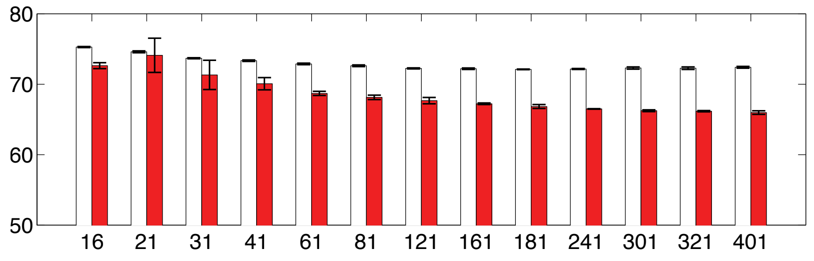


Figure 6: Mean square error multiplied by 100 obtained with 13 structures with error bars, sorted by number of dictionary elements from 16 to 401. Red plain bars represents the tree-structured dictionaries. White bars correspond to the flat dictionary model containing the same number of dictionary as the tree-structured one. For readability purpose, the y-axis of the graph starts at 50.

We extracted 100 000 patches of size $m = 8 \times 8$ pixels from the Berkeley segmentation database of natural images (Martin et al., 2001), which contains a high variability of scenes. We then split this data set into a training set $\mathbf{X}_{tr}$, a validation set $\mathbf{X}_{val}$, and a test set $\mathbf{X}_{te}$, respectively of size 50 000, 25 000, and 25 000 patches. All the patches are centered and normalized to have unit ℓ_2-norm.

For the first experiment, the dictionary $\mathbf{D}$ is learned on $\mathbf{X}_{tr}$ using the formulation of Equation (10), with $\mu = 0$ for $\mathcal{D}_\mu$ as defined in Equation (11). The validation and test sets are corrupted by removing a certain percentage of pixels, the task being to reconstruct the missing pixels from the known pixels. We thus introduce for each element $\mathbf{x}$ of the validation/test set, a vector $\tilde{\mathbf{x}}$, equal to $\mathbf{x}$ for the known pixel values and 0 otherwise. Similarly, we define $\tilde{\mathbf{D}}$ as the matrix equal to $\mathbf{D}$, except for the rows corresponding to missing pixel values, which are set to 0. By decomposing $\tilde{\mathbf{x}}$ on $\tilde{\mathbf{D}}$, we obtain a sparse code α, and the estimate of the reconstructed patch is defined as $\mathbf{D}\alpha$. Note that this procedure assumes that we know which pixel is missing and which is not for every element $\mathbf{x}$.

The parameters of the experiment are the regularization parameter λ_{tr} used during the training step, the regularization parameter λ_{te} used during the validation/test step, and the structure of the

22. Note that we study the ability of the model to reconstruct independent patches, and additional work is required to apply our framework to a full image processing task, where patches usually overlap (Elad and Aharon, 2006; Mairal et al., 2009b).

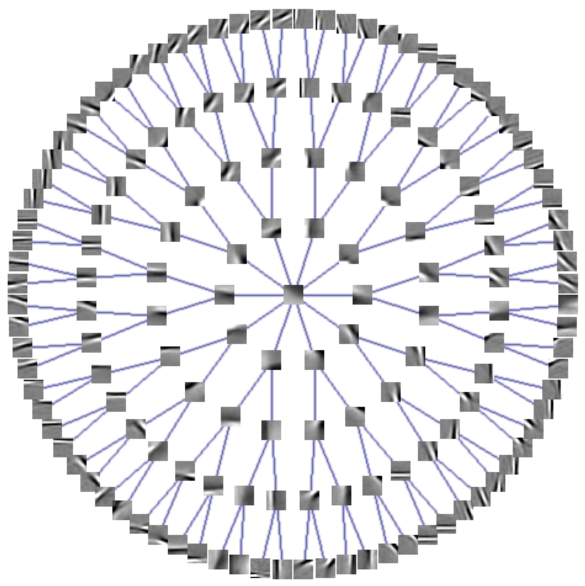


Figure 7: Learned dictionary with a tree structure of depth 5. The root of the tree is in the middle of the figure. The branching factors are $p_1 = 10$, $p_2 = 2$, $p_3 = 2$, $p_4 = 2$. The dictionary is learned on $50,000$ patches of size 16×16 pixels.

tree. For every reported result, these parameters were selected by taking the ones offering the best performance on the *validation* set, before reporting any result from the *test* set. The values for the regularization parameters $\lambda_{tr}, \lambda_{te}$ were selected on a logarithmic scale $\{2^{-10}, 2^{-9}, \ldots, 2^2\}$, and then further refined on a finer logarithmic scale with multiplicative increments of $2^{-1/4}$. For simplicity, we chose arbitrarily to use the ℓ_∞-norm in the structured norm Ω, with all the weights equal to one. We tested 21 balanced tree structures of depth 3 and 4, with different *branching factors* $p_1, p_2, \ldots, p_{d-1}$, where d is the depth of the tree and p_k, $k \in \{1, \ldots, d-1\}$ is the number of children for the nodes at depth k. The branching factors tested for the trees of depth 3 where $p_1 \in \{5, 10, 20, 40, 60, 80, 100\}$, $p_2 \in \{2, 3\}$, and for trees of depth 4, $p_1 \in \{5, 10, 20, 40\}$, $p_2 \in \{2, 3\}$ and $p_3 = 2$, giving 21 possible structures associated with dictionaries with at most 401 elements. For each tree structure, we evaluated the performance obtained with the tree-structured dictionary along with a non-structured dictionary containing the same number of elements. These experiments were carried out four times, each time with a different initialization, and with a different noise realization.

Quantitative results are reported in Table 2. For all fractions of missing pixels considered, the tree-structured dictionary outperforms the "unstructured one", and the most significant improvement is obtained in the noisiest setting. Note that having more dictionary elements is worthwhile when using the tree structure. To study the influence of the chosen structure, we report in Figure 6 the results obtained with the 13 tested structures of depth 3, along with those obtained with unstructured dictionaries containing the same number of elements, when 90% of the pixels are missing. For each dictionary size, the tree-structured dictionary significantly outperforms the unstructured one. An example of a learned tree-structured dictionary is presented on Figure 7. Dictionary elements naturally organize in groups of patches, often with low frequencies near the root of the tree, and high frequencies near the leaves.

5.5 Text Documents

This last experimental section shows that our approach can also be applied to model text corpora. The goal of probabilistic topic models is to find a low-dimensional representation of a collection of documents, where the representation should provide a semantic description of the collection. Approaching the problem in a parametric Bayesian framework, latent Dirichlet allocation (LDA) Blei et al. (2003) model documents, represented as vectors of word counts, as a mixture of a predefined number of *latent topics* that are distributions over a fixed vocabulary. LDA is fundamentally a matrix factorization problem: Buntine (2002) shows that LDA can be interpreted as a Dirichlet-multinomial counterpart of factor analysis. The number of topics is usually small compared to the size of the vocabulary (e.g., 100 against 10000), so that the topic proportions of each document provide a compact representation of the corpus. For instance, these new features can be used to feed a classifier in a subsequent classification task. We similarly use our dictionary learning approach to find low-dimensional representations of text corpora.

Suppose that the signals $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n]$ in $\mathbb{R}^{m \times n}$ are each the *bag-of-word* representation of each of n documents over a vocabulary of m words, the k-th component of $\mathbf{x}^i$ standing for the frequency of the k-th word in the document i. If we further assume that the entries of $\mathbf{D}$ and $\mathbf{A}$ are nonnegative, and that the dictionary elements $\mathbf{d}^j$ have unit ℓ_1-norm, the decomposition $(\mathbf{D}, \mathbf{A})$ can be interpreted as the parameters of a topic-mixture model. The regularization Ω induces the organization of these topics on a tree, so that, if a document involves a certain topic, then all ancestral topics in the tree are also present in the topic decomposition. Since the hierarchy is shared by all documents, the topics at the top of the tree participate in every decomposition, and should therefore gather the lexicon which is common to all documents. Conversely, the deeper the topics in the tree, the more specific they should be. An extension of LDA to model topic hierarchies was proposed by Blei et al. (2010), who introduced a non-parametric Bayesian prior over trees of topics and modelled documents as convex combinations of topics selected along a path in the hierarchy. We plan to compare our approach with this model in future work.

5.5.1 VISUALIZATION OF NIPS PROCEEDINGS

We qualitatively illustrate our approach on the NIPS proceedings from 1988 through 1999 (Griffiths and Steyvers, 2004). After removing words appearing fewer than 10 times, the data set is composed of 1714 articles, with a vocabulary of 8274 words. As explained above, we consider $\mathcal{D}_1^+$ and take $\mathcal{A}$ to be $\mathbb{R}_+^{p \times n}$. Figure 8 displays an example of a learned dictionary with 13 topics, obtained by using the ℓ_∞-norm in Ω and selecting manually $\lambda = 2^{-15}$. As expected and similarly to Blei et al. (2010), we capture the stopwords at the root of the tree, and topics reflecting the different subdomains of the conference such as neurosciences, optimization or learning theory.

5.5.2 POSTING CLASSIFICATION

We now consider a binary classification task of n postings from the 20 Newsgroups data set.[23] We learn to discriminate between the postings from the two newsgroups *alt.atheism* and *talk.religion.misc*, following the setting of Lacoste-Julien et al. (2008) and Zhu et al. (2009). After removing words appearing fewer than 10 times and standard stopwords, these postings form a data set of 1425 documents over a vocabulary of 13312 words. We compare different dimensionality reduction tech-

23. Available at `http://people.csail.mit.edu/jrennie/20Newsgroups/`.

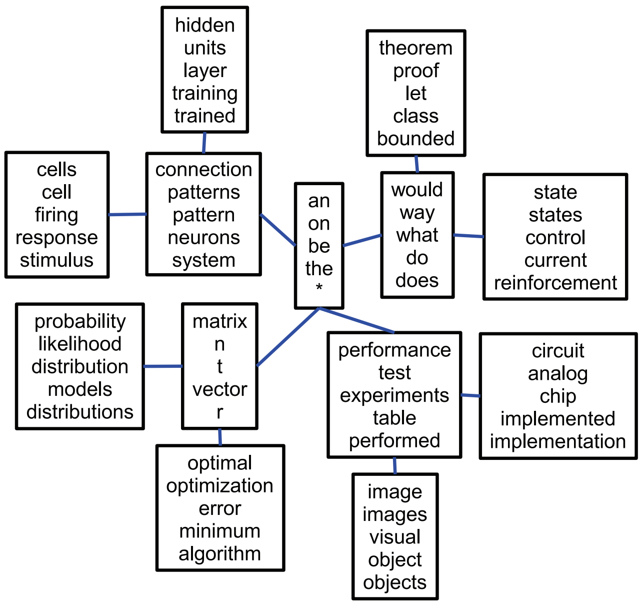

an on be the *
connection patterns pattern neurons system
hidden units layer training trained
cells cell firing response stimulus
would way what do does
theorem proof let class bounded
state states control current reinforcement
matrix n t vector r
probability likelihood distribution models distributions
optimal optimization error minimum algorithm
performance test experiments table performed
circuit analog chip implemented implementation
image images visual object objects

Figure 8: Example of a topic hierarchy estimated from 1714 NIPS proceedings papers (from 1988 through 1999). Each node corresponds to a topic whose 5 most important words are displayed. Single characters such as n, t, r are part of the vocabulary and often appear in NIPS papers, and their place in the hierarchy is semantically relevant to children topics.

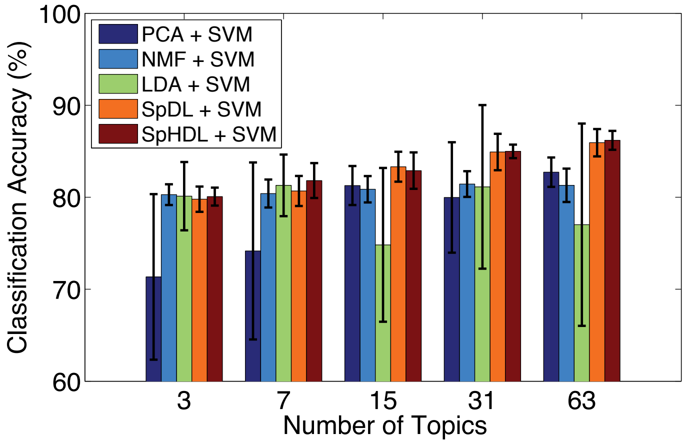


Figure 9: Binary classification of two newsgroups: classification accuracy for different dimensionality reduction techniques coupled with a linear SVM classifier. The bars and the errors are respectively the mean and the standard deviation, based on 10 random splits of the data set. Best seen in color.

niques that we use to feed a linear SVM classifier, that is, we consider (i) LDA, with the code from Blei et al. (2003), (ii) principal component analysis (PCA), (iii) nonnegative matrix factorization (NMF), (iv) standard sparse dictionary learning (denoted by SpDL) and (v) our sparse hierarchical approach (denoted by SpHDL). Both SpDL and SpHDL are optimized over $\mathcal{D}_1^+$ and $\mathcal{A} = \mathbb{R}_+^{p \times n}$, with the weights ω_g equal to 1. We proceed as follows: given a random split into a training/test set of $1\,000/425$ postings, and given a number of topics p (also the number of components for PCA, NMF, SpDL and SpHDL), we train an SVM classifier based on the low-dimensional representation of the postings. This is performed on a training set of $1\,000$ postings, where the parameters, $\lambda \in \{2^{-26}, \ldots, 2^{-5}\}$ and/or $C_{\text{svm}} \in \{4^{-3}, \ldots, 4^{1}\}$ are selected by 5-fold cross-validation. We report in Figure 9 the average classification scores on the test set of 425 postings, based on 10 random splits, for different number of topics. Unlike the experiment on image patches, we consider only complete binary trees with depths in $\{1, \ldots, 5\}$. The results from Figure 9 show that SpDL and SpHDL perform better than the other dimensionality reduction techniques on this task. As a baseline, the SVM classifier applied directly to the raw data (the 13312 words) obtains a score of 90.9 ± 1.1, which is better than all the tested methods, but without dimensionality reduction (as already reported by Blei et al., 2003). Moreover, the error bars indicate that, though nonconvex, SpDL and SpHDL do not seem to suffer much from instability issues. Even if SpDL and SpHDL perform similarly, SpHDL has the advantage to provide a more interpretable topic mixture in terms of hierarchy, which standard unstructured sparse coding does not.

6. Discussion

We have applied hierarchical sparse coding in various settings, with fixed/learned dictionaries, and based on different types of data, namely, natural images and text documents. A line of research to pursue is to develop other optimization tools for structured norms with general overlapping groups. For instance, Mairal et al. (2010b) have used network flow optimization techniques for that purpose, and Bach (2010) submodular function optimization. This framework can also be used in the context of hierarchical kernel learning (Bach, 2008), where we believe that our method can be more efficient than existing ones.

This work establishes a connection between dictionary learning and probabilistic topic models, which should prove fruitful as the two lines of work have focused on different aspects of the same unsupervised learning problem: Our approach is based on convex optimization tools, and provides experimentally more stable data representations. Moreover, it can be easily extended with the same tools to other types of structures corresponding to other norms (Jenatton et al., 2009; Jacob et al., 2009). It should be noted, however, that, unlike some Bayesian methods, dictionary learning by itself does not provide mechanisms for the automatic selection of model hyper-parameters (such as the dictionary size or the topology of the tree). An interesting common line of research to pursue could be the supervised design of dictionaries, which has been proved useful in the two frameworks (Mairal et al., 2009a; Bradley and Bagnell, 2009; Blei and McAuliffe, 2008).

Acknowledgments

This paper was partially supported by grants from the Agence Nationale de la Recherche (MGA Project) and from the European Research Council (SIERRA Project 239993). The authors would like to thank Jean Ponce for interesting discussions and suggestions for improving this manuscript. They also would like to thank Volkan Cevher for pointing out links between our approach and non-convex tree-structured regularization and for insightful discussions. Finally, we thank the reviewers for their constructive and helpful comments.

Appendix A. Links with Tree-Structured Nonconvex Regularization

We present in this section an algorithm introduced by Donoho (1997) in the more general context of approximation from dyadic partitions (see Section 6 in Donoho, 1997). This algorithm solves the following problem

$$\min_{\mathbf{v}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \lambda \sum_{g\in\mathcal{G}} \delta^g(\mathbf{v}), \tag{12}$$

where the $\mathbf{u}$ in $\mathbb{R}^p$ is given, λ is a regularization parameter, $\mathcal{G}$ is a set of tree-structured groups in the sense of definition 1, and the functions δ^g are defined as in Equation (4)—that is, $\delta^g(\mathbf{v}) = 1$ if there exists j in g such that $\mathbf{v}_j \neq 0$, and 0 otherwise. This problem can be viewed as a proximal operator for the nonconvex regularization $\sum_{g\in\mathcal{G}} \delta^g(\mathbf{v})$. As we will show, it can be solved efficiently, and in fact it can be used to obtain approximate solutions of the nonconvex problem presented in Equation (1), or to solve tree-structured wavelet decompositions as done by Baraniuk et al. (2010).

We now briefly show how to derive the dynamic programming approach introduced by Donoho (1997). Given a group g in $\mathcal{G}$, we use the same notations root(g) and children(g) introduced in Section 3.5. It is relatively easy to show that finding a solution of Equation (12) amounts to finding the support $S \subseteq \{1,\dots,p\}$ of its solution and that the problem can be equivalently rewritten

$$\min_{S\subseteq\{1,\dots,p\}} -\frac{1}{2}\|\mathbf{u}_S\|_2^2 + \lambda \sum_{g\in\mathcal{G}} \delta^g(S), \tag{13}$$

with the abusive notation $\delta^g(S) = 1$ if $g \cap S \neq \emptyset$ and 0 otherwise. We now introduce the quantity

$$\psi_g(S) \triangleq \begin{cases} 0 & \text{if } g\cap S = \emptyset \\ -\frac{1}{2}\|\mathbf{u}_{\text{root}(g)}\|_2^2 + \lambda + \sum_{h\in\text{children}(g)} \psi_h(S) & \text{otherwise.} \end{cases}$$

After a few computations, solving Equation (13) can be shown to be equivalent to minimizing $\psi_{g_0}(S)$ where g_0 is the root of the tree. It is then easy to prove that for any group g in $\mathcal{G}$, we have

$$\min_{S\subseteq\{1,\dots,p\}} \psi_g(S) = \min\Big(0, -\frac{1}{2}\|\mathbf{u}_{\text{root}(g)}\|_2^2 + \lambda + \sum_{h\in\text{children}(g)} \min_{S'\subseteq\{1,\dots,p\}} \psi_h(S')\Big),$$

which leads to the following dynamic programming approach presented in Algorithm 4. This algorithm shares several conceptual links with Algorithm 2 and 3. It traverses the tree in the same order, has a complexity in $O(p)$, and it can be shown that the whole procedure actually performs a sequence of thresholding operations on the variable $\mathbf{v}$.

Appendix B. Proofs

We gather here the proofs of the technical results of the paper.

B.1 Proof of Lemma 3

Proof The proof relies on tools from conic duality (Boyd and Vandenberghe, 2004). Let us introduce the cone $\mathcal{C} \triangleq \{(\mathbf{v},z)\in\mathbb{R}^{p+1};\ \|\mathbf{v}\|\leq z\}$ and its dual counterpart $\mathcal{C}^* \triangleq \{(\xi,\tau)\in\mathbb{R}^{p+1};\ \|\xi\|_*\leq\tau\}$. These cones induce generalized inequalities for which Lagrangian duality also applies. We refer the interested readers to Boyd and Vandenberghe (2004) for further details.

Algorithm 4 Computation of the Proximal Operator for the Nonconvex Approach

```
Inputs: $\mathbf{u} \in \mathbb{R}^p$, a tree-structured set of groups $\mathcal{G}$ and $g_0$ (root of the tree).
Outputs: $\mathbf{v}$ (primal solution).
Initialization: $\mathbf{v} \leftarrow \mathbf{u}$.
Call recursiveThresholding($g_0$).
Procedure recursiveThresholding($g$)
1: $\eta \leftarrow \min\Big(0, -\frac{1}{2}\|\mathbf{u}_{\text{root}(g)}\|_2^2 + \lambda + \sum_{h\in\text{children}(g)}$ recursiveThresholding($h$)$\Big)$.
2: if $\eta = 0$ then
3:     $\mathbf{v}_g \leftarrow 0$.
4: end if
5: Return $\eta$.
```

We can rewrite problem (7) as

$$\min_{\mathbf{v}\in\mathbb{R}^p, \mathbf{z}\in\mathbb{R}^{|\mathcal{G}|}} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \lambda\sum_{g\in\mathcal{G}} \omega_g z_g, \text{ such that } (\mathbf{v}_{|g}, z_g) \in \mathcal{C}, \ \forall g \in \mathcal{G},$$

by introducing the primal variables $\mathbf{z} = (z_g)_{g\in\mathcal{G}} \in \mathbb{R}^{|\mathcal{G}|}$, with the additional $|\mathcal{G}|$ conic constraints $(\mathbf{v}_{|g}, z_g) \in \mathcal{C}$, for $g \in \mathcal{G}$.

This primal problem is convex and satisfies Slater's conditions for generalized conic inequalities (i.e., existence of a feasible point in the interior of the domain), which implies that strong duality holds (Boyd and Vandenberghe, 2004). We now consider the Lagrangian $\mathcal{L}$ defined as

$$\mathcal{L}(\mathbf{v}, \mathbf{z}, \tau, \xi) = \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \lambda\sum_{g\in\mathcal{G}} \omega_g z_g - \sum_{g\in\mathcal{G}} \begin{pmatrix} z_g \\ \mathbf{v}_{|g} \end{pmatrix}^\top \begin{pmatrix} \tau_g \\ \xi^g \end{pmatrix},$$

with the dual variables $\tau = (\tau_g)_{g\in\mathcal{G}}$ in $\mathbb{R}^{|\mathcal{G}|}$, and $\xi = (\xi^g)_{g\in\mathcal{G}}$ in $\mathbb{R}^{p\times|\mathcal{G}|}$, such that for all $g \in \mathcal{G}$, $\xi^g_j = 0$ if $j \notin g$ and $(\xi^g, \tau_g) \in \mathcal{C}^*$.

The dual function is obtained by minimizing out the primal variables. To this end, we take the derivatives of $\mathcal{L}$ with respect to the primal variables $\mathbf{v}$ and $\mathbf{z}$ and set them to zero, which leads to

$$\mathbf{v} - \mathbf{u} - \sum_{g\in\mathcal{G}} \xi^g = 0 \quad \text{and} \quad \forall g \in \mathcal{G}, \ \lambda\omega_g - \tau_g = 0.$$

After simplifying the Lagrangian and flipping (without loss of generality) the sign of ξ, we obtain the dual problem in Equation (8). We derive the optimality conditions from the Karush–Kuhn–Tucker conditions for generalized conic inequalities (Boyd and Vandenberghe, 2004). We have that $\{\mathbf{v}, \mathbf{z}, \tau, \xi\}$ are optimal if and only if

$$\begin{array}{rll} & \forall g \in \mathcal{G}, z_g\tau_g - \mathbf{v}_{|g}^\top\xi^g & = 0, \quad \text{(Complementary slackness)} \\ \forall g \in \mathcal{G}, (\mathbf{v}_{|g}, z_g) \in \mathcal{C}, & \forall g \in \mathcal{G}, \lambda\omega_g - \tau_g & = 0, \\ \forall g \in \mathcal{G}, (\xi^g, \tau_g) \in \mathcal{C}^*, & \mathbf{v} - \mathbf{u} + \sum_{g\in\mathcal{G}} \xi^g & = 0. \end{array}$$

Combining the complementary slackness with the definition of the dual norm, we have

$$\forall g \in \mathcal{G}, \ z_g\tau_g = \mathbf{v}_{|g}^\top\xi^g \leq \|\mathbf{v}_{|g}\|\|\xi^g\|_*.$$

Furthermore, using the fact that $\forall g \in \mathcal{G},\ (\mathbf{v}_{|g}, z_g) \in \mathcal{C}$ and $(\xi^g, \tau_g) = (\xi^g, \lambda\omega_g) \in \mathcal{C}^*$, we obtain the following chain of inequalities

$$\forall g \in \mathcal{G},\ \lambda z_g \omega_g = \mathbf{v}_{|g}^\top \xi^g \leq \|\mathbf{v}_{|g}\| \|\xi^g\|_* \leq z_g \|\xi^g\|_* \leq \lambda z_g \omega_g,$$

for which equality must hold. In particular, we have $\mathbf{v}_{|g}^\top \xi^g = \|\mathbf{v}_{|g}\| \|\xi^g\|_*$ and $z_g \|\xi^g\|_* = \lambda z_g \omega_g$. If $\mathbf{v}_{|g} \neq 0$, then z_g cannot be equal to zero, which implies in turn that $\|\xi^g\|_* = \lambda\omega_g$. Eventually, applying Lemma 9 gives the advertised optimality conditions.

Conversely, starting from the optimality conditions of Lemma 3, and making use again of Lemma 9, we can derive the Karush–Kuhn–Tucker conditions displayed above. More precisely, we define for all $g \in \mathcal{G}$,

$$\tau_g \triangleq \lambda\omega_g \quad \text{and} \quad z_g \triangleq \|\mathbf{v}_{|g}\|.$$

The only condition that needs to be discussed is the complementary slackness condition. If $\mathbf{v}_{|g} = 0$, then it is easily satisfied. Otherwise, combining the definitions of τ_g, z_g and the fact that

$$\mathbf{v}_{|g}^\top \xi^g = \|\mathbf{v}_{|g}\| \|\xi^g\|_* \text{ and } \|\xi^g\|_* = \lambda\omega_g,$$

we end up with the desired complementary slackness. ■

B.2 Optimality Condition for the Projection on the Dual Ball

Lemma 9 (Projection on the dual ball)
Let $\mathbf{w} \in \mathbb{R}^p$ *and* $t > 0$. *We have* $\kappa = \Pi_{\|.\|_* \leq t}(\mathbf{w})$ *if and only if*

$$\begin{cases} \textit{if } \|\mathbf{w}\|_* \leq t, & \kappa = \mathbf{w}, \\ \textit{otherwise}, & \|\kappa\|_* = t \ \textit{ and } \ \kappa^\top(\mathbf{w} - \kappa) = \|\kappa\|_* \|\mathbf{w} - \kappa\|. \end{cases}$$

Proof When the vector $\mathbf{w}$ is already in the ball of $\|.\|_*$ with radius t, that is, $\|\mathbf{w}\|_* \leq t$, the situation is simple, since the projection $\Pi_{\|.\|_* \leq t}(\mathbf{w})$ obviously gives $\mathbf{w}$ itself. On the other hand, a necessary and sufficient optimality condition for having $\kappa = \Pi_{\|.\|_* \leq t}(\mathbf{w}) = \arg\min_{\|\mathbf{y}\|_* \leq t} \|\mathbf{w} - \mathbf{y}\|_2$ is that the residual $\mathbf{w} - \kappa$ lies in the normal cone of the constraint set (Borwein and Lewis, 2006), that is, for all $\mathbf{y}$ such that $\|\mathbf{y}\|_* \leq t$, $(\mathbf{w} - \kappa)^\top(\mathbf{y} - \kappa) \leq 0$. The displayed result then follows from the definition of the dual norm, namely $\|\kappa\|_* = \max_{\|\mathbf{z}\| \leq 1} \mathbf{z}^\top \kappa$. ■

B.3 Proof of Lemma 4

Proof First, notice that the conclusion $\xi^h = \Pi_{\|.\|_* \leq \lambda\omega_h}(\mathbf{v}_{|h} + \xi^h)$ simply comes from the definition of ξ^h and $\mathbf{v}$, along with the fact that $\xi^g = \xi^g_{|h}$ since $g \subseteq h$. We now examine ξ^g.

The proof mostly relies on the optimality conditions characterizing the projection onto a ball of the dual norm $\|\cdot\|_*$. Precisely, by Lemma 9, we need to show that either

$$\xi^g = \mathbf{u}_{|g} - \xi^h_{|g}, \text{ if } \|\mathbf{u}_{|g} - \xi^h_{|g}\|_* \leq t_g,$$

or

$$\|\xi^g\|_* = t_g \ \text{ and } \ \xi^{g\top}(\mathbf{u}_{|g} - \xi^h_{|g} - \xi^g) = \|\xi^g\|_* \|\mathbf{u}_{|g} - \xi^h_{|g} - \xi^g\|.$$

Note that the feasibility of ξ^g, that is, $\|\xi^g\|_* \leq t_g$, holds by definition of κ^g.

Let us first assume that $\|\xi^g\|_* < t_g$. We necessarily have that $\mathbf{u}_{|g}$ also lies in the interior of the ball of $\|.\|_*$ with radius t_g, and it holds that $\xi^g = \mathbf{u}_{|g}$. Since $g \subseteq h$, we have that the vector $\mathbf{u}_{|h} - \xi^g = \mathbf{u}_{|h} - \mathbf{u}_{|g}$ has only zero entries on g. As a result, $\xi^h_g = 0$ (or equivalently, $\xi^h_{|g} = 0$) and we obtain

$$\xi^g = \mathbf{u}_{|g} = \mathbf{u}_{|g} - \xi^h_{|g},$$

which is the desired conclusion. From now on, we assume that $\|\xi^g\|_* = t_g$. It then remains to show that

$$\xi^{g\top}(\mathbf{u}_{|g} - \xi^h_{|g} - \xi^g) = \|\xi^g\|_* \|\mathbf{u}_{|g} - \xi^h_{|g} - \xi^g\|.$$

We now distinguish two cases, according to the norm used.

ℓ_2-norm: As a consequence of Lemma 9, the optimality condition reduces to the conditions for equality in the Cauchy-Schwartz inequality, that is, when the vectors have same signs and are linearly dependent. Applying these conditions to individual projections we get that there exists $\rho_g, \rho_h > 0$ such that

$$\rho_g \xi^g = \mathbf{u}_{|g} - \xi^g \quad \text{and} \quad \rho_h \xi^h = \mathbf{u}_{|h} - \xi^g - \xi^h. \tag{14}$$

Note that the case $\rho_h = 0$ leads to $\mathbf{u}_{|h} - \xi^g - \xi^h = 0$, and therefore $\mathbf{u}_{|g} - \xi^g - \xi^h_{|g} = 0$ since $g \subseteq h$, which directly yields the result. The case $\rho_g = 0$ implies $\mathbf{u}_{|g} - \xi^g = 0$ and therefore $\xi^h_{|g} = 0$, yielding the result as well. Now, we can therefore assume $\rho_h > 0$ and $\rho_g > 0$. From the first equality of (14), we have $\xi^g = \xi^g_{|g}$ since $(\rho_g + 1)\xi^g = \mathbf{u}_{|g}$. Further using the fact that $g \subseteq h$ in the second equality of (14), we obtain

$$(\rho_h + 1)\xi^h_{|g} = \mathbf{u}_{|g} - \xi^g = \rho_g \xi^g.$$

This implies that $\mathbf{u}_{|g} - \xi^g - \xi^h_{|g} = \rho_g \xi^g - \frac{\rho_g}{\rho_h + 1}\xi^g$, which eventually leads to

$$\xi^g = \frac{\rho_h + 1}{\rho_g \rho_h}(\mathbf{u}_{|g} - \xi^g - \xi^h_{|g}).$$

The desired conclusion follows $\xi^{g\top}(\mathbf{u}_{|g} - \xi^g - \xi^h_{|g}) = \|\xi^g\|_2 \|\mathbf{u}_{|g} - \xi^g - \xi^h_{|g}\|_2$.

ℓ_∞-norm: In this case, the optimality corresponds to the conditions for equality in the ℓ_∞-ℓ_1 Hölder inequality. Specifically, $\xi^g = \Pi_{\|.\|_* \leq t_g}(\mathbf{u}_{|g})$ holds if and only if for all $\xi^g_j \neq 0, j \in g$, we have

$$\mathbf{u}_j - \xi^g_j = \|\mathbf{u}_{|g} - \xi^g\|_\infty \operatorname{sign}(\xi^g_j).$$

Looking at the same condition for ξ^h, we have that $\xi^h = \Pi_{\|.\|_* \leq t_h}(\mathbf{u}_{|h} - \xi_g)$ holds if and only if for all $\xi^h_j \neq 0, j \in h$, we have

$$\mathbf{u}_j - \xi^g_j - \xi^h_j = \|\mathbf{u}_{|h} - \xi^g - \xi^h\|_\infty \operatorname{sign}(\xi^h_j).$$

From those relationships we notably deduce that for all $j \in g$ such that $\xi^g_j \neq 0$, $\operatorname{sign}(\xi^g_j) = \operatorname{sign}(\mathbf{u}_j) = \operatorname{sign}(\xi^h_j) = \operatorname{sign}(\mathbf{u}_j - \xi^g_j) = \operatorname{sign}(\mathbf{u}_j - \xi^g_j - \xi^h_j)$. Let $j \in g$ such that $\xi^g_j \neq 0$. At this point, using the equalities we have just presented,

$$|\mathbf{u}_j - \xi^g_j - \xi^h_j| = \begin{cases} \|\mathbf{u}_{|g} - \xi^g\|_\infty & \text{if } \xi^h_j = 0 \\ \|\mathbf{u}_{|h} - \xi^g - \xi^h\|_\infty & \text{if } \xi^h_j \neq 0. \end{cases}$$

Since $\|\mathbf{u}_{|g} - \xi^g\|_\infty \geq \|\mathbf{u}_{|g} - \xi^g - \xi^h_{|g}\|_\infty$ (which can be shown using the sign equalities above), and $\|\mathbf{u}_{|h} - \xi^g - \xi^h\|_\infty \geq \|\mathbf{u}_{|g} - \xi^g - \xi^h_{|g}\|_\infty$ (since $g \subseteq h$), we have

$$\|\mathbf{u}_{|g} - \xi^g - \xi^h_{|g}\|_\infty \geq |\mathbf{u}_j - \xi^g_j - \xi^h_j| \geq \|\mathbf{u}_{|g} - \xi^g - \xi^h_{|g}\|_\infty,$$

and therefore for all $\xi^g_j \neq 0$, $j \in g$, we have $\mathbf{u}_j - \xi^g_j - \xi^h_j = \|\mathbf{u}_{|g} - \xi^g - \xi^h_{|g}\|_\infty \operatorname{sign}(\xi^g_j)$, which yields the result. ■

B.4 Proof of Lemma 8

Proof Notice first that the procedure `computeSqNorm` is called exactly once for each group g in $\mathcal{G}$, computing a set of scalars $(\rho_g)_{g \in \mathcal{G}}$ in an order which is compatible with the convergence in one pass of Algorithm 1—that is, the children of a node are processed prior to the node itself. Following such an order, the update of the group g in the original Algorithm 1 computes the variable ξ^g which updates implicitly the primal variable as follows

$$\mathbf{v}_{|g} \leftarrow \big(1 - \frac{\lambda \omega_g}{\|\mathbf{v}_{|g}\|_2}\big)_+ \mathbf{v}_{|g}.$$

It is now possible to show by induction that for all group g in $\mathcal{G}$, after a call to the procedure `computeSqNorm`(g), the auxiliary variable η_g takes the value $\|\mathbf{v}_{|g}\|_2^2$ where $\mathbf{v}$ has the same value as during the iteration g of Algorithm 1. Therefore, after calling the procedure `computeSqNorm`(g_0), where g_0 is the root of the tree, the values ρ_g correspond to the successive scaling factors of the variable $\mathbf{v}_{|g}$ obtained during the execution of Algorithm 1. After having computed all the scaling factors ρ_g, $g \in \mathcal{G}$, the procedure `recursiveScaling` ensures that each variable j in $\{1, \ldots, p\}$ is scaled by the product of all the ρ_h, where h is an ancestor of the variable j.

The complexity of the algorithm is easy to characterize: Each procedure `computeSqNorm` and `recursiveScaling` is called p times, each call for a group g has a constant number of operations plus as many operations as the number of children of p. Since each child can be called at most one time, the total number of operation of the algorithm is $O(p)$. ■

B.5 Sign Conservation by Projection

The next lemma specifies a property for projections when $\|.\|$ is further assumed to be a ℓ_q-norm (with $q \geq 1$). We recall that in that case, $\|.\|_*$ is simply the $\ell_{q'}$-norm, with $q' = (1 - 1/q)^{-1}$.

Lemma 10 (Projection on the dual ball and sign property)
Let $\mathbf{w} \in \mathbb{R}^p$ and $t > 0$. Let us assume that $\|.\|$ is a ℓ_q-norm (with $q \geq 1$). Consider also a diagonal matrix $\mathbf{S} \in \mathbb{R}^{p \times p}$ whose diagonal entries are in $\{-1, 1\}$. We have $\Pi_{\|.\|_* \leq t}(\mathbf{w}) = \mathbf{S}\Pi_{\|.\|_* \leq t}(\mathbf{S}\mathbf{w})$.

Proof Let us consider $\kappa = \Pi_{\|.\|_* \leq t}(\mathbf{w})$. Using essentially the same argument as in the proof of Lemma 9, we have for all $\mathbf{y}$ such that $\|\mathbf{y}\|_{q'} \leq t$, $(\mathbf{w} - \kappa)^\top(\mathbf{y} - \kappa) \leq 0$. Noticing that $\mathbf{S}^\top\mathbf{S} = \mathbf{I}$ and $\|\mathbf{y}\|_{q'} = \|\mathbf{S}\mathbf{y}\|_{q'}$, we further obtain $(\mathbf{S}\mathbf{w} - \mathbf{S}\kappa)^\top(\mathbf{y}' - \mathbf{S}\kappa) \leq 0$ for all $\mathbf{y}'$ with $\|\mathbf{y}'\|_{q'} \leq t$. This implies in turn that $\mathbf{S}\Pi_{\|.\|_* \leq t}(\mathbf{w}) = \Pi_{\|.\|_* \leq t}(\mathbf{S}\mathbf{w})$, which is equivalent to the advertised conclusion. ■

Based on this lemma, note that we can assume without loss of generality that the vector we want to project (in this case, $\mathbf{w}$) has only nonnegative entries. Indeed, it is sufficient to store beforehand the signs of that vector, compute the projection of the vector with nonnegative entries, and assign the stored signs to the result of the projection.

B.6 Non-negativity Constraint for the Proximal Operator

The next lemma shows how we can easily add a non-negativity constraint on the proximal operator when the norm Ω is *absolute* (Stewart and Sun, 1990, Definition 1.2), that is, a norm for which the relation $\Omega(\mathbf{u}) \leq \Omega(\mathbf{w})$ holds for any two vectors $\mathbf{w}$ and $\mathbf{u} \in \mathbb{R}^p$ such that $|\mathbf{u}_j| \leq |\mathbf{w}_j|$ for all j.

Lemma 11 (Non-negativity constraint for the proximal operator)
Let $\kappa \in \mathbb{R}^p$ and $\lambda > 0$. Consider an absolute norm Ω. We have

$$\operatorname*{arg\,min}_{\mathbf{z}\in\mathbb{R}^p} \Big[\frac{1}{2}\|[\kappa]_+ - \mathbf{z}\|_2^2 + \lambda\Omega(\mathbf{z})\Big] = \operatorname*{arg\,min}_{\mathbf{z}\in\mathbb{R}^p_+} \Big[\frac{1}{2}\|\kappa - \mathbf{z}\|_2^2 + \lambda\Omega(\mathbf{z})\Big]. \tag{15}$$

Proof Let us denote by $\hat{\mathbf{z}}^+$ and $\hat{\mathbf{z}}$ the unique solutions of the left- and right-hand side of (15) respectively. Consider the normal cone $\mathcal{N}_{\mathbb{R}^p_+}(\mathbf{z}_0)$ of $\mathbb{R}^p_+$ at the point $\mathbf{z}_0$ (Borwein and Lewis, 2006) and decompose κ into its positive and negative parts, $\kappa = [\kappa]_+ + [\kappa]_-$. We can now write down the optimality conditions for the two convex problems above (Borwein and Lewis, 2006): $\hat{\mathbf{z}}^+$ is optimal if and only if there exists $\mathbf{w} \in \partial\Omega(\hat{\mathbf{z}}^+)$ such that $\hat{\mathbf{z}}^+ - [\kappa]_+ + \lambda\mathbf{w} = \mathbf{0}$. Similarly, $\hat{\mathbf{z}}$ is optimal if and only if there exists $(\mathbf{s},\mathbf{u}) \in \partial\Omega(\hat{\mathbf{z}}) \times \mathcal{N}_{\mathbb{R}^p_+}(\hat{\mathbf{z}})$ such that $\hat{\mathbf{z}} - \kappa + \lambda\mathbf{s} + \mathbf{u} = \mathbf{0}$. We now prove that $[\kappa]_- = \kappa - [\kappa]_+$ belongs to $\mathcal{N}_{\mathbb{R}^p_+}(\hat{\mathbf{z}}^+)$. We proceed by contradiction. Let us assume that there exists $\mathbf{z} \in \mathbb{R}^p_+$ such that $[\kappa]_-^\top(\mathbf{z} - \hat{\mathbf{z}}^+) > 0$. This implies that there exists $j \in \{1,\ldots,p\}$ for which $[\kappa_j]_- < 0$ and $\mathbf{z}_j - \hat{\mathbf{z}}_j^+ < 0$. In other words, we have $0 \leq \mathbf{z}_j = \mathbf{z}_j - [\kappa_j]_+ < \hat{\mathbf{z}}_j^+ = \hat{\mathbf{z}}_j^+ - [\kappa_j]_+$. With the assumption made on Ω and replacing $\hat{\mathbf{z}}_j^+$ by $\mathbf{z}_j$, we have found a solution to the left-hand side of (15) with a stricly smaller cost function than the one evaluated at $\hat{\mathbf{z}}^+$, hence the contradiction. Putting the pieces together, we now have

$$\hat{\mathbf{z}}^+ - [\kappa]_+ + \lambda\mathbf{w} = \hat{\mathbf{z}}^+ - \kappa + \lambda\mathbf{w} + [\kappa]_- = \mathbf{0}, \text{ with } (\mathbf{w}, [\kappa]_-) \in \partial\Omega(\hat{\mathbf{z}}^+) \times \mathcal{N}_{\mathbb{R}^p_+}(\hat{\mathbf{z}}^+),$$

which shows that $\hat{\mathbf{z}}^+$ is the solution of the right-hand side of (15). ∎

Appendix C. Counterexample for ℓ_q-norms, with $q \notin \{1,2,\infty\}$.

The result we have proved in Proposition 5 in the specific setting where $\|.\|$ is the ℓ_2- or ℓ_∞-norm does not hold more generally for ℓ_q-norms, when q is not in $\{1,2,\infty\}$. Let $q > 1$ satisfying this condition. We denote by $q' \triangleq (1-q^{-1})^{-1}$ the norm parameter dual to q. We keep the same notation as in Lemma 4 and assume from now on that $\|\mathbf{u}_{|g}\|_{q'} > t_g$ and $\|\mathbf{u}_{|h}\|_{q'} > t_g + t_h$. These two inequalities guarantee that the vectors $\mathbf{u}_{|g}$ and $\mathbf{u}_{|h} - \xi^g$ do not lie in the interior of the $\ell_{q'}$-norm balls, of respective radius t_g and t_h.

We show in this section that there exists a setting for which the conclusion of Lemma 4 does not hold anymore. We first focus on a necessary condition of Lemma 4:

Lemma 12 (Necessary condition of Lemma 4)
Let $\|.\|$ be a ℓ_q-norm, with $q \notin \{1,2,\infty\}$. If the conclusion of Lemma 4 holds, then the vectors $\xi^g_{|g}$ and $\xi^h_{|g}$ are linearly dependent.

Proof According to our assumptions on $\mathbf{u}_{|g}$ and $\mathbf{u}_{|h} - \xi^g$, we have that $\|\xi^g\|_{q'} = t_g$ and $\|\xi^h\|_{q'} = t_h$. In this case, we can apply the second optimality conditions of Lemma 9, which states that equality holds in the ℓ_q-$\ell_{q'}$ Hölder inequality. As a result, there exists $\rho_g, \rho_h > 0$ such that for all j in g:

$$|\xi^g_j|^{q'} = \rho_g |\mathbf{u}_j - \xi^g_j|^q \quad \text{and} \quad |\xi^h_j|^{q'} = \rho_h |\mathbf{u}_j - \xi^g_j - \xi^h_j|^q.$$

If the conclusion of Lemma 4 holds—that is, we have $\xi^g = \Pi_{\|.\|_* \le t_g}(\mathbf{u}_{|g} - \xi^h_{|g})$, notice that it is not possible to have the following scenarios, as proved below by contradiction:

- If $\|\mathbf{u}_{|g} - \xi^h_{|g}\|_{q'} < t_g$, then we would have $\xi^g = \mathbf{u}_{|g} - \xi^h_{|g}$, which is impossible since $\|\xi^g\|_{q'} = t_g$.
- If $\|\mathbf{u}_{|g} - \xi^h_{|g}\|_{q'} = t_g$, then we would have for all j in g, $|\xi^h_j|^{q'} = \rho_h |\mathbf{u}_j - \xi^g_j - \xi^h_j|^q = 0$, which implies that $\xi^h_{|g} = 0$ and $\|\mathbf{u}_{|g}\|_{q'} = t_g$. This is impossible since we assumed $\|\mathbf{u}_{|g}\|_{q'} > t_g$.

We therefore have $\|\mathbf{u}_{|g} - \xi^h_{|g}\|_{q'} > t_g$ and using again the second optimality conditions of Lemma 9, there exists $\rho > 0$ such that for all j in g, $|\xi^g_j|^{q'} = \rho |\mathbf{u}_j - \xi^g_j - \xi^h_j|^q$. Combined with the previous relation on $\xi^h_{|g}$, we obtain for all j in g, $|\xi^g_j|^{q'} = \frac{\rho}{\rho_h} |\xi^h_j|^{q'}$. Since we can assume without loss of generality that $\mathbf{u}$ only has nonnegative entries (see Lemma 10), the vectors ξ^g and ξ^h can also be assumed to have nonnegative entries, hence the desired conclusion. ■

We need another intuitive property of the projection $\Pi_{\|.\|_* \le t}$ to derive our counterexample:

Lemma 13 (Order-preservation by projection)
Let $\|.\|$ be a ℓ_q-norm, with $q \notin \{1,\infty\}$ and $q' \triangleq 1/(1-q^{-1})$. Let us consider the vectors $\kappa, \mathbf{w} \in \mathbb{R}^p$ such that $\kappa = \Pi_{\|.\|_* \le t}(\mathbf{w}) = \arg\min_{\|\mathbf{y}\|_{q'} \le t} \|\mathbf{y} - \mathbf{w}\|_2$, with the radius t satisfying $\|\mathbf{w}\|_{q'} > t$. If we have $\mathbf{w}_i < \mathbf{w}_j$ for some (i,j) in $\{1,\dots,p\}^2$, then it also holds that $\kappa_i < \kappa_j$.

Proof Let us first notice that given the assumption on t, we have $\|\kappa\|_{q'} = t$. The Lagrangian $\mathcal{L}$ associated with the convex minimization problem underlying the definition of $\Pi_{\|.\|_* \le t}$ can be written as

$$\mathcal{L}(\mathbf{y}, \alpha) = \frac{1}{2}\|\mathbf{y} - \mathbf{w}\|_2^2 + \alpha\big[\|\mathbf{y}\|_{q'}^{q'} - t^{q'}\big], \text{ with the Lagrangian parameter } \alpha \ge 0.$$

At optimality, the stationarity condition for κ leads to

$$\forall\, j \in \{1,\dots,p\},\ \kappa_j - \mathbf{w}_j + \alpha q' |\kappa_j|^{q'-1} = 0.$$

We can assume without loss of generality that $\mathbf{w}$ only has nonnegative entries (see Lemma 10). Since the components of κ and $\mathbf{w}$ have the same signs (see Lemma 10), we therefore have $|\kappa_j| = \kappa_j \ge 0$, for all j in $\{1,\dots,p\}$. Note that α cannot be equal to zero because of $\|\kappa\|_{q'} = t < \|\mathbf{w}\|_{q'}$.

Let us consider the continuously differentiable function $\varphi_w : \kappa \mapsto \kappa - w + \alpha q' \kappa^{q'-1}$ defined on $(0,\infty)$. Since $\varphi_w(0) = -w < 0$, $\lim_{\kappa \to \infty} \varphi_w(\kappa) = \infty$ and φ_w is strictly nondecreasing, there exists a unique $\kappa^*_w > 0$ such that $\varphi_w(\kappa^*_w) = 0$. If we now take $w < v$, we have

$$\varphi_v(\kappa^*_w) = \varphi_w(\kappa^*_w) + w - v = w - v < 0 = \varphi_v(\kappa^*_v).$$

With φ_v being strictly nondecreasing, we thus obtain $\kappa_w^* < \kappa_v^*$. The desired conclusion stems from the application of the previous result to the stationarity condition of κ. ■

Based on the two previous lemmas, we are now in position to present our counterexample:

Proposition 14 (Counterexample)
Let $\|.\|$ be a ℓ_q-norm, with $q \notin \{1,2,\infty\}$ and $q' \triangleq 1/(1-q^{-1})$. Let us consider $\mathcal{G} = \{g,h\}$, with $g \subseteq h \subseteq \{1,\dots,p\}$ and $|g| > 1$. Let $\mathbf{u}$ be a vector in $\mathbb{R}^p$ that has at least two different nonzero entries in g, that is, there exists (i,j) in $g \times g$ such that $0 < |\mathbf{u}_i| < |\mathbf{u}_j|$. Let us consider the successive projections

$$\xi^g \triangleq \Pi_{\|.\|_* \le t_g}(\mathbf{u}_{|g}) \text{ and } \xi^h \triangleq \Pi_{\|.\|_* \le t_h}(\mathbf{u}_{|h} - \xi^g)$$

with $t_g, t_h > 0$ satisfying $\|\mathbf{u}_{|g}\|_{q'} > t_g$ and $\|\mathbf{u}_{|h}\|_{q'} > t_g + t_h$. Then, the conclusion of Lemma 4 does not hold.

Proof We apply the same rationale as in the proof of Lemma 13. Writing the stationarity conditions for ξ^g and ξ^h, we have for all j in g

$$\xi_j^g + \alpha q' (\xi_j^g)^{q'-1} - \mathbf{u}_j = 0, \quad \text{and} \quad \xi_j^h + \beta q' (\xi_j^h)^{q'-1} - (\mathbf{u}_j - \xi_j^g) = 0,$$

with Lagrangian parameters $\alpha, \beta > 0$. We now proceed by contradiction and assume that $\xi^g = \Pi_{\|.\|_* \le t_g}(\mathbf{u}_{|g} - \xi^h_{|g})$. According to Lemma 12, there exists $\rho > 0$ such that for all j in g, $\xi_j^h = \rho \xi_j^g$. If we combine the previous relations on ξ^g and ξ^h, we obtain for all j in g,

$$\xi_j^g = C(\xi_j^g)^{q'-1}, \text{ with } C \triangleq \frac{q'(\alpha - \beta\rho^{q'-1})}{\rho}.$$

If $C < 0$, then we have a contradiction, since the entries of ξ^g and $\mathbf{u}_{|g}$ have the same signs. Similarly, the case $C = 0$ leads a contradiction, since we would have $\mathbf{u}_{|g} = 0$ and $\|\mathbf{u}_{|g}\|_{q'} > t_g$. As a consequence, it follows that $C > 0$ and for all j in g, $\xi_j^g = \exp\left\{\frac{\log(C)}{2-q'}\right\}$, which means that all the entries of the vector ξ_g^g are identical. Using Lemma 13, since there exists $(i,j) \in g \times g$ such that $\mathbf{u}_i < \mathbf{u}_j$, we also have $\xi_i^g < \xi_j^g$, which leads to a contradiction. ■

References

M. Aharon, M. Elad, and A. M. Bruckstein. The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations. *IEEE Transactions on Signal Processing*, 54(11): 4311–4322, 2006.

R. K. Ahuja, T. L. Magnanti, and J. Orlin. *Network Flows*. Prentice Hall, 1993.

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems*, 2008.

F. Bach. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Information Processing Systems*, 2010.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. In *Optimization for Machine Learning*. MIT Press, 2011. To appear.

R. Baraniuk. Optimal tree approximation with wavelets. *Wavelet Applications in Signal and Image Processing VII*, 3813:206214, 1999.

R. G. Baraniuk, R. A. DeVore, G. Kyriazis, and X. M. Yu. Near best tree approximation. *Advances in Computational Mathematics*, 16(4):357–373, 2002.

R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56:1982–2001, 2010.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

S. Becker, J. Bobin, and E. Candes. NESTA: A Fast and Accurate First-order Method for Sparse Recovery. Technical report, Preprint arXiv:0904.3367, 2009.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

P. Bickel, Y. Ritov, and A. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Annals of Statistics*, 37(4):1705–1732, 2009.

D. Blei and J. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems*, 2008.

D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

D. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, 2010.

J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. Springer, 2006.

S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

D. M. Bradley and J. A. Bagnell. Differentiable sparse coding. In *Advances in Neural Information Processing Systems*, 2009.

P. Brucker. An O(n) algorithm for quadratic knapsack problems. *Operations Research Letters*, 3: 163–166, 1984.

W. L. Buntine. Variational Extensions to EM and Multinomial PCA. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2002.

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.

X. Chen, Q. Lin, S. Kim, J. Peña, J. G. Carbonell, and E. P. Xing. An efficient proximal-gradient method for single and multi-task regression with structured sparsity. Technical report, Preprint arXiv:1005.4717, 2010.

R. R. Coifman and D. L. Donoho. Translation-invariant de-noising. *Lectures Notes in Statistics*, pages 125–125, 1995.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Springer, 2010.

M. Crouse, R. D. Nowak, and R. G. Baraniuk. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46(4):886–902, 1998.

D. L. Donoho. CART and best-ortho-basis: a connection. *Annals of Statistics*, pages 1870–1911, 1997.

D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432), 1995.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32 (2):407–451, 2004.

M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 54(12):3736–3745, December 2006.

J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. Technical report, Preprint arXiv:1001.0736, 2010.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl 1):5228, 2004.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer, 2009.

L. He and L. Carin. Exploiting structure in wavelet-based Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 57:3488–3497, 2009.

C. Hu, J. T. Kwok, and W. Pan. Accelerated gradient methods for stochastic optimization and online learning. In *Advances in Neural Information Processing Systems*, 2009.

J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

L. Jacob, G. Obozinski, and J.-P. Vert. Group Lasso with overlaps and graph Lasso. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. Technical report, Preprint arXiv:0904.3523, 2009.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

S. Kim and E. P. Xing. Tree-guided group Lasso for multi-task regression with structured sparsity. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

S. Lacoste-Julien, F. Sha, and M. I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Advances in Neural Information Processing Systems*, 2008.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, 2007.

N. Maculan and J. R. G. Galdino de Paula. A linear-time median-finding algorithm for projecting a vector on the simplex of Rn. *Operations Research Letters*, 8(4):219–222, 1989.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *Advances in Neural Information Processing Systems*, 2009a.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009b.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1):19–60, 2010a.

J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In *Advances in Neural Information Processing Systems*, 2010b.

S. G. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2001.

C. A. Micchelli, J. M. Morales, and M. Pontil. A family of penalty functions for structured sparsity. In *Advances in Neural Information Processing Systems*, 2010.

J. J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *C. R. Acad. Sci. Paris Sér. A Math.*, 255:2897–2899, 1962.

D. Needell and J. A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, 2007.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

M. Schmidt and K. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, 1993.

P. Sprechmann, I. Ramirez, G. Sapiro, and Y. C. Eldar. Collaborative hierarchical sparse modeling. Technical report, Preprint arXiv:1003.0400, 2010.

G. W. Stewart and Ji-Guang Sun. *Matrix Perturbation Theory (Computer Science and Scientific Computing)*. Academic Press, 1990.

M. Stojnic, F. Parvaresh, and B. Hassibi. On the reconstruction of block-sparse signals with an optimal number of measurements. *IEEE Transactions on Signal Processing*, 57(8):3075–3085, 2009.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B*, pages 267–288, 1996.

J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

J. A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3), 2006.

M. J. Wainwright. Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1-constrained quadratic programming. *IEEE Transactions on Information Theory*, 55:2183–2202, 2009.

S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *Advances in Neural Information Processing Systems*, 2009.

G. X. Yuan, K. W. Chang, C. J. Hsieh, and C. J. Lin. Comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society. Series B*, 68(1):49–67, 2006.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, 37(6A):3468–3497, 2009.

J. Zhu, A. Ahmed, and E. P. Xing. MedLDA: maximum margin supervised topic models for regression and classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

D. Zoran and Y. Weiss. The "tree-dependent components" of natural scenes are edge filters. In *Advances in Neural Information Processing Systems*, 2009.