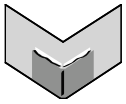
The Journal of Machine Learning Research
Volume 7
Print-Archive Edition

Pages 1385–2769



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 7
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 2006.

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 Advisory Board

JMLR Web Master

JMLR Journal of Machine Learning Research Vol. 7 2006

Journal of Machine Learning Research 7 (2006) 1385–1407 Submitted 9/05; Revised 1/06; Published 7/06

Learning Sparse Representations by Non-Negative Matrix Factorization and Sequential Cone Programming

Matthias Heiler HEILER@UNI-MANNHEIM.DE
Christoph Schnörr SCHNOERR@UNI-MANNHEIM.DE
Computer Vision, Graphics, and Pattern Recognition Group
Department of Mathematics and Computer Science
University of Mannheim
D-68131 Mannheim, Germany

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

We exploit the biconvex nature of the Euclidean non-negative matrix factorization (NMF) optimization problem to derive optimization schemes based on sequential quadratic and second order cone programming. We show that for ordinary NMF, our approach performs as well as existing state-of-the-art algorithms, while for sparsity-constrained NMF, as recently proposed by P. O. Hoyer in *JMLR 5 (2004)*, it outperforms previous methods. In addition, we show how to extend NMF learning within the same optimization framework in order to make use of class membership information in supervised learning problems.

Keywords: non-negative matrix factorization, second-order cone programming, sequential convex optimization, reverse-convex programming, sparsity

1. Introduction

Originally proposed to model physical and chemical processes (Shen and Israël, 1989; Paatero and Tapper, 1994), *non-negative matrix factorization (NMF)* has become increasingly popular for feature extraction in machine learning, computer vision, and signal processing (e.g., Hoyer and Hyvärinen, 2002; Xu et al., 2003; Smaragdis and Brown, 2003). One reason for this popularity is that NMF codes naturally favor sparse, parts-based representations (Lee and Seung, 1999; Donoho and Stodden, 2004) which in the context of recognition can be more robust than non-sparse, global features. In some application domains, researchers suggested various extensions of NMF in order to enforce very localized representations (Li et al., 2001; Hoyer, 2002; Wang et al., 2004; Chichocki et al., 2006). For example, Hoyer (2004) recently proposed NMF subject to additional constraints that allow particularly accurate control over sparseness and, indirectly, over the localization of features—see Figure 1 for an illustration.

From the viewpoint of optimization, NMF amounts to solving a difficult non-convex optimization problem. So far, learning NMF codes relied on variations of the gradient descent scheme which tend to be less efficient in the presence of additional sparsity constraints. Therefore, in this work, we exploit both the biconvex nature of the Euclidean NMF optimization criterion and the reverse-convex structure of the sparsity constraints to derive efficient optimization schemes using convex quadratic and second order cone programming (Lobo et al., 1998) as core subroutines. We show

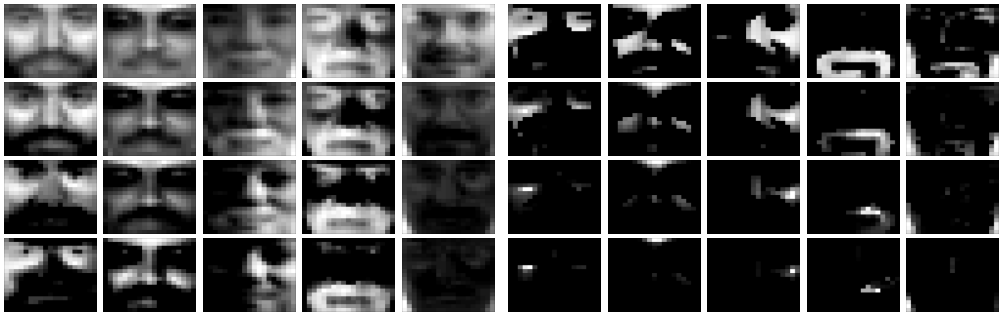


Figure 1: **Motivation of NMF with sparseness constraints.** Five basis functions (columns) with sparseness constraints ranging from 0.1 (first row, left) to 0.8 (last row, right) on W were trained on the CBCL face database. A moderate amount of sparseness encourages localized, visually meaningful base functions.

that for ordinary NMF our schemes perform as well as existing state-of-the-art algorithms, while for sparsity-constrained NMF they outperform previous methods. In addition, our approach easily extends to supervised settings similar to Fisher-NMF (Wang et al., 2004).

Organization. In Section 2, we introduce various versions of the NMF optimization problem. Then, we first consider the unconstrained case in Section 3. The representation of the sparsity constraints and the corresponding optimality conditions are described in Section 4. Using convex optimization problems as basic components, we suggest algorithms for solving the general NMF problem in Section 5. Numerical experiments validate our approach in Section 6. We conclude in Section 7.

Notation. For any $m \times n$-matrix A, we denote columns by $A = (A_{\bullet 1}, \ldots, A_{\bullet n})$ and rows by $A = (A_{1\bullet}, \ldots, A_{m\bullet})^\top$. $V \in \mathbb{R}_+^{m \times n}$ is a non-negative matrix of n data samples, and $W \in \mathbb{R}_+^{m \times r}$ a corresponding basis with loadings $H \in \mathbb{R}_+^{r \times n}$. Furthermore, we denote by e the column vector with all entries set to 1. $\|x\|_p$ represents the ℓ_p-norm for vectors x, $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$, and $\|A\|_F$ the Frobenius norm for matrices A: $\|A\|_F^2 = \sum_{i,j} A_{ij}^2 = \mathrm{tr}(A^\top A)$. $\mathrm{vec}(A) = (A_{\bullet 1}^\top, \ldots, A_{\bullet n}^\top)^\top$ is the vector obtained by concatenating the columns of the matrix A. The Kronecker product of two matrices A and B is written $A \otimes B$ (see, e.g., Graham, 1981). As usual, relations between vectors and matrices, like $x \geq 0, A \geq 0$, are understood elementwise.

2. Variations of the NMF Optimization Problem

In this section we introduce a number of optimization problems related to NMF. Corresponding optimization algorithms are developed in Sections 3 to 5.

2.1 Unconstrained NMF

The original NMF problem reads with a non-negative matrix of n data samples $V \in \mathbb{R}_+^{m\times n}$, a matrix of basis functions $W \in \mathbb{R}_+^{m\times r}$, and corresponding loadings $H \in \mathbb{R}_+^{r\times n}$:

$$\begin{aligned} \min_{W,H} \quad & \|V - WH\|_F^2 \\ \text{s.t.} \quad & 0 \leq W, H. \end{aligned} \tag{1}$$

This problem is non-convex. There are algorithms that compute the *global* optimum for such problems (Floudas and Visweswaran, 1993), however, they do not yet scale up to the large problems common in, e.g., machine learning, computer vision, or engineering. As a result, we will confine ourselves to efficiently compute a *local* optimum by solving a sequence of convex programs (Section 3.2).

2.2 Sparsity-Constrained NMF

Although NMF codes tend to be sparse (Lee and Seung, 1999), it has been suggested to control sparsity by more direct means. A particularly attractive solution was proposed by Hoyer (2004) where the following sparseness measure for vectors $x \in \mathbb{R}_+^n$, $x \neq 0$, was used:

$$\mathrm{sp}(x) := \frac{1}{\sqrt{n}-1}\left(\sqrt{n} - \frac{\|x\|_1}{\|x\|_2}\right). \tag{2}$$

Because of the relations

$$\frac{1}{\sqrt{n}}\|x\|_1 \leq \|x\|_2 \leq \|x\|_1 , \tag{3}$$

the latter being a consequence of the Cauchy-Schwarz inequality, this sparseness measure is bounded:

$$0 \leq \mathrm{sp}(x) \leq 1 . \tag{4}$$

The bounds are attained for minimal sparse vectors with equal non-zero components where $\mathrm{sp}(x) = 0$ and for maximal sparse vectors with all but one vanishing components where $\mathrm{sp}(x) = 1$. These bounds are useful from a practical viewpoint since they make it relatively intuitive to estimate sparseness for a given vector x. In addition, we found that (2) can conveniently be represented in terms of second order cones (Section 4), allowing efficient numerical solvers to be applied.

In this text, we will sometimes write $\mathrm{sp}(M) \in \mathbb{R}^n$, meaning $\mathrm{sp}(\cdot)$ is applied to each column of matrix $M \in \mathbb{R}^{m\times n}$ and the results are stacked in a column vector. Using this convention, the following constrained NMF problem was proposed in (Hoyer, 2004):

$$\begin{aligned} \min_{W,H} \quad & \|V - WH\|_F^2 \\ \text{s.t.} \quad & 0 \leq W, H \\ & \mathrm{sp}(W) = s_w \\ & \mathrm{sp}(H^\top) = s_h , \end{aligned} \tag{5}$$

where s_w, s_h are user parameters. The sparsity constraints control (i) to what extent basis functions are sparse, and (ii) how much each basis function contributes to the reconstruction of only a subset

of the data V. In a pattern recognition application the sparsity constraints effectively weight the desired generality over the specificity of the basis functions.

Instead of using equality constraints we will slightly generalize the constraints in this work to intervals $s_w^{\min} \leq \mathrm{sp}(W) \leq s_w^{\max}$ and $s_h^{\min} \leq \mathrm{sp}(H^\top) \leq s_h^{\max}$. This disburdens the user from choosing exact parameter values s_w, s_h, which can be difficult to find in realistic scenarios. In particular, it allows for $s_h^{\max} = s_w^{\max} = 1$, which may often be useful.

Consequently, we define the *sparsity-constrained NMF problem* as follows:

$$\begin{aligned} \min_{0 \leq W,H} \quad & \|V - WH\|_F^2 \\ \text{s.t.} \quad & s_w^{\min} \leq \mathrm{sp}(W) \leq s_w^{\max} \\ & s_h^{\min} \leq \mathrm{sp}(H^\top) \leq s_h^{\max}, \end{aligned} \tag{6}$$

where $s_w^{\min}, s_w^{\max}, s_h^{\min}, s_h^{\max}$ are user parameters. See Figure 1 and Section 6 for illustrations.

Efficient algorithms for solving (6) are developed in Section 5.

2.3 Supervised NMF

When NMF bases are used for recognition, it can be beneficial to introduce information about class membership in the training process. Doing so encourages NMF codes that not only describe the input data well, but also allow for good discrimination in a subsequent classification stage. We propose a formulation, similar to Fisher-NMF (Wang et al., 2004), that leads to particularly efficient algorithms in the training stage.

The basic idea is to restrict, for each class i and for each of its vectors j, the coefficients $H_{j\bullet}$ to a cone around the class center μ_i which is implicitly computed in the optimization process:

$$\begin{aligned} \min_{W,H} \quad & \|V - WH\|_F^2 \\ \text{s.t.} \quad & 0 \leq W, H \\ & \|\mu_i - H_{j\bullet}\|_2 \leq \lambda \|\mu_i\|_1 \quad \forall i, \forall j \in \mathrm{class}(i)\ . \end{aligned} \tag{7}$$

As will be explained in Section 5.4, these additional constraints are no more difficult from the viewpoint of optimization than are the previously introduced constraints in (6). On the other hand, they offer greatly increased classification performance for some problems (Section 6). Of course, if the application suggests, supervised NMF (7) can be conducted with the additional sparsity constraints from (6).

2.4 Assumptions

Throughout the remainder of this paper, the following assumptions are made:

1. The matrices $W^\top W$ and $HH^\top$ are positive definite.
2. $s_h^{\min} < s_h^{\max}$ and $s_w^{\min} < s_w^{\max}$ in (6).
3. The min-sparsity constraints in (6) are essential in the sense that each global optimum of the problem with min-sparsity constraints removed violates at least one such constraint on W and H.

The first assumption is introduced to simplify reasoning about convergence. In applications, it will regularly be satisfied as long as the number of basis functions r does not exceed size or dimension of the training data: $r \leq m,n$. Assumption two has been discussed above in connection with (6). Finally, assumption three is natural, because without the min-sparsity constraint problem (6) would essentially correspond to (1) which is less involved.

3. Solving Unconstrained NMF Problems

We introduced various forms of the NMF problem in the previous section. Next, we concentrate on practical algorithms to find locally optimal solutions. Unlike previous work, where variations of the gradient descent scheme were applied (Paatero, 1997; Hoyer, 2004), our algorithms' basic building blocks are convex programs for which fast and robust solvers exist. As a side effect, we avoid introducing additional optimization parameters like step-sizes or damping-constants, which is convenient for the user and increases robustness. As a result, just like with Lee and Seung's fast NMF algorithm (Lee and Seung, 2000), there is no artificial step size parameter to be determined, removing a potential source of errors and inefficiencies.

We next recall briefly the definition of quadratic programs, and then explain our approach to unconstrained NMF. Comparisons to existing work are reported in Section 6.

3.1 Convex Quadratic Programs (QP)

Convex quadratic programs (QP) are optimization problems involving convex quadratic objectives functions and linear constraints. In connection with unconstrained NMF (1), the QPs to be defined in the next section take the following general form:

$$\min_x \frac{1}{2} x^\top A x - b^\top x\,, \quad 0 \leq x\,, \quad A \text{ positive semidefinite.} \tag{8}$$

We denote the quadratic program (8) with parameters A, b:

$$\mathrm{QP}(A,b) \tag{9}$$

Note, that for QPs efficient and robust algorithms exist (e.g., Wright, 1996) and software for large-scale problems is available.

3.2 NMF by Quadratic Programming

The unconstrained NMF problem (1) reads:

$$\begin{aligned} \min_{W,H} \quad & \|V - WH\|_F^2 \\ \text{s.t.} \quad & 0 \leq W, H\,. \end{aligned}$$

Let us fix W and expand the objective function:

$$\begin{aligned} \|V - WH\|_F^2 &= \mathrm{tr}\big[(V - WH)^\top (V - WH)\big] \\ &= \mathrm{tr}(H^\top W^\top W H) - 2\mathrm{tr}(V^\top W H) + \mathrm{tr}(V^\top V)\,. \end{aligned}$$

Algorithm 3.1 QP-based NMF algorithm in pseudocode.

initialize W^0, $H^0 \geq 0$ randomly, $k \leftarrow 0$
repeat
  $H^{k+1} \leftarrow$ QP-result(W^k, V) using eqn. (10)
  $W^{k+1} \leftarrow$ QP-result(H^{k+1}, V) using eqn. (12)
  $k \leftarrow k+1$
until $\left| \|V - W^{k-1}H^{k-1}\| - \|V - W^kH^k\| \right| \leq \varepsilon$

Together with the non-negativity constraints $0 \leq H$, this amounts to solving the QPs:

$$\mathrm{QP}(W^\top W, W^\top V_{\bullet i}) \,, \quad i = 1, \ldots n \,, \tag{10}$$

for $H_{\bullet 1}, \ldots, H_{\bullet n}$. Conversely, fixing H we obtain:

$$\|V - WH\|_F^2 = \mathrm{tr}(WHH^\top W^\top) - 2\mathrm{tr}(VH^\top W^\top) + \mathrm{tr}(V^\top V) \,, \tag{11}$$

which amounts to solve the QPs:

$$\mathrm{QP}(HH^\top, HV_{i\bullet}) \,, \quad i = 1, \ldots, m \,, \tag{12}$$

for $W_{1\bullet}, \ldots, W_{m\bullet}$.

We emphasize that by using a batch-processing scheme *problems of almost arbitrary size* can be handled: The only hard limitation is the number of basis vectors r; the dimension of the basis vectors m can, in principle, grow almost arbitrarily large. This is particularly important for image processing applications where m represents the number of pixels which can be large.

The algorithm is summarized in Alg. 3.1. Note, that *the same* target function (1) is optimized alternately with respect to H and W. As a result, the algorithm performs a *block coordinate descent* (cf. Bertsekas, 1999). Furthermore, we may assume that the QPs in (10) and (12) are strictly convex, because typically $r \ll m, n$ (c.f. Section 2.4).

Proposition 1 *Under the assumptions of Section 2.4, the algorithm stated in Alg. 3.1 converges to a local minimum of problem* (1)*.*

Proof See Bertsekas (1999), Prop. 2.7.1. ■

4. Sparsity Constraints and Optimality

In this section we develop a geometric formulation of problem (6) in terms of second order cones that fits into the framework of reverse-convex programming (Section 5.1). We note that from the viewpoint of optimization problem (6) is considerably more involved than (1) because the lower sparsity bound imposed in terms of $s_w^{\min}$, $s_h^{\min}$ destroys convexity.

4.1 Second Order Cone Programms (SOCP) and Sparsity

The *second order cone* $\mathcal{L}^{n+1} \subset \mathbb{R}^{n+1}$ is the convex set (Lobo et al., 1998):

$$\mathcal{L}^{n+1} := \left\{ \begin{pmatrix} x \\ t \end{pmatrix} = (x_1, \ldots, x_n, t)^\top \,\middle|\, \|x\|_2 \leq t \right\} \,, \tag{13}$$

The problem of minimizing a linear objective function, subject to the constraints that several affine functions of the variables are required to lie in $\mathcal{L}^{n+1}$, is called a *second order cone program (SOCP)*:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f^\top x \\ \text{s.t.} \quad & \begin{pmatrix} A_i x + b_i \\ c_i^\top x + d_i \end{pmatrix} \in \mathcal{L}^{n+1}\,, \qquad i = 1,\ldots,m\,. \end{aligned} \tag{14}$$

Note, that efficient and robust solvers for solving SOCPs exist in software (Sturm, 2001; Mittelmann, 2003; Mosek 2005). Furthermore, additional linear constraints and, in particular, the condition $x \in \mathbb{R}^n_+$ are admissible, as they are special cases of constraints of the form (14). Our approach to sparsity-constrained NMF, to be developed below, is based on this class of convex optimization problems.

Motivated by the sparseness measure (2) and our goal to compute non-negative representations, we consider the family of *convex* sets parametrized by a sparsity parameter s:

$$\mathcal{C}(s) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{pmatrix} x \\ \frac{1}{c_{n,s}} e^\top x \end{pmatrix} \in \mathcal{L}^{n+1} \right\}\,, \quad c_{n,s} := \sqrt{n} - (\sqrt{n}-1)s\,. \tag{15}$$

Inserting the bounds (4) for s, we obtain from (3):

$$\mathcal{C}(0) = \left\{ \lambda e\,,\; 0 < \lambda \in \mathbb{R} \right\} \qquad \text{and} \qquad \mathbb{R}^n_+ \subset \mathcal{C}(1)\,. \tag{16}$$

This raises the question as to when non-negativity constraints must be imposed explicitly.

Proposition 2 *The set $\mathcal{C}(s)$ contains non-positive vectors $x \neq 0$ if:*

$$\frac{\sqrt{n} - \sqrt{n-1}}{\sqrt{n}-1} < s \leq 1\,, \quad n \geq 3\,. \tag{17}$$

Proof We observe that if $x \in \mathcal{C}(s)$, then $\lambda x \in \mathcal{C}(s)$ for arbitrary $0 < \lambda \in \mathbb{R}$, because $\|\lambda x\|_2 - e^\top(\lambda x)/c_{n,s} = \lambda(\|x\|_2 - e^\top x/c_{n,s}) \leq 0$. Hence it suffices to consider vectors x with $\|x\|_2 = 1$. According to definition (15), such vectors tend to be in $\mathcal{C}(s)$ the more they are aligned with e. Therefore, w.l.o.g., set $x_n = 0$ and $x_i = (n-1)^{-1/2}$, $i = 1,\ldots,n-1$. Then $x \in \mathcal{C}(s)$ if $c_{n,s} < \sqrt{n-1}$, and the result follows from the definition of $c_{n,s}$ in (15). Finally, for $n = 2$ the lower bound for s equals 1, that is no non-positive vectors exist for all admissible values of s. ■

The arguments above show that:

$$\mathcal{C}(s') \subseteq \mathcal{C}(s) \quad \text{for} \quad s' \leq s\,. \tag{18}$$

Thus, to represent the feasible set of problem (6), we combine the convex non-negativity condition with the convex upper bound constraint

$$\left\{ x \in \mathbb{R}^n_+ \;\middle|\; \mathrm{sp}(x) \leq s \right\} \;=\; \mathbb{R}^n_+ \cap \mathcal{C}(s) \tag{19}$$

and impose the *reverse-convex* lower bound constraint by subsequently removing $\mathcal{C}(s')$:

$$\left\{ x \in \mathbb{R}^n_+ \;\middle|\; s' \leq \mathrm{sp}(x) \leq s\,,\; s' < s \right\} \;=\; \left(\mathbb{R}^n_+ \cap \mathcal{C}(s)\right) \setminus \mathcal{C}(s')\,. \tag{20}$$

To reformulate (6), we define accordingly, based on (15):

$$\mathcal{C}_w(s) := \left\{ W \in \mathbb{R}^{m\times r} \;\middle|\; W_{\bullet i} \in \mathcal{C}(s)\,,\; i=1,\ldots,r \right\}, \tag{21}$$

$$\mathcal{C}_h(s) := \left\{ H \in \mathbb{R}^{r\times n} \;\middle|\; H_{i\bullet}^\top \in \mathcal{C}(s)\,,\; i=1,\ldots,r \right\}. \tag{22}$$

As a result, the sparsity-constrained NMF problem (6) now reads:

$$\begin{aligned} \min_{W,H} \quad & \|V-WH\|_F^2 \\ \text{s.t.} \quad & W \in \left(\mathbb{R}_+^{m\times r} \cap \mathcal{C}_w(s_w^{\max})\right) \setminus \mathcal{C}_w(s_w^{\min}) \\ & H \in \left(\mathbb{R}_+^{r\times n} \cap \mathcal{C}_h(s_h^{\max})\right) \setminus \mathcal{C}_h(s_h^{\min})\,. \end{aligned} \tag{23}$$

This formulation makes explicit that enforcing sparse NMF solutions introduces a single additional *reverse-convex* constraint for W and H, respectively. Consequently, not only the joint optimization of W,H is non-convex, but individual optimization of W and H is also.

4.2 Optimality Conditions

We state the first-order optimality conditions for problem (23). They will be used to verify the algorithms in Section 5.

To this end, we define in view of (15) and (23):

$$f(W,H) := \|V-WH\|_F^2\,, \tag{24a}$$

$$Q := Q_w \times Q_h\,, \quad Q_w := \mathbb{R}_+^{m\times r} \cap \mathcal{C}_w(s_w^{\max})\,,\; Q_h := \mathbb{R}_+^{r\times n} \cap \mathcal{C}_h(s_h^{\max})\,, \tag{24b}$$

$$G_w(W) := \left(\|W_{\bullet 1}\|_2 - \frac{1}{c_{n,s_w^{\min}}}\|W_{\bullet 1}\|_1\,,\; \ldots\,,\; \|W_{\bullet r}\|_2 - \frac{1}{c_{n,s_w^{\min}}}\|W_{\bullet r}\|_1 \right)^\top, \tag{24c}$$

$$G_h(H) := \left(\|H_{1\bullet}\|_2 - \frac{1}{c_{n,s_h^{\min}}}\|H_{1\bullet}\|_1\,,\; \ldots\,,\; \|H_{r\bullet}\|_2 - \frac{1}{c_{n,s_h^{\min}}}\|H_{r\bullet}\|_1 \right)^\top. \tag{24d}$$

Note that Q represents the convex constraints of problem (23) while $G_w(W)$ and $G_h(H)$ are non-negative exactly when sparsity is at least $s_w^{\min}$ and $s_h^{\min}$. For non-negative W and H computing the ℓ_1 norm is a linear operation, that is, $W \geq 0 \Rightarrow \|W_{\bullet 1}\|_1 \equiv \langle W_{\bullet 1}, e\rangle$.

Problem (23) then can be rewritten so as to directly apply standard results from variational analysis (Rockafellar and Wets, 1998):

$$\min_{(W,H)\in Q} f(W,H)\,, \quad G_w(W) \in \mathbb{R}_+^r\,, \quad G_h(H) \in \mathbb{R}_+^r\,. \tag{25}$$

With the corresponding Lagrangian L and multipliers λ_w, λ_h,

$$L(W,H,\lambda_w,\lambda_h) = f(W,H) + \lambda_w^\top G_w(W) + \lambda_h^\top G_h(H)\,, \tag{26}$$

the first-order conditions for a locally optimal point (W^*,H^*) are:

$$-\left(\frac{\partial L}{\partial W}, \frac{\partial L}{\partial H}\right)^\top \in N_Q(W^*,H^*) = N_{Q_w}(W^*) \times N_{Q_h}(H^*)\,, \tag{27a}$$

$$G_w(W^*) \in \mathbb{R}_+^r\,,\; G_h(H^*) \in \mathbb{R}_+^r\,, \tag{27b}$$

$$\lambda_w^*, \lambda_h^* \in \mathbb{R}_-^r\,, \tag{27c}$$

$$\left\langle \lambda_w^*, G_w(W^*) \right\rangle = 0\,,\; \left\langle \lambda_h^*, G_h(H^*) \right\rangle = 0\,, \tag{27d}$$

where $N_X(x)$ denotes the normal cone to a set X at point x (see, e.g., Rockafellar and Wets, 1998).

5. Algorithms

In this section, we present two algorithms for solving problem (23). The algorithm discussed in Section 5.2 is very efficient and computes a good local optimum if it converges. However, it may oscillate in rare cases. Therefore, Section 5.3 presents a slightly less efficient but "save" and convergent algorithm. Both algorithms can also be applied to the supervised setting just by adding additional convex constraints—see Sections 2.3 and 5.4.

5.1 Reverse-Convex Programs (RCP)

The computational framework of our algorithms is reverse-convex programming (RCP) which considers problems of the form

$$\min_x f(x)\,, \quad g(x) \leq 0\,, \quad 0 \leq h(x)\,, \tag{28}$$

where f, g, and h are convex (Singer, 1980; Tuy, 1987; Horst and Tuy, 1996). Geometrically, the feasible set X has the form $X = G \setminus H$ where G and H are convex sets.

RPCs are closely related to the class of d.c. programs (Toland, 1979; Hiriart-Urruty, 1985; Tuy, 1995; Yuille and Rangarajan, 2003). In fact, a d.c. program in standard form can be written as RCP:

$$\begin{array}{ll} \min\limits_{x} & f_1(x) - f_2(x) \\ \text{s.t.} & g(x) \leq 0 \end{array} \quad \Rightarrow \quad \begin{array}{ll} \min\limits_{x,z} & f_1(x) - z \\ \text{s.t.} & g(x) \leq 0 \\ & 0 \leq f_2(x) - z\,. \end{array} \tag{29}$$

Existing strategies for globally optimizing RPCs (Horst and Tuy, 1996) can be only applied for small or medium-sized problems. Accordingly, our algorithms below focus on the realistic goal to efficiently compute a local minimum for larger learning problems.

5.2 Cone Programming with Tangent-Plane Constraints

In this section, we present an optimization scheme for sparsity-controlled NMF which relies on linear approximation of the reverse-convex constraint in (23). As in the case of unconstrained NMF, we alternately minimize (23) with respect to W and H. It thus suffices to concentrate on the H-step:

$$\begin{array}{ll} \min\limits_{H} & f(H) = \|V - WH\|_F^2 \\ \text{s.t.} & H \in \left(\mathbb{R}_+^{r\times n} \cap \mathcal{C}_h(s_h^{\max})\right) \setminus \mathcal{C}_h(s_h^{\min})\,. \end{array} \tag{30}$$

Recall the assumptions made in Section 2.4.

5.2.1 TANGENT-PLANE CONSTRAINT (TPC) ALGORITHM

The tangent-plane constraint algorithm solves a sequence of SOCPs where the convex max-sparsity constraints are modeled as second order cones and the min-sparsity cone is linearized: In an initialization step we solve a SOCP ignoring the min-sparsity constraint and examine the solution. For the rows of H that violate the min-sparsity constraint we compute tangent planes to the min-sparsity cone and solve the SOCP again with additional tangent-plane constraints in place. This is repeated

Algorithm 5.1 Tangent-plane approximation algorithm in pseudocode.

$H^0 \leftarrow$ solution of (32), $J^0 \leftarrow \emptyset$, $k \leftarrow 0$
repeat
  $\tilde{H}^k \leftarrow H^k$
  repeat
    $J^k \leftarrow J^k \cup \{ j \in 1,\ldots,r : \tilde{H}^k_{j\bullet} \in \mathcal{C}(s_h^{\min}) \}$
    $t_j^k \leftarrow \nabla \mathcal{C}(s_h^{\min})(\pi(\tilde{H}^k_{j\bullet})) \quad \forall j \in J^k$
    $\tilde{H}^k \leftarrow$ solution of (33) replacing H^k by $\tilde{H}^k$
  until $\tilde{H}^k$ is feasible
  $H^{k+1} \leftarrow \tilde{H}^k$, $J^{k+1} \leftarrow J^k$, $k \leftarrow k+1$
until $|f(H^k) - f(H^{k-1})| \leq \varepsilon$

until all necessary tangent-planes are identified. During iteration we repeatedly solve this SOCP where the tangent planes are permanently updated to follow their corresponding entries in H: This ensures that they constrain the feasible set no more than necessary. This process of updating the tangent planes and computing new estimates for H is repeated until the objective function no longer improves.

The TPC algorithm consists of the following steps:

- **Initialization.** The algorithm starts by setting $s_h^{\min} = 0$ in (30), and by computing the global optimum of the convex problem: $\min f(H),\ H \in \mathcal{C}_h(s_h^{\max})$, denoted by $\tilde{H}^0$. Rewriting the objective function:

$$\begin{aligned} f(H) &= \|V^\top - H^\top W^\top\|_F^2 \\ &= \|\mathrm{vec}(V^\top) - (W \otimes I)\mathrm{vec}(H^\top)\|_2^2 \,, \end{aligned} \tag{31}$$

  we observe that $\tilde{H}^0$ solves the SOCP:

$$\min_{H,z} z \,, \quad H \in \mathbb{R}_+^{r\times n} \cap \mathcal{C}_h(s_h^{\max}) \,, \quad \begin{pmatrix} \mathrm{vec}(V^\top) - (W \otimes I)\mathrm{vec}(H^\top) \\ z \end{pmatrix} \in \mathcal{L}^{r\times n+1} \,. \tag{32}$$

  Note that $\tilde{H}^0$ will be infeasible w.r.t. the original problem because the reverse-convex constraint of (30) is not imposed in (32). We determine the index set $J^0 \subseteq \{1,\ldots,r\}$ of those vectors $\tilde{H}^0_{j\bullet}$ violating the reverse-convex constraint, that is $\tilde{H}^0_{j\bullet} \in \mathcal{C}(s_h^{\min})$.
  Let $\pi(\tilde{H}^0_{j\bullet})$ denote the projections of $\tilde{H}^0_{j\bullet}$ onto $\partial\mathcal{C}(s_h^{\min}), \forall j \in J^0$ (Hoyer, 2004). Further, let t_j^0 denote the tangent plane normals to $\mathcal{C}_h(s_h^{\min})$ at these points, and $H^0 \leftarrow \pi(\tilde{H}^0)$ a feasible starting point. We initialize the iteration counter $k \leftarrow 0$.

- **Iteration.** Given J^k, $k = 0,1,2,\ldots$, we once more solve (32) with additional linear constraints enforcing feasibility of each $H^k_{j\bullet}$, $j \in J^k$:

$$\begin{aligned} &\min_{H,z} z \,, \quad H \in \mathbb{R}_+^{r\times n} \cap \mathcal{C}_h(s_h^{\max}) \,, \quad \begin{pmatrix} \mathrm{vec}(V^\top) - (W \otimes I)\mathrm{vec}(H^\top) \\ z \end{pmatrix} \in \mathcal{L}^{r\times n+1} \\ &\qquad \left\langle t_j^k, H_{j\bullet} - \pi(H^k_{j\bullet}) \right\rangle \geq 0 \,, \quad \forall j \in J^k \,. \end{aligned} \tag{33}$$

Let us denote the solution by $\tilde{H}^{k+1}$. It may occur that because of the additional constraints new rows $\tilde{H}^{k+1}_{j\bullet}$ of $\tilde{H}^{k+1}$ became infeasible for indices $j \notin J^k$. In this case we augment J^k accordingly, and solve (33) again until the solution is feasible.

Note that we never remove indices from J^k. Instead, we permanently re-adjust the corresponding tangent plane constraints t^k_j, setting $t^k_j \leftarrow \nabla_C(s^{\min}_h)(\pi(\tilde{H}^k_{j\bullet})), \forall j \in J^k$. This ensures that the constraints are not active at termination unless a component $\tilde{H}^k_{j\bullet}$ is actually on the boundary of the min-sparsity cone.

Finally, we rectify the vectors $\tilde{H}^{k+1}_{j\bullet}$ by projection, as in the initialization, provided this further minimizes the objective function f. The result is denoted by H^{k+1} and the corresponding index set by J^{k+1}. At last, we increment the iteration counter: $k \leftarrow k+1$

- **Termination criterion.** We check whether H^{k+1} satisfies the termination criterion $|f(H^{k+1}) - f(H^k)| < \varepsilon$. If not, we continue the iteration.

The algorithm is summarized in pseudocode in Alg. 5.1.

Remark 3 *The projection operator π mapping a point $x \in \mathbb{R}^m_+$ onto the boundary of the min-sparsity cone can be implemented using either the method of Hoyer (2004) or a fast approximation. In the approximation, used exclusively in our experiments (Section 6), each element x_i in x is exponentiated and replaced by $c \cdot x_i^\alpha$, with $\alpha \geq 1$ chosen such that the min-sparsity constraint is not violated. The factor $c = c(x,\alpha)$ ensures that the ℓ_2-norm of x is not affected by this transformation.*

Remark 4 *Problems* (32) *and* (33) *are formulated in terms of the* rows *of H, complying with the sparsity constraints* (22)*. Unfortunatly, matrix $W \otimes I$ in* (32) *is not block-diagonal, so we cannot separately solve for each $H_{j\bullet}$. Nevertheless, the algorithm is efficient (cf. Section 6).*

Remark 5 *Multiple tangent-planes with reversed signs can also be used to approximate the convex max-sparsity constraints. Then problem* (33) *reduces to a QP. Except for solvers for linear programs, QP solvers are usually among the most efficient mathematical programming codes available. Thus, for a given large-scale problem additional speed might be gained by using QP instead of SOCP solvers. In particular, this holds for the important special case when no non-trivial max-sparsity constraints are specified at all (i.e., $s^{\max}_h = s^{\max}_w = 1$).*

Remark 6 *A final remark concerns the termination criterion (Step 10 in Alg. 5.1). While in principle it can be chosen almost arbitrarily rigid, an overly small ε might not help in the overall optimization w.r.t. W* and *H. As long as, e.g., W is known only approximately, we need not compute the corresponding H to the last digit. In our experiments we chose relatively large ε so that the outer loop (steps 2 to 10 in Tab. 5.1) was executed only once or twice before the variable under optimization was switched.*

5.2.2 CONVERGENCE PROPERTIES

In the following discussion we use matrices $T^k = (t^k_j)_{j\in 1,\ldots,r}$ that have tangent plane vector t^k_j as j-th column when $j \in J^k$ and zeros elsewhere.

Proposition 7 *Under the assumptions stated in Section 2.4 Algorithm 5.1 yields a sequence $H^1, H^2, \ldots$ of feasible points, every cluster point of which is a local optimum.*

Proof Our proof follows (Tuy, 1987, Prop. 3.2). First, note that for every $k > 0$ the solution H^k of iteration k is a feasible point for the SOCP solved in iteration $k+1$. Therefore, $\{f(H^k)\}_{k=1,\dots}$ is a decreasing sequence, bounded from below and thus convergent. By the first assumption in Section 2.4, the objective function (31) is strictly convex, because $(W \otimes I)^\top (W \otimes I) = (W^\top W \otimes I)$ has positive eigenvalues $\lambda_i(W^\top W)\lambda_j(I) = \lambda_i(W^\top W)\,,\ \forall i,j$ (Graham, 1981). Consequently, $\{H : f(H) \leq f(H^k)\}$ is bounded for each k. Let $\{H^{k_\nu}\}_{\nu=1,\dots}$ denote a subsequence of solutions to (33) converging to a cluster point $\bar{H}$, and let $\{T^{k_\nu}\}_{\nu=1,\dots}$ resp. $\bar{T}$ denote the corresponding tangent planes. We have

$$f(H^{k_\nu}) \leq f(H), \qquad \forall H \in \mathcal{C}(s_h^{\max}) \text{ with } T^{k_\nu\top} H \geq 0, \tag{34}$$

and in the limit $\nu \to \infty$

$$f(\bar{H}) \leq f(H), \qquad \forall H \in \mathcal{C}(s_h^{\max}) \text{ with } \bar{T}^\top H \geq 0. \tag{35}$$

Note that the constraints active in $\bar{T}$ correspond to entries $\bar{H}_{j\bullet} \in \partial\mathcal{C}(s_h^{\min}), j \in J$. According to (35) there is no feasible descent direction at $\bar{H}$ and, thus, it must be a stationary point. Since the target function is quadratic positive-semidefinite by assumption, $\bar{H}$ is an optimum. ■

Thus, the TPC algorithm yields locally optimal W and H. However, this holds for the *individual* optimizations of W and H only. The same cannot be claimed for the *alternating sequence* of optimizations in W and H necessary to solve (6). Because of the intervening optimization of, e.g., W, we cannot derive a bound on $f(H)$ from a previously found locally optimal H. In rare cases, this can lead to undesirable oscillations. When this happens, we must introduce some damping term or simply switch to the convergent sparsity maximization algorithm described in Section 5.3.

On the other hand, if the TPC algorithm converges it does in fact yield a locally optimal solution.

Proposition 8 *If the TPC algorithm converges to a point (W^*, H^*) and the assumptions stated in Section 2.4 hold then (W^*, H^*) satisfies the first-order necessary optimality conditions 4.2 of problem* (23).

Proof For H^* we have from (33) using the notation from Sec. 4.2

$$H^* = \arg\min_{H \in Q_h} \quad \|V - W^* H\|_F^2 \tag{36a}$$

$$\text{s.t.} \quad \left\langle t_j^k,\, H_{j\bullet} - \pi(H_{j\bullet}^{*k}) \right\rangle \geq 0\,, \quad \forall j \in J^k. \tag{36b}$$

Since $t_j^k = \nabla \mathrm{sp}(\pi(H_{j\bullet}^{*k})^\top)$ constraint (36b) ensures that the min-sparsity constraint is enforced at H^* when necessary (c.f. Prop. 7). Applying π on each column of H^{*k} simultaneously and introducing Lagrange parameters $\lambda_f^*,\ \tilde{\lambda}_h^*$ for this convex problem yields that the result of (36) adheres to the first-order condition

$$\begin{aligned} -\lambda_f^* \frac{\partial}{\partial H} f(W^*, H^*) - \tilde{\lambda}_h^* \frac{\partial}{\partial H} \nabla \mathrm{sp}(\pi(H^{*k})^\top)(H^* - \pi(H^{*k})) \;&\in\; N_{Q_h}(H^*) \\ \Leftrightarrow \quad -\lambda_f^* \frac{\partial}{\partial H} f(W^*, H^*) - \tilde{\lambda}_h^* \nabla \mathrm{sp}(\pi(H^{*k})^\top) \;&\in\; N_{Q_h}(H^*) \\ \Leftrightarrow \quad -\frac{\partial}{\partial H}\left(\lambda_f^* f(W^*, H^*) + \left\langle \tilde{\lambda}_h^*, G_h(H^*) \right\rangle\right) \;&\in\; N_{Q_h}(H^*) \end{aligned} \tag{37}$$

which coincides with the condition on H in (27a). The W-part can be treated in the same way. ■

5.3 Sparsity-Maximization Algorithm

In this section we present an optimization scheme for sparsity-controlled NMF for which global convergence can be proven, even when W and H are optimized alternately. Here, global convergence means that the algorithm *always converges* to a *local* optimum. As in the previous sections, we assume our standard scenario (Section 2.4) and independently optimize for W and for H. Thus, it suffices to focus on the H-step.

Our algorithm is inspired by the reverse-convex optimization scheme suggested by Tuy (1987). This scheme is a *global* optimization algorithm in the sense that it finds a true global optimum. However, as already pointed out in Tuy (1987), it does so at a considerable computational cost. Furthermore, it does not straightforwardly generalize to *multiple* reverse-convex constraints that are essential for sparsity-controlled NMF. We avoid these difficulties by confining ourselves with a *locally* optimal solution.

The general idea of our algorithm is as follows: After an initialization step, it alternates between two convex optimization problems. One maximizes sparsity subject to the constraint that the objective value must not increase. Dually, the other optimizes the objective function under the condition that the min-sparsity constraint may not be violated.

5.3.1 SPARSITY-MAXIMIZATION ALGORITHM (SMA)

The sparsity-maximization algorithm is described below. A summary in pseudocode is outlined in Alg. 5.2

- **Initialization.** For initialization we start with any point $H^0 \in \partial\mathcal{C}\,(s_h^{\min})$ on the boundary of the min-sparsity cone. It may be obtained by solving (30) without the min-sparsity constraints and projecting the solution onto $\partial\mathcal{C}\,(s_h^{\min})$. We set $k \leftarrow 0$.

- **First step.** Given the current iterate H^k, we solve the SOCP

$$\begin{aligned} \max_{H,t} \quad & t \\ \text{s.t.} \quad & H \in \mathbb{R}_+^{r\times n} \cap \mathcal{C}_h(s_h^{\max}) && \text{(38a)} \\ & f(H) \le f(H^k) && \text{(38b)} \\ & t \le \mathrm{sp}(H^k_{j\bullet}) + \langle \nabla_{H_{j\bullet}} \mathrm{sp}(H^k_{j\bullet}), H_{j\bullet} - H^k_{j\bullet} \rangle\,, \quad j = 1,\ldots,r && \text{(38c)} \end{aligned}$$

  where constraint (38b) ensures that the objective value will not deteriorate. In standard form this constraint translates to

$$\begin{pmatrix} \mathrm{vec}(V^\top) - (W \otimes I)\mathrm{vec}(H^\top) \\ f(H^k) \end{pmatrix} \in \mathcal{L}^{rn+1}. \tag{39}$$

  We denote the result by H^{sp}. Note that this step maximizes sparsity in the sense that $\mathrm{sp}(H^k) \le \mathrm{sp}(H^{\mathrm{sp}})$, due to (38c) and the convexity of $\mathrm{sp}(\cdot)$.

- **Second step.** While the intermediate solution H^{sp} satisfies the min-sparsity constraint, it may not be an optimal local solution to the overall problem. Therefore, in a second step, we solve

the SOCP

$$\begin{aligned} \min_{H} \quad & f(H) \\ \text{s.t.} \quad & H \in \mathbb{R}_+^{r\times n} \cap \mathcal{C}_h(s_h^{\max}) && \text{(40a)} \\ & \|H_{j\bullet} - H^{\mathrm{sp}}_{j\bullet}\|_2 \leq \min_{q\in\mathcal{C}(s_h^{\min})} \|q - H^{\mathrm{sp}}_{j\bullet}\|_2 \,, \quad j = 1,\ldots,r && \text{(40b)} \end{aligned}$$

which reads in standard form

$$\begin{aligned} \min_{H,t} \quad & t \\ \text{s.t.} \quad & \begin{pmatrix} \mathrm{vec}(V^\top) - (W\otimes I)\mathrm{vec}(H^\top) \\ t \end{pmatrix} \in \mathcal{L}^{rn+1} \\ & \begin{pmatrix} H_{j\bullet} - H^{\mathrm{sp}}_{j\bullet} \\ \min_{q\in\mathcal{C}(s_h^{\min})} \|q - H^{\mathrm{sp}}_{j\bullet}\|_2 \end{pmatrix} \in \mathcal{L}^{n+1}, \quad \forall j \\ & H \in \mathbb{R}_+^{r\times n} \cap \mathcal{C}_h(s_h^{\max}). \end{aligned} \qquad \text{(41)}$$

Here, the objective function f is minimized subject to the constraint that the solution must not be too distant from H^{sp}. To this end, the non-convex min-sparsity constraint is replaced by a convex max-distance constraint (40b), in effect defining a spherical *trust region*. The radius $\min_{q\in\mathcal{C}(s_h^{\min})} \|q - H^{\mathrm{sp}}_{j\bullet}\|_2$ of the trust region is computed by a small SOCP.

- **Termination.** As long as the termination criterion $|f(H^k) - f(H^{k-1})| \leq \varepsilon$ is not met we continue with the first step.

When the algorithm terminates a locally optimal H for the current configuration of W is found. In subsequent runs we will not initialize the algorithm with an arbitrary H^0, but simply continue alternating between step one and step two using the current best estimate for H as a starting point[1]. This way, we can be sure that the sequence of H^k is monotonous, even when W is occasionally changed in between.

Remark 9 *The requirement that the feasible set has an non-empty interior is important. If* $s_h^{\max} = s_h^{\min}$*, the approximate approach in* (38) *breaks down, and each iteration just yields* $H^k = H^{\mathrm{sp}} = H^{k+1}$*. In this situation, it is necessary to temporarily weaken the max-sparsity constraint. Fortunately, max-sparsity constraints seem to be less important in many applications.*

5.3.2 CONVERGENCE PROPERTIES

We check the convergence properties of the SMA.

Proposition 10 *Under the assumptions stated in Section 2.4 and 4.2, the SMA (Alg. 5.2) converges to a point* (W^*, H^*) *satisfying the first-order necessary optimality conditions of problem* (23)*.*

1. Note that while such a scheme could be implemented with TPC as well, it would perform poorly in practice: Without proper initialization TPC locks too early onto bad local optima.

Algorithm 5.2 Sparsity-maximization algorithm in pseudocode.

$H^0 \leftarrow$ solution of (32) projected on $\partial C_h(s_h^{\min})$, $k \leftarrow 0$
repeat
  $H^{\mathrm{sp}} \leftarrow$ solution of (38)
  $H^{k+1} \leftarrow$ solution of (40)
  $k \leftarrow k+1$
until $|f(H^k) - f(H^{k-1})| \leq \varepsilon$

Proof Under the assumptions stated in Section 2.4, the feasible set is bounded. Furthermore, Alg. 5.2, alternately applied to the optimization of W and H, respectively, computes a sequence of feasible points $\{W^k, H^k\}$ that steadily decreases the objective function value. Thus, by taking a convergent subsequence, we obtain a cluster point (W^*, H^*) whose components separately optimize (38) when the other component is held fixed. It remains to check that conditions (27) are satisfied after convergence.

We focus on H without loss of generality. Taking into account the additional non-negativity condition, condition (38c) is equivalent to $t \leq \mathrm{sp}(H_{j\bullet})$, because $\mathrm{sp}(\cdot)$ is convex. Moreover, $t = s_h^{\min}$ because after convergence of iterating (38) and (40), the min-sparsity constraint will be active for some of the indices $j \in \{1, \ldots, r\}$. Therefore, using the notation (24), the solution to problem (38) satisfies

$$\max_{t, H \in Q_h} t^* = s_h^{\min}, \quad f(H^k) - f(H^*) \geq 0, \quad G_h(H^*) \in \mathbb{R}^r_+ . \tag{42}$$

Using multipliers $\lambda_f^*, \tilde{\lambda}_h^*$, the relevant first-order condition with respect to H is:

$$-\frac{\partial}{\partial H}\left(\lambda_f^* f(H^*) + \left\langle \tilde{\lambda}_h^*, G_h(H^*) \right\rangle\right) \in N_{Q_h}(H^*) . \tag{43}$$

This corresponds to the condition on H in (27a). The W-part can be handled the same way. ■

Remark 11 *While convergence is guaranteed and high-quality results are obtained (Section 6), SMA can be slower than the TPC method presented in the previous section. This is especially the case when $s_h^{\max} \approx s_h^{\min}$. Then, in order to solve a problem most efficiently, one will start with the tangent-plane method and only if it starts oscillating switch to sparsity-maximization mode.*

5.4 Solving Supervised NMF

The supervised NMF problem (7) is solved either by the tangent-plane constraint or the sparsity-maximization algorithm presented above. We merely add constraints ensuring that the coefficients belonging to class i, abbreviated $H_{(i)} \in \mathbb{R}_+^{r \times n_i}$ below, stay in a cone centered at mean $\mu_i = 1/n_i H_{(i)} e$. Then, the supervised constraint in (7) translates to

$$\begin{pmatrix} 1/n_i H_{(i)} e - H_{j*} \\ \lambda / n_i e^\top H_{(i)} e \end{pmatrix} \in \mathcal{L}^{n+1}, \quad \forall i, \forall j \in \mathrm{class}(i) . \tag{44}$$

It is an important advantage that the algorithms above can easily be augmented by various convex constraints (e.g., Heiler and Schnörr, 2005).

		$r=2$	$r=5$	$r=10$	$r=15$	$r=20$	$r=25$
MU	$\varepsilon=10^{-1}$	0.13	0.15	0.17	0.30	0.49	0.68
MU	$\varepsilon=10^{-2}$	0.21	0.44	0.60	0.71	0.82	0.91
MU	$\varepsilon=10^{-3}$	0.36	0.29	0.33	0.37	0.41	0.45
MU	$\varepsilon=10^{-4}$	0.69	2.46	3.18	3.77	4.47	5.11
MU	$\varepsilon=10^{-5}$	2.64	4.29	5.99	8.08	9.70	11.83
QP	$\varepsilon=10^{-1}$	0.20	0.35	0.67	1.14	1.74	2.53
QP	$\varepsilon=10^{-2}$	0.17	0.36	0.67	1.13	1.74	2.52
QP	$\varepsilon=10^{-3}$	0.26	0.45	0.79	1.27	1.97	2.60
QP	$\varepsilon=10^{-4}$	0.36	1.25	1.57	2.54	3.88	5.61
QP	$\varepsilon=10^{-5}$	1.46	2.05	2.56	4.55	7.62	12.21

Table 1: **Unconstrained NMF.** Comparison between QP algorithm and multiplicative updates (MU). A medium-sized computer vision data set, $V \in \mathbb{R}^{1200\times 150}$, was factorized using multiplicative updates and the QP algorithm (Alg. 3.1) using different numbers of basis functions r and different accuracies ε. Average run time in seconds over 10 repeated runs is reported. Overall, the QP algorithm shows similar performance to multiplicative updates.

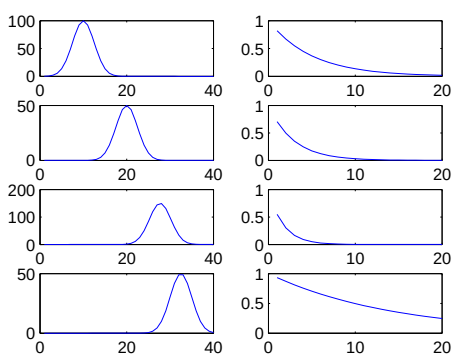
6. Experiments

In this section we perform comparisons with established algorithms on artificial and on real-world data sets to validate our results from a practical point of view. We also provide evidence that the local sparsity maximization seems not prone to end in bad local optima. Finally, we show that the supervised constraints from eqn. (7) can lead to NMF codes that are more useful for recognition.

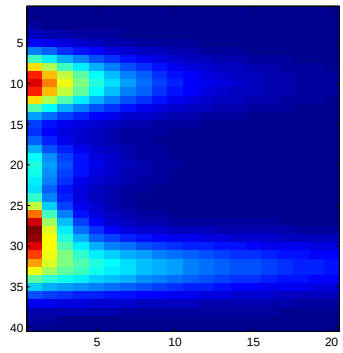
6.1 Unconstrained NMF

In a first experiment, we validated that the quadratic programming algorithm (Tab. 3.1) yields results similar to the fast and stable *multiplicative update* (MU) algorithm by Lee and Seung (2000). To this end, we factorized a data set from facial expression classification (Buciu and Pitas, 2004; Lyons et al., 1998) using both algorithms on subproblems of different sizes and different requirements for accuracy. To make a fair comparison, we ensured that the reconstruction error $f(W,H) = \|V - WH\|_F$ of the QP algorithm was at least as small as the corresponding error from a previous run of the MU algorithm. We performed 10 repeated runs, each time starting from a randomly chosen initialization W,H that was identical for both methods. The results[2] are summarized in Tab. 1. Both methods perform well on the data set. MU has an edge with the smaller problems, while QP has advantages when high accuracies are required. Overall, both methods are practical for solving real-world problems.

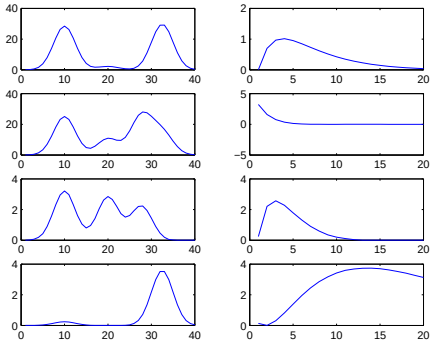
2. All run times are reported in seconds using a 3 GHz Pentium IV running Linux, Matlab, and the Mosek 3.1 solvers (Mosek 2005). In preliminary experiments we found that the SeDuMi SOCP solver (Sturm, 2001) and the CPLEX QP solver (Cplex 2001) can be used as well.



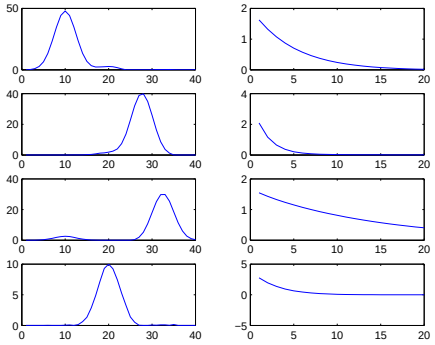
(a) Ground truth W and H.



(b) $X = WH + \eta$.



(c) Recovered bases using $s_w^{\min} = 0.0$.



(d) Recovered bases using $s_w^{\min} = 0.6$.

Figure 2: **Paatero experiments.** The entries of the factors W and H are displayed (Figure 2(a)) as well as the resulting data matrix X (Figure 2(b)). In the experiments, a small amount of Gaussian noise $\eta \sim \mathcal{N}(0, 0.1)$ is added to the factors. The results for different values of the min-sparsity constraint are shown in Figure 2(c) and 2(d): Only an active constraint allows to correctly recover W and H.

6.2 Sparsity-Controlled NMF

To examine the performance of the sparsity-controlled NMF algorithms we repeated an experiment suggested by Paatero (1997). Here, a synthetic data set consisting of products of Gaussian and exponential distributions is analyzed using NMF. This data set (Figure 2(a)) is designed to resemble data from spectroscopic experiments in chemistry and physics and is *not* easily analyzed: without prior knowledge, NMF is reported to fail to recover the original factors in the data set. As a remedy, Paatero hints that a "target shape" extension to NMF is beneficial. We will show that for this data set imposing an additional min-sparsity constraint on W is sufficient to lead to correct factorizations.

In Tab. 2 we report the results for 10 repeated runs of the tangent-plane constraints and the sparsity-maximization algorithm using different choices of the min-sparsity constraint. The most important figure is the number of correct recoveries of the basis functions. We counted a NMF-result correct if it showed the correct number of modes at the correct locations. First note that,

$s_w^{\min}$		0.0	0.2	0.4	0.6	0.8
TPC	# correct results	0	0	0	7	4
TPC	med run time (sec.)	20.7	16.8	14.5	53.7	117.2
TPC	min obj. value	0.26	0.26	0.24	0.25	210.57
SMA	# correct results	0	0	0	9	0
SMA	med run time (sec.)	142.0	113.3	60.3	54.1	14.0
SMA	min obj. value	0.25	0.26	0.24	0.26	161.5

Table 2: **Sparsity-controlled NMF.** Statistics of the Paatero experiment collected over 10 runs for the tangent-plane constraints (TPC) and the sparsity-maximization algorithm (SMA). The number of correct reconstructions (see text), the median run time, and the best objective value obtained are reported for different choices of the sparsity constraint. Correct reconstructions are found in seven resp. nine out of ten trials for a sufficiently strong sparsity constraint: $s_w^{\min} = 0.6$. This quota can be increased at the expense of longer running times.

consistent with Paatero (1997), the basis functions are not recovered correctly without additional prior information in the form of constraints. Also, the objective value $f(W,H) = \|V - WH\|_F$ is *not* indicative of correct results. Only for the extremely sparse case with $s_w^{\min} = 0.8$ did we obtain noticeably worse objective values. However, not shown in the table, for the interesting case $s_w^{\min} = 0.6$ the objective values of the correct recoveries were all below 0.4 while from the remaining incorrect recoveries each was above 5.0. Thus, while the objective value is not useful for model selection purposes it seems to indicate good solutions once a suitable model is defined.

Finally, we point out that for the correct value of the sparsity constraint the number of correct recoveries is essentially a function of the stopping parameters. With more conservative stopping parameters one can ensure that in every single case bases are recovered correctly. But then running times increase. In our experiment we favored a short run time over perfect success rate. Accordingly, the best combination of W and H was found after just 9 seconds of computation.

6.3 Global Approaches

A potential source of difficulties with the sparsity-maximization algorithm is that the lower bound on sparsity is optimized only locally in (38). Through the proximity constraint in (40) the amount of sparsity obtained in effect limits the step size of the algorithm. Insufficient sparsity optimization may, in the worst case, lead to convergence to a bad local optimum.

To see if this worst-case scenario is relevant in practice, we discretized the problem by sampling the sparsity cones using rotated and scaled version of the current estimate H^k and then evaluated $f(W,H)$ using samples from each individual sparsity cone. Then we picked one sample from each cone and computed (38) replacing the starting point H^k by the sampled coordinates. For an exhaustive search on r cones each sampled with s points we have s^r starting points to consider.

For demonstration we used the artificial data set from Paatero (1997) consisting of products of Gaussian and exponential functions (Figure 2). This data set is suitable since it is not overly large and sparsity control is crucial for its successful factorization.

In the sparsity-maximization algorithm we first sampled the four sparsity cones corresponding to each basis function of the data for $s_w \geq 0.6$ sparsely, using only 10 rotations on each cone. We

$s_{w,h}^{\min,\max}$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
time TPC	42.08	36.68	38.00	65.23	58.90	51.16	66.77	71.41	111.35
time PGD	133.00	292.73	2046.92	1269.06	453.38	713.41	568.20	129.44	1463.89
quotient	3.16	7.98	53.87	19.46	7.70	13.94	8.51	1.81	13.15
error TPC	0.19	0.17	0.18	0.19	0.43	0.72	0.89	1.01	1.07
error PGD	0.21	0.16	0.17	0.19	0.48	0.79	0.95	1.05	1.08
error quotient	1.09	0.98	0.94	0.99	1.11	1.10	1.07	1.04	1.01

Table 3: **Comparison.** Tangent-plane constraint (TPC) algorithm and projected gradient descent (PGD). The algorithms were used to find sparse decompositions of the CBCL face data set. TPC outperforms PGD w.r.t. computational effort (measured in seconds) while keeping errors small.

then combined the samples on each cone in each possible way and evaluated g for all corresponding starting points. In a second experiment we placed 1000 points on each sparsity cone, and randomly selected 10^4 combinations as starting points. The best results obtained over four runs and 80 iterations with our local linearization method used in SMA and the sparse enumeration (first) and the sampling (second) strategy, are reported below:

Algorithm	min-sparsity	objective value
SMA	0.60	0.24
sparse enumeration	0.60	0.26
sampling	0.60	0.26

We see that the local sparsity maximization in SMA yields results comparable to the sampling strategies. In fact, it is better: Over four repeated runs the sampling strategies each produced outliers with very bad objective values (not shown). This is most likely caused by severe under-sampling of the sparsity cones. This problem is not straightforward to circumvent: With above sampling schemes a run over 80 iterations takes about 24h of computation, so more sampling is not an option. In comparison, the proposed algorithm finishes in few seconds.

6.4 Real-World Data and Comparison with PGD

For a test with real-world data we used the CBCL face data set (CBCL, 2000). For different values of the sparsity constraints we derived NMF bases (Figure 1) and examined reconstruction error g and training time. In this experiment we used the tangent-plane constraints method and $s_w^{\min} = s_w^{\max} = s_h^{\min} = s_h^{\max}$. For comparison, we also employed the *projected gradient descent* (PGD) algorithm from Hoyer (2004) using the code provided on the author's homepage[3]. While the comparison in speed should be taken with a grain of salt—both methods use very different stopping criteria—the results (Tab. 3) show that the TPC method is competitive in speed and quality of its solutions.

6.5 Large-Scale Factorization of Image Data

To examine performance on a larger data set we sampled 10 000 image patches of size 11×11 from the Caltech-101 image database (Fei-Fei et al., 2004). Using a QP solver (Remark 5) and the TPC

3. To increase speed logging to file and screen were manually removed from the program.

ε	$r=2$	$r=4$	$r=6$	$r=8$	$r=10$
1.00	5.99	49.32	98.91	222.01	278.76
0.50	5.97	54.67	103.93	230.45	256.98
0.25	10.22	72.75	133.23	224.62	363.62

Table 4: **Large-scale performance.** A matrix containing $n = 10\,000$ image patches with $m = 121$ pixels was factorized using r basis functions and different stopping criteria for the TPC/QP algorithm (see text). The median CPU time (sec.) for three repeated runs is shown. Even the largest experiment with over 100 000 unknown variables is solved within 6 min.

algorithm we computed image bases with $r = 2,4,\ldots,10$ and $r = 50$ basis functions using $s_w^{\min} = 0.5$. In addition, we varied the stopping criterion from $\varepsilon \in \{1, 0.5, 0.25\}$. Note that the corresponding QP instances contained roughly 100 000 to over half a million unknowns, so a stopping criterion of $\varepsilon = 1$ translates to very small changes in the entries of W and H. We did *not* use any batch processing scheme but solved the QP instances directly, requiring between 100 MB and 2 GB of memory.

We show the median CPU time over three repeated runs for this experiment in Table 4: While the stopping criterion has only minor influence on the run time the number of basis functions is critical. All problems with up to 10 basis functions are solved within 6 min. For the large problem with 50 basis functions we measured a CPU time of 3, 5, and 7 hours for $\varepsilon \in \{1, 0.5, 0.25\}$. Memory consumption was roughly 2 GB. We conclude that factorization problems with half a million unknowns can be comfortably solved on current office equipment.

6.6 Supervised NMF

We examined how supervised NMF contributes to solve a classification task. Using overall 100 training samples we trained an $r = 4$ dimensional NMF basis for the digits 0, 3, 5, and 8 from the optdigits database. Subsequently, the remaining 1421 digits were classified using the nearest neighbor classifier. The penalty parameter λ in (7) was chosen as $\lambda = (\infty, 5, 2, 1, 0.75, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.01)$, where ∞ corresponds to classical NMF without class label information. The experiment was repeated 30 times, and the mean classification error is depicted in Figure 3. For comparison, a nearest-neighbor classification using a PCA basis of equal dimension generated 109 errors on the test data. It is evident that by strengthening the supervised label constraint we reduce the classification error significantly, increasing recognition accuracy by a factor of two.

7. Conclusion

We have shown that Euclidean NMF with and without sparsity constraints fits nicely within the framework of sequential quadratic and second order cone programming. For these problems, progress in numerical analysis has lead to highly efficient solvers which we exploit.

As a result, we propose efficient and robust algorithms for NMF which are competitive with or better than state-of-the-art alternatives. Besides performance and robustness, a key advantage of our approach is that incorporating prior knowledge in form of additional constraints will often be possible in a controlled and systematic way. For instance, information on class membership avail-

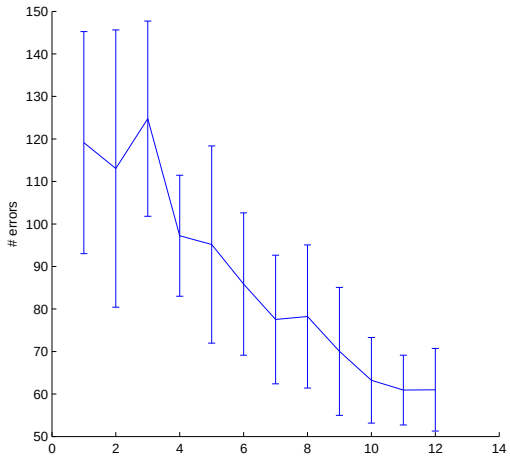


Figure 3: **Supervised NMF.** Reduction of classification error by supervised NMF. The letters 8, 5, 3, and 0 from the optdigits database are classified using a NMF basis of dimension $m = 4$ and overall 100 training and 1421 test samples. From left to right various values for the supervised label constraint, $\lambda = (\infty, 5, 2, 1, 0.75, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.01)$, were applied. Each experiment was repeated 30 times, mean performance and standard deviations of the nearest neighbor classificator are reported. As λ decreases the supervised label constraint is strengthened, reducing the classification error by a factor of two.

able in supervised classification settings leads to additional *convex* constraints that do not further complicate the optimization problem in a noteworthy way: no new algorithms need to be derived, no suitable, typically more stringent, learning rates need to be determined.

Acknowledgments

The authors thank P. O. Hoyer for making his NMF-pack software publicly available. We also thank Bo Jensen for answering any question about the MOSEK solver quickly and reliably. Last but not least we thank the anonymous reviewers for their valuable comments.

References

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, 1999.

I. Buciu and I. Pitas. Application of non-negative and local non negative matrix factorization to facial expression recognition. In *Proc. of Intl. Conf. Pattern Recogn. (ICPR)*, pages 288–291, 2004.

CBCL. CBCL face database #1. MIT Center For Biological and Computational Learning, http://cbcl.mit.edu/software-datasets, 2000.

A. Chichocki, R. Zdunek, and S. Amari. Csiszár's divergences for non-negative matrix factorization: Family of new algorithms. In *6th Intl. Conf. on ICA and Blind Sign. Separ.*, March 2006.

D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Adv. in Neur. Inform. Proc. Syst. (NIPS)*, volume 17, 2004.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Comp. Vis. and Pattern Recogn. (CVPR) 2004, Workshop on Generative-Model Based Vision*, 2004.

C. A. Floudas and V. Visweswaran. A primal-relaxed dual global optimization approach. *J. of Optim. Theory and Applic.*, 78(2), 1993.

A. Graham. *Kronecker Products and Matrix Calculus With Applications*. Ellis Horwood Lim., Chichester, 1981.

M. Heiler and C. Schnörr. Reverse-convex programming for sparse image codes. In *Proc. of Energy Minim. Methods in Comp. Vision and Pattern Recog. (EMMCVPR)*, volume 3757 of *LNCS*, pages 600–616. Springer, 2005.

J. B. Hiriart-Urruty. Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. In J. Ponstein, editor, *Lect. Not. Econom. Math. Systems*, volume 256, pages 37–70. Springer, 1985.

R. Horst and H. Tuy. *Global Optimization*. Springer, Berlin, 1996.

P. O. Hoyer. Non-negative sparse coding. *Proc. of the IEEE Works. on Neur. Netw. for Sig. Proc.*, pages 557–565, 2002.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. of Mach. Learning Res.*, 5:1457–1469, 2004.

P. O. Hoyer and A. Hyvärinen. A multi-layer sparse coding network learns contour coding from natural images. *Vision Research*, 42(12):1593–1605, 2002.

Cplex 2001. *ILOG CPLEX 7.5 User's Manual*. ILOG Inc., France, 2001.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Adv. in Neur. Inform. Proc. Syst. (NIPS)*, 2000.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(21):788–791, October 1999.

S. Z. Li, X. W. Hou, H. J. Zhang, and Q. S. Cheng. Learning spatially localized, parts-based representation. In *Proc. of Comp. Vision and Pattern Recog. (CVPR)*, 2001.

M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

M. J. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with Gabor wavelets. In *Third IEEE Intl. Conf. on Auto. Face and Gesture Recog.*, pages 200–205, 1998.

H. D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Math. Programming, Series B*, 95(2):407–430, 2003.

Mosek 2005. *The MOSEK optimization tools version 3.2 (Revision 8) User's manual and reference*. MOSEK ApS, Denmark, 2005.

P. Paatero. Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37, 1997.

P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*, volume 317 of *Grundlehren der math. Wissenschaften*. Springer, 1998.

J. Shen and G. W. Israël. A receptor model using a specific non-negative transformation technique for ambient aerosol. *Atmospheric Environment*, 23(10):2289–2298, 1989.

I. Singer. Minimization of continuous convex functionals on complements of convex subsets of locally convex spaces. *Math. Oper. Statist. Ser. Optim.*, 11:221–234, 1980.

P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *IEEE Workshop on Appl. of Sign. Proc. to Audio and Acoustics*, pages 177–180, 2003.

J. F. Sturm. *Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones (updated version 1.05)*. Department of Econometrics, Tilburg University, Tilburg, The Netherlands, 2001.

J. F. Toland. A duality principle for nonconvex optimization and the calculus of variation. *Arch. Rat. Mech. Anal.*, 71:41–61, 1979.

H. Tuy. D.c. optimization: Theory, methods and algorithms. In R. Horst and M. Pardalos, editors, *Handbook of global optimization*, pages 149–216. Kluwer, Dordrecht-Boston-London, 1995.

H. Tuy. Convex programs with an additional reverse convex constraint. *J. of Optim. Theory and Applic.*, 52(3):463–486, March 1987.

Y. Wang, Y. Jia, C. Hu, and M. Turk. Fisher non-negative matrix factorization for learning local features. In *Proc. Asian Conf. on Comp. Vision*, 2004.

S. J. Wright. *Primal–Dual Interior–Point Methods*. SIAM, 1996.

W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR '03: Proc. of the 26th Ann. Intl. ACM SIGIR Conf. on Res. and Developm. in Info. Retrieval*, pages 267–273. ACM Press, 2003. ISBN 1-58113-646-3. doi: http://doi.acm.org/10.1145/860435.860485.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.

Journal of Machine Learning Research 7 (2006) 1409–1436 Submitted 10/05; Revised 3/06; Published 7/06

Fast SDP Relaxations of Graph Cut Clustering, Transduction,and Other Combinatorial Problems

Tijl De Bie TIJL.DEBIE@GMAIL.COM
Southampton University, ECS, ISIS
SO17 1BJ, United Kingdom
and
K. U. Leuven, OKP Research Group
Tiensestraat 102
3000 Leuven, Belgium

Nello Cristianini NELLO@SUPPORT-VECTOR.NET
University of Bristol
Department of Engineering Mathematics and Department of Computer Science
Queen's Building, University Walk
Bristol, BS8 1TR, United Kingdom

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

The rise of convex programming has changed the face of many research fields in recent years, machine learning being one of the ones that benefitted the most. A very recent developement, the relaxation of combinatorial problems to semi-definite programs (SDP), has gained considerable attention over the last decade (Helmberg, 2000; De Bie and Cristianini, 2004a). Although SDP problems can be solved in polynomial time, for many relaxations the exponent in the polynomial complexity bounds is too high for scaling to large problem sizes. This has hampered their uptake as a powerful new tool in machine learning.

In this paper, we present a new and fast SDP relaxation of the normalized graph cut problem, and investigate its usefulness in unsupervised and semi-supervised learning. In particular, this provides a convex algorithm for transduction, as well as approaches to clustering. We further propose a whole cascade of fast relaxations that all hold the middle between older spectral relaxations and the new SDP relaxation, allowing one to trade off computational cost versus relaxation accuracy. Finally, we discuss how the methodology developed in this paper can be applied to other combinatorial problems in machine learning, and we treat the max-cut problem as an example.

Keywords: convex transduction, normalized graph cut, semi-definite programming, semi-supervised learning, relaxation, combinatorial optimization, max-cut

1. Introduction

Let us assume a data sample $\mathcal{S}$ containing n points is given. Between every pair of samples $(\mathbf{x}_i, \mathbf{x}_j)$, an *affinity measure* $\mathbf{A}(i,j) = a(\mathbf{x}_i, \mathbf{x}_j)$ is defined, making up an *affinity matrix* $\mathbf{A}$. We assume the function a is symmetric and positive, however, no positive definiteness of $\mathbf{A}$ will be necessary, probably making the application domain larger than that of kernel based methods as discussed in e.g. Chapelle et al. (2003); De Bie and Cristianini (2004a).

Graph cut clustering Informally speaking, in this paper we are seeking to divide these data points into two coherent sets, denoted by $\mathcal{P}$ and $\mathcal{N}$, such that $\mathcal{P} \bigcup \mathcal{N} = \mathcal{S}$ and $\mathcal{P} \bigcap \mathcal{N} = \emptyset$. In the fully unsupervised-learning scenario, no prior information is given as to which class the points belong to. A number of approaches to bipartitioning sets of data, known as graph cut clustering approaches, make use of an edge-weighted graph, where the nodes in the graph represent the data points and the edges between them are weighted with the affinities between the data points. Bipartitioning the data set then corresponds to cutting the graph in two parts. Intuitively, the fewer high affinity edges are cut, the better the division into two coherent and mutually different parts will be. In Section 1.1 we recall a few graph cut cost functions that have been proposed in literature.

Graph cut transduction Besides this clustering scenario, we also consider the transduction scenario, where part of the class labels is specified. Transduction has received much attention in the past years as a promising middle ground between supervised and unsupervised learning, but major computational obstacles have so far prevented it from becoming a standard piece in the toolbox of practitioners, despite the fact that many natural learning situations directly translate into a transduction problem. In graph cut approaches, the problem of transduction can naturally be approached by restricting the search for a low cost graph cut to graph cuts that do not violate the label information.

Even more generally, one can consider the case where labels are not exactly specified, but where equivalence or inequivalence constraints (Shental et al., 2004) are given instead, specifying equality or non-equality of the labels respectively.

1.1 Cut, Average Cut and Normalized Cut Cost Functions

Several graph cut cost functions have been proposed in literature in the context of clustering, among which the *cut cost*, the *average cut cost (ACut)* and the *normalized cut cost (NCut)* (Shi and Malik, 2000).

The Cut cost is computationally the easiest to handle in a transduction setting (see Blum and Chawla, 2001), however as clearly motivated in Joachims (2003), it often leads to degenerate results with one of both clusters extremely small. This problem could largely be solved by using the ACut or NCut cost functions, of which the ACut cost seems to be more vulnerable to outliers (atypical data points, meaning that they have low affinity to the rest of the sample). However, both optimizing the ACut and NCut costs are NP-complete problems (Shi and Malik, 2000).

To get around this, *spectral* relaxations of the ACut and NCut optimization problems have been proposed in a clustering (Shi and Malik, 2000; Ng et al., 2002; Cristianini et al., 2002) and more recently also in a transduction setting (Kamvar et al., 2003; Joachims, 2003; De Bie et al., 2004). Xing and Jordan (2003) also proposed an interesting SDP relaxation for the NCut optimization problem in a multiclass *clustering* setting, however, the computational cost to solve this relaxation turns out to be too high to cluster data sets of more than about 150 data points, which makes it impractical in real situations.

1.2 Paper Outline

We should emphasize that we are not as much interested in making claims concerning the usefulness of the normalized graph cut for (constrained) clustering problems. A statistical study of the NCut cost function is still lacking, such that claims are necessarily data-dependent, and hence conflicting opinions exist. Instead, we mainly focus on the algorithmic problem involved in the optimization of the normalized graph cut as an interesting object of study on itself, because of its direct applicability to machine learning algorithms design. Furthermore, we will show how the methodologies presented in the context of the NCut optimization problem have a wider applicability, and can be of use to approximately solve other combinatorial problems as well. Our results are structured as follows.

- In Section 2 we recapitulate the well known spectral relaxation of the NCut problem to an eigenvalue problem. Subsequently, a first main result of this paper is presented, which is an efficiently solvable SDP relaxation of the NCut optimization problem. Lastly, this section contains a methodology to construct a cascade of SDP relaxations, all tighter than the spectral relaxation and looser than the SDP relaxation, and with a computational cost in between the cost of both extremes.
- In Section 3 we introduce the so-called *subspace trick*, and show two of its applications. In Section 3.1 we observe how it enables one to efficiently impose equivalence and inequivalence constraints between the labels on the solution of the relaxations. Hence, also transduction problems with the NCut cost can be tackled efficiently. Section 3.2 contains a second application of the subspace trick, consisting of a further speed-up of the relaxations derived in Section 2.
- Lastly, in Section 4 we illustrate how the relaxation cascade and the subspace trick can be applied to speed up relaxations of other combinatorial problems as well, by applying it to the max-cut problem.

We conclude with empirical results for the normalized cut and for the max-cut problems.

2. Relaxations of the Normalized Graph Cut Problem

The NCut cost function for a partitioning of the sample $\mathcal{S}$ into a positive $\mathcal{P}$ and a negative $\mathcal{N}$ set is given by (as originally denoted in Shi and Malik (2000)):

$$\frac{\text{cut}(\mathcal{P},\mathcal{N})}{\text{assoc}(\mathcal{P},\mathcal{S})} + \frac{\text{cut}(\mathcal{N},\mathcal{P})}{\text{assoc}(\mathcal{N},\mathcal{S})} = \left(\frac{1}{\text{assoc}(\mathcal{P},\mathcal{S})} + \frac{1}{\text{assoc}(\mathcal{N},\mathcal{S})}\right) \cdot \text{cut}(\mathcal{P},\mathcal{N}), \tag{1}$$

where $\text{cut}(\mathcal{P},\mathcal{N}) = \text{cut}(\mathcal{N},\mathcal{P}) = \sum_{i:\mathbf{x}_i\in\mathcal{P},j:\mathbf{x}_j\in\mathcal{N}} \mathbf{A}(i,j)$ is the cut between sets $\mathcal{P}$ and $\mathcal{N}$, and $\text{assoc}(\mathcal{P},\mathcal{S}) = \sum_{i:\mathbf{x}_i\in\mathcal{P},j:\mathbf{x}_j\in\mathcal{S}} \mathbf{A}(i,j)$ the association between sets $\mathcal{P}$ and the full sample $\mathcal{S}$. (Note that in fact $\text{cut}(\mathcal{P},\mathcal{N}) = \text{assoc}(\mathcal{P},\mathcal{N})$.) Intuitively, it is clear that the second factor $\text{cut}(\mathcal{P},\mathcal{N})$ defines how well the two clusters separate. The first factor $\left(\frac{1}{\text{assoc}(\mathcal{P},\mathcal{S})} + \frac{1}{\text{assoc}(\mathcal{N},\mathcal{S})}\right)$ measures how well the clusters are balanced. This specific measure of imbalancedness can be seen to improve robustness against atypical data points:[1]

1. This property seems even more important in the relaxations of NCut based methods: the variables then have even more freedom, often making the methods more vulnerable to outliers.

such outliers have a small cut cost with the other data points, making it beneficial to separate them out into a cluster of their own, which would lead to a useless result in our 2-class setting. However, they also have a small association with the rest of the sample $\mathcal{S}$, which on the other hand increases the cost function. In other words, the NCut cost function promotes partitions that are balanced in the sense that both clusters are roughly equally 'coherent', while at the same time 'distant' from each other. It is this feature that makes it preferable over the ACut cost function.[2]

To optimize this cost function, we reformulate it into algebraic terms using the unknown label vector $\mathbf{y} \in \{-1,1\}^n$, the affinity matrix $\mathbf{A}$, the degree vector $\mathbf{d} = \mathbf{A1}$ and associated matrix $\mathbf{D} = \text{diag}(\mathbf{d})$, and shorthand notations $s_+ = \text{assoc}(\mathcal{P}, \mathcal{S})$ and $s_- = \text{assoc}(\mathcal{N}, \mathcal{S})$.

Observe that $\text{cut}(\mathcal{P}, \mathcal{N}) = \frac{(\mathbf{1}+\mathbf{y})'}{2}\mathbf{A}\frac{(\mathbf{1}-\mathbf{y})'}{2} = \frac{1}{4}\left(-\mathbf{y}'\mathbf{A}\mathbf{y} + \mathbf{1}'\mathbf{A}\mathbf{1}\right) = \frac{1}{4}\mathbf{y}'(\mathbf{D}-\mathbf{A})\mathbf{y}$. Furthermore, $s_+ = \text{assoc}(\mathcal{P}, \mathcal{S}) = \frac{1}{2}\mathbf{1}'\mathbf{A}(\mathbf{1}+\mathbf{y}) = \frac{1}{2}\mathbf{d}'(\mathbf{1}+\mathbf{y})$ and $s_- = \frac{1}{2}\mathbf{d}'(\mathbf{1}-\mathbf{y})$. Then we can write the combinatorial optimization problem as:

$$\begin{aligned} \min_{\mathbf{y}, s_+, s_-} \quad & \frac{1}{4}\left(\frac{1}{s_+} + \frac{1}{s_-}\right) \cdot \mathbf{y}'(\mathbf{D}-\mathbf{A})\mathbf{y} \\ \text{s.t.} \quad & \mathbf{y} \in \{-1,1\}^n, \\ & \left\{ \begin{array}{l} s_+ = \frac{1}{2}\mathbf{d}'(\mathbf{1}+\mathbf{y}) \\ s_- = \frac{1}{2}\mathbf{d}'(\mathbf{1}-\mathbf{y}) \end{array} \right. \Leftrightarrow \left\{ \begin{array}{c} \mathbf{d}'\mathbf{y} = s_+ - s_- \\ \mathbf{d}'\mathbf{1} = s_+ + s_- = s, \end{array} \right. \end{aligned}$$

where we introduced the additional symbol s for the constant $s = s_+ + s_- = \mathbf{d}'\mathbf{1} = \mathbf{1}'\mathbf{D}\mathbf{1}$. In this new notation, the optimization problem becomes:

$$\begin{aligned} \min_{\mathbf{y}, s_+, s_-} \quad & \frac{s}{4 s_+ s_-} \cdot \mathbf{y}'(\mathbf{D}-\mathbf{A})\mathbf{y} \\ \text{s.t.} \quad & \mathbf{y} \in \{-1,1\}^n, \\ & \mathbf{d}'\mathbf{y} = s_+ - s_-, \\ & s_+ + s_- = s. \end{aligned} \tag{2}$$

Unfortunately, the resulting optimization problem is known to be NP-complete. Therefore, we approach the problem by relaxing it to more tractable optimization problems. This is the subject of what follows below.

As a guide for the reader, the main notation is summarized in Table 1. We would like to note that we suppress matrix symmetricity constraints where these can be understood from the context.

2.1 A Spectral Relaxation

We now provide a short derivation of the *spectral* relaxation of the NCut optimization problem as first given in Shi and Malik (2000). Let us introduce the variable $\widetilde{\mathbf{y}}$ defined as:

$$\begin{aligned} \widetilde{\mathbf{y}} &= \sqrt{\frac{s}{4 s_+ s_-}}\left(\mathbf{y} - \mathbf{1}\frac{s_+ - s_-}{s}\right) \\ &= \sqrt{\frac{s}{4 s_+ s_-}}\left(\mathbf{I} - \frac{\mathbf{1}\mathbf{d}'}{s}\right)\mathbf{y}, \end{aligned}$$

2. Note however that when a k-nearest neighbor affinity matrix is used, as in Kamvar et al. (2003), every sample has the same affinity with the remainder of the data set, such that the ACut and the NCut costs become equivalent.

Symbol		Definition		Useful identities
$\mathbf{A}$	$=$	affinity matrix $\in \Re_+^{n\times n}$		
$\mathbf{1}$	$=$	$\{1\}^n$		
$\mathbf{I}$	$=$	$\mathrm{diag}(\mathbf{1})$		
$\mathbf{d}$	$=$	$\mathbf{A1}$		
$\mathbf{D}$	$=$	$\mathrm{diag}(\mathbf{d})$		
s_+	$=$	$\mathrm{assoc}(\mathcal{P},\mathcal{S})$	$=$	$\sum_{i:y_i=1} d_i = \mathbf{d}'\frac{\mathbf{1}+\mathbf{y}}{2} = \mathbf{1}'\mathbf{A}\frac{\mathbf{1}+\mathbf{y}}{2}$
s_-	$=$	$\mathrm{assoc}(\mathcal{N},\mathcal{S})$	$=$	$\sum_{i:y_i=-1} d_i = \mathbf{d}'\frac{\mathbf{1}-\mathbf{y}}{2} = \mathbf{1}'\mathbf{A}\frac{\mathbf{1}-\mathbf{y}}{2}$
s	$=$	$\mathrm{assoc}(\mathcal{S},\mathcal{S})$	$=$	$s_+ + s_- = \mathbf{1}'\mathbf{d} = \mathbf{1}'\mathbf{D1} = \mathbf{1}'\mathbf{A1}$
$\mathbf{y}$	$\in$	$\{-1,1\}^n$		
$\widetilde{\mathbf{y}}$	$=$	$\sqrt{\frac{s}{4s_+s_-}}\left(\mathbf{y} - \mathbf{1}\frac{s_+-s_-}{s}\right)$	$=$	$\sqrt{\frac{s}{4s_+s_-}}\left(\mathbf{I} - \frac{\mathbf{1d}'}{s}\right)\mathbf{y}$
p	$=$	$\frac{4s_+s_-}{s^2}$		
q	$=$	$\frac{1}{p}$		
$\mathbf{\Gamma}$	$=$	$\mathbf{y}\mathbf{y}'$		
$\widehat{\mathbf{\Gamma}}$	$=$	$\frac{1}{p}\mathbf{\Gamma}$	$=$	$q\mathbf{\Gamma}$
$\widetilde{\mathbf{\Gamma}}$	$=$	$\frac{s}{4s_+s_-}\left(\mathbf{I} - \frac{\mathbf{1d}'}{s}\right)\cdot\mathbf{\Gamma}\cdot\left(\mathbf{I} - \frac{\mathbf{1d}'}{s}\right)'$	$=$	$\widetilde{\mathbf{y}}\widetilde{\mathbf{y}}'$
$\mathbf{W}$	$\in$	$\Re^{n\times m}$		
$\mathbf{V}$	$=$	eigenvectors of spectral relaxation		
$\widehat{\mathbf{\Gamma}}$	$=$	$\mathbf{V}\mathbf{M}\mathbf{V}'$		
$\mathbf{R}$	$=$	subspace constraint matrix $\in \Re^{n\times d}$		
$\widehat{\mathbf{\Gamma}}$	$=$	$\mathbf{R}\mathbf{M}\mathbf{R}'$		
		(subspace-constrained label matrix)		
$\mathbf{L}$	$=$	label constraint matrix		
$\widehat{\mathbf{\Gamma}}$	$=$	$\mathbf{L}\mathbf{M}\mathbf{L}'$		
		(label-constrained label matrix)		

Table 1: Notation summary. Note that some equalities should be replaced by approximate equalities depending on the context. Throughout the paper, label matrices are understood to be symmetric and any symmetricity constraints are suppressed for conciseness.

and rewrite the optimization problem in terms of this variable by accordingly substituting $\mathbf{y} = \sqrt{\frac{4s_+s_-}{s}}\widetilde{\mathbf{y}} + \mathbf{1}\frac{s_+-s_-}{s}$:

$$\begin{array}{rl} \min_{\widetilde{\mathbf{y}},s_+,s_-} & \widetilde{\mathbf{y}}'(\mathbf{D}-\mathbf{A})\widetilde{\mathbf{y}} \\ \text{s.t.} & \widetilde{\mathbf{y}} \in \left\{-\sqrt{\frac{s_+}{ss_-}}, \sqrt{\frac{s_-}{ss_+}}\right\}^n, \\ & \mathbf{d}'\widetilde{\mathbf{y}} = 0, \\ & s_+ + s_- = s. \end{array} \tag{3}$$

Proposition 1 *The constraints of optimization problem (3) imply that the* $\mathbf{D}$*-weighted 2-norm of* $\widetilde{\mathbf{y}}$ *is constant and equal to* $\widetilde{\mathbf{y}}'\mathbf{D}\widetilde{\mathbf{y}} = 1$.

Proof

$$\begin{aligned} \widetilde{\mathbf{y}}'\mathbf{D}\widetilde{\mathbf{y}} &= \frac{s}{4s_+s_-}\mathbf{y}'\left(\mathbf{I} - \frac{\mathbf{d}\mathbf{1}'}{s}\right)\mathbf{D}\left(\mathbf{I} - \frac{\mathbf{1}\mathbf{d}'}{s}\right)\mathbf{y} \\ &= \frac{s}{4s_+s_-}\mathbf{y}'\left(\mathbf{D} - \frac{\mathbf{d}\mathbf{d}'}{s}\right)\mathbf{y} \\ &= \frac{s}{4s_+s_-}\left(s - \frac{(s_+-s_-)^2}{s}\right) \\ &= 1. \end{aligned}$$

■

Hence we can add the (redundant) constraint $\widetilde{\mathbf{y}}'\mathbf{D}\widetilde{\mathbf{y}} = 1$ to the optimization problem without altering the result. The spectral relaxation is obtained by doing so, and subsequently dropping the combinatorial constraint on $\widetilde{\mathbf{y}}$. The result is:

$$\textbf{Spectral}\left\{\begin{array}{rl} \min_{\widetilde{\mathbf{y}}} & \widetilde{\mathbf{y}}'(\mathbf{D}-\mathbf{A})\widetilde{\mathbf{y}} \\ \text{s.t.} & \widetilde{\mathbf{y}}'\mathbf{D}\widetilde{\mathbf{y}} = 1, \\ & \mathbf{d}'\widetilde{\mathbf{y}} = 0, \end{array}\right. \tag{4}$$

which is solved by taking the (generalized) eigenvector $\widetilde{\mathbf{y}}$ corresponding to the second smallest generalized eigenvalue σ_2 of the generalized eigenvalue problem $(\mathbf{D}-\mathbf{A})\mathbf{v} = \sigma\mathbf{D}\mathbf{v}$. Note that the smallest generalized eigenvalue is $\sigma_1 = 0$, corresponding to the eigenvector $\frac{1}{\sqrt{s}}\mathbf{1}$.

2.2 An SDP Relaxation

We start from formulation (2), and introduce the notation $\mathbf{\Gamma} = \mathbf{y}\mathbf{y}'$. Then, we can write the equivalent optimization problem:

$$\begin{array}{rl} \min_{\mathbf{\Gamma},s_+,s_-} & \frac{s}{4s_+s_-}\langle\mathbf{\Gamma}, \mathbf{D}-\mathbf{A}\rangle \\ \text{s.t.} & \mathbf{\Gamma} = \mathbf{y}\mathbf{y}', \\ & \mathbf{y} \in \{-1,1\}^n, \\ & \langle\mathbf{\Gamma}, \mathbf{d}\mathbf{d}'\rangle = (s_+-s_-)^2 = (s_++s_-)^2 - 4s_+s_-, \\ & s_+ + s_- = s, \quad s_+ > 0, \quad s_- > 0. \end{array} \tag{5}$$

Note that these constraints imply that $(\mathbf{\Gamma}' =)\mathbf{\Gamma} \succeq \mathbf{0}$ and $\text{diag}(\mathbf{\Gamma}) = \mathbf{1}$ (where $\mathbf{A} \succeq \mathbf{B}$ means that $\mathbf{A}-\mathbf{B}$ is positive semi-definite). Hence we can relax the constraint set by adding these

two redundant constraints (we suppress the symmetricity constraint on $\mathbf{\Gamma}$ from the notation for conciseness), and dropping $\mathbf{\Gamma} = \mathbf{y}\mathbf{y}'$ and $\mathbf{y} \in \{-1, 1\}^n$. (While this is a tight relaxation, tighter relaxations are possible at higher computational cost, see Helmberg (2000).) If we further use the notation $p = \frac{4s_+s_-}{s^2}$, we get:

$$\begin{array}{rl} \min_{\mathbf{\Gamma},p} & \frac{1}{p}\langle \mathbf{\Gamma}, \frac{\mathbf{D}-\mathbf{A}}{s}\rangle \\ \text{s.t.} & \mathbf{\Gamma} \succeq \mathbf{0}, \\ & \operatorname{diag}(\mathbf{\Gamma}) = \mathbf{1}, \\ & \frac{1}{s^2}\langle \mathbf{\Gamma}, \mathbf{d}\mathbf{d}'\rangle = 1 - p, \\ & 0 < p \leq 1. \end{array} \tag{6}$$

By once again reparameterizing with $\widehat{\mathbf{\Gamma}} = \frac{\mathbf{\Gamma}}{p}$ and $q = 1/p$, we obtain:

$$\begin{array}{rl} \min_{\widehat{\mathbf{\Gamma}},q} & \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{D}-\mathbf{A}}{s}\rangle \\ \text{s.t.} & \widehat{\mathbf{\Gamma}} \succeq \mathbf{0}, \\ & \operatorname{diag}(\widehat{\mathbf{\Gamma}}) = q\mathbf{1}, \\ & \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{d}\mathbf{d}'}{s^2}\rangle = q - 1, \\ & q \geq 1. \end{array}$$

Note that $\langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{d}\mathbf{d}'}{s^2}\rangle \geq 0$ such that the constraint $\langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{d}\mathbf{d}'}{s^2}\rangle = q - 1$ implies the inequality constraint $q \geq 1$. Hence it does not need to be mentioned explicitly. The result is an optimization problem with a linear objective, $n + 1$ linear equality constraints, and a PSD constraint on a matrix of size n that is linear in the parameters. Hence, we have reshaped the relaxed problem into a standard SDP formulation.

The Lagrange dual We will now derive the dual of this optimization problem, as it will be helpful in the theoretical understanding of the optimization problem as well as for its implementation. To this end we use a symmetric matrix $\mathbf{\Xi} \in \Re^{n\times n}$, a vector $\boldsymbol{\lambda} \in \Re^n$ and a scalar μ as *Lagrange multipliers* (also called *dual variables* in the sequel). Then we can write the Lagrangian as:

$$\begin{aligned} \mathcal{L}(\widehat{\mathbf{\Gamma}}, q, \mathbf{\Xi}, \boldsymbol{\lambda}, \mu) &= \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{D}-\mathbf{A}}{s}\rangle - \langle \widehat{\mathbf{\Gamma}}, \mathbf{\Xi}\rangle - \boldsymbol{\lambda}'\left(\operatorname{diag}(\widehat{\mathbf{\Gamma}}) - q\mathbf{1}\right) - \mu\left((q-1) - \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{d}\mathbf{d}'}{s^2}\rangle\right) \\ &= \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{D}-\mathbf{A}}{s} - \mathbf{\Xi} - \operatorname{diag}(\boldsymbol{\lambda}) + \mu\frac{\mathbf{d}\mathbf{d}'}{s^2}\rangle + q(\mathbf{1}'\boldsymbol{\lambda} - \mu) + \mu, \end{aligned}$$

and the primal optimization problem is equivalent with:

$$\operatorname{opt}_{\text{primal}} = \min_{\widehat{\mathbf{\Gamma}},q}\left[\max_{\mathbf{\Xi}\succeq\mathbf{0},\boldsymbol{\lambda},\mu} \mathcal{L}(\widehat{\mathbf{\Gamma}}, q, \mathbf{\Xi}, \boldsymbol{\lambda}, \mu)\right].$$

Indeed, either the primal constraints are fulfilled and then the inner maximization reduces to the primal objective, or the maximum over the dual constraints is unbounded:

$$\max_{\mathbf{\Xi}\succeq\mathbf{0},\boldsymbol{\lambda},\mu} \mathcal{L}(\widehat{\mathbf{\Gamma}}, q, \mathbf{\Xi}, \boldsymbol{\lambda}, \mu) = \begin{cases} \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{D}-\mathbf{A}}{s}\rangle & \text{if the primal constraints are fulfilled, and} \\ \infty & \text{otherwise.} \end{cases}$$

Thus, in order to minimize this maximum, the primal variables $\widehat{\mathbf{\Gamma}}$ and q will be such that the constraints are met.

The so-called *dual optimization problem* is obtained by interchanging the maximization and minimization in this optimization problem:

$$\text{opt}_{\text{dual}} = \max_{\mathbf{\Xi}\succeq\mathbf{0},\boldsymbol{\lambda},\mu} \left[\min_{\widehat{\mathbf{\Gamma}},q} \mathcal{L}(\widehat{\mathbf{\Gamma}}, q, \mathbf{\Xi}, \boldsymbol{\lambda}, \mu) \right].$$

A very useful relation between the primal and dual optima (on its own already warranting the study of the dual problem) is known as *weak duality*, and says that the dual maximum is a lower bound for the primal minimum (see e.g. Boyd and Vandenberghe (2004)). I.e.:

$$\text{opt}_{\text{primal}} \geq \text{opt}_{\text{dual}}.$$

Let us further focus on the dual optimization problem. The inner minimization can be explicitly solved. Indeed, it is easy to see that it is equal to μ if the following conditions hold:

$$\begin{aligned} \frac{\mathbf{D}-\mathbf{A}}{s} - \mathbf{\Xi} - \text{diag}(\boldsymbol{\lambda}) + \mu\frac{\mathbf{d}\mathbf{d}'}{s^2} &= \mathbf{0} \\ \mathbf{1}'\boldsymbol{\lambda} - \mu &= 0, \end{aligned}$$

and unbounded from below otherwise. Following a similar reasoning as above, this implies that these equalities will hold at the optimum. Hence, we obtain as a dual optimization problem:

$$\begin{array}{ll} \max_{\mathbf{\Xi},\boldsymbol{\lambda},\mu} & \mu, \\ \text{s.t.} & \mathbf{\Xi} \succeq \mathbf{0}, \\ & \mathbf{\Xi} = \frac{\mathbf{D}-\mathbf{A}}{s} - \text{diag}(\boldsymbol{\lambda}) + \mu\frac{\mathbf{d}\mathbf{d}'}{s^2}, \\ & \mathbf{1}'\boldsymbol{\lambda} = \mu. \end{array}$$

The matrix $\mathbf{\Xi}$ is easily eliminated from these constraints, which gives us the final formulation. We state both the primal and the dual:

$$\mathbf{P}^{\mathbf{clust}}_{\mathbf{SDP}} \left\{ \begin{array}{ll} \min_{\widehat{\mathbf{\Gamma}},q} & \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{D}-\mathbf{A}}{s} \rangle \\ \text{s.t.} & \widehat{\mathbf{\Gamma}} \succeq \mathbf{0}, \\ & \text{diag}(\widehat{\mathbf{\Gamma}}) = q\mathbf{1}, \\ & \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{d}\mathbf{d}'}{s^2} \rangle = q - 1. \end{array} \right. \qquad \mathbf{D}^{\mathbf{clust}}_{\mathbf{SDP}} \left\{ \begin{array}{ll} \max_{\boldsymbol{\lambda},\mu} & \mu, \\ \text{s.t.} & \frac{\mathbf{D}-\mathbf{A}}{s} - \text{diag}(\boldsymbol{\lambda}) + \mu\frac{\mathbf{d}\mathbf{d}'}{s^2} \succeq \mathbf{0}, \\ & \mathbf{1}'\boldsymbol{\lambda} = \mu. \end{array} \right.$$

Importantly, this relaxation contains only $n+1$ dual variables. It is thanks to this feature that this relaxation leads to a much more efficient algorithm than the one presented in Xing and Jordan (2003). But we postpone a detailed computational study until Section 2.4.

2.3 A Cascade of Relaxations Tighter Than Spectral and Looser Than SDP

Still, in many cases the SDP relaxation is too complex, while the spectral relaxation is computationally feasible but too loose. Whereas numerous efforts have been made in literature to further tighten SDP relaxations of (other) combinatorial problems by adding in

additional constraints and using so-called lifting techniques (see e.g. Anjos and Wolkowicz (2002)), contributions to further relax the SDP problem without considerably degrading the solution and while gaining on the computational side, have remained limited. Here we present a set of such relaxations, and we will show that they hold the middle between the SDP relaxation and the spectral relaxation, both in terms of computational complexity and in terms of accuracy.

The basic observation to be made is the fact that the constraint $\text{diag}(\widehat{\mathbf{\Gamma}}) = q\mathbf{1}$ implies:

$$\mathbf{W}'\text{diag}(\widehat{\mathbf{\Gamma}}) = q\mathbf{W}'\mathbf{1},$$

for $\mathbf{W} \in \Re^{n\times m}$ (which we choose to be of full column rank, so with $1 \leq m \leq n$). Hence, we can relax the constraint $\text{diag}(\widehat{\mathbf{\Gamma}}) = q\mathbf{1}$ to this weaker constraint. The resulting primal and dual optimization problems are:

$$\mathbf{P}^{\mathbf{clust}}_{\mathbf{m\text{-}SDP}} \begin{cases} \min_{\widehat{\mathbf{\Gamma}},q} & \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{D}-\mathbf{A}}{s} \rangle \\ \text{s.t.} & \widehat{\mathbf{\Gamma}} \succeq \mathbf{0}, \\ & \mathbf{W}'\text{diag}(\widehat{\mathbf{\Gamma}}) = q\mathbf{W}'\mathbf{1}, \\ & \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{d}\mathbf{d}'}{s^2} \rangle = q-1. \end{cases} \qquad \mathbf{D}^{\mathbf{clust}}_{\mathbf{m\text{-}SDP}} \begin{cases} \max_{\boldsymbol{\lambda},\mu} & \mu, \\ \text{s.t.} & \frac{\mathbf{D}-\mathbf{A}}{s} - \text{diag}(\mathbf{W}\boldsymbol{\lambda}) \\ & \quad +\mu\frac{\mathbf{d}\mathbf{d}'}{s^2} \succeq \mathbf{0}, \\ & \mu = \mathbf{1}'\mathbf{W}\boldsymbol{\lambda}. \end{cases}$$

The attractive feature of the relaxation cascade is the fact that the number of dual parameters is only $m+1$, as opposed to $n+1$ for the basic SDP relaxation. Hence, for smaller m, the optimization can be carried out more efficiently.

In general, it is clear that a relaxation is tighter than another if the column space of the matrix $\mathbf{W}$ used in the first one contains the full column space of $\mathbf{W}$ of the second. In particular, for $d = n$ the original SDP relaxation is obtained. At the other extreme, for $m = 1$, let us take $\mathbf{W} = \mathbf{d}$. Then essentially the spectral relaxation is obtained.

Theorem 2 *The SDP relaxation from the cascade with $m = 1$ and $\mathbf{W} = \mathbf{d}$ is (essentially) equivalent to the spectral relaxation.*

Proof Let us write $\widehat{\mathbf{\Gamma}} = \mathbf{V}\mathbf{M}\mathbf{V}'$ with $\mathbf{M} \in \Re^{n\times n}$ a symmetric matrix and with the eigenvectors $\mathbf{v}$ of the spectral relaxation $(\mathbf{D}-\mathbf{A})\mathbf{v} = \sigma\mathbf{D}\mathbf{v}$ as the columns of $\mathbf{V}$, in order of increasing eigenvalue σ, and normalized such that $\mathbf{V}'\mathbf{D}\mathbf{V} = \mathbf{I}$. I.e., the first column of $\mathbf{V}(:,1) = \frac{\mathbf{1}}{\sqrt{s}}$, and the second column $\mathbf{V}(:,2) = \widetilde{\mathbf{y}}$ is the relaxed label vector obtained using the spectral relaxation. Then we have that $\mathbf{V}'(\mathbf{D}-\mathbf{A})\mathbf{V} = \mathbf{\Sigma}$, a diagonal matrix with the generalized eigenvalues in ascending order on the diagonal, i.e. $\mathbf{\Sigma}(1,1) = \sigma_1 = 0$ and $\mathbf{\Sigma}(2,2) = \sigma_2$. Using this reparameterization we can rewrite $\mathbf{P}^{\mathbf{clust}}_{\mathbf{m\text{-}SDP}}$ with $\mathbf{W} = \mathbf{d}$ as:

$$\begin{array}{rl} \min_{\mathbf{M},q} & \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{D}-\mathbf{A}}{s} \rangle \equiv \langle \mathbf{M}, \frac{\mathbf{\Sigma}}{s} \rangle \\ \text{s.t.} & \widehat{\mathbf{\Gamma}} \succeq \mathbf{0} \Leftrightarrow \mathbf{M} \succeq \mathbf{0}, \\ & \mathbf{d}'\text{diag}(\mathbf{V}\mathbf{M}\mathbf{V}') \equiv \langle \mathbf{D}, \mathbf{V}\mathbf{M}\mathbf{V}' \rangle \equiv \langle \mathbf{I}, \mathbf{M} \rangle = qs \equiv q\mathbf{d}'\mathbf{1}, \\ & \langle \mathbf{V}\mathbf{M}\mathbf{V}', \frac{\mathbf{d}\mathbf{d}'}{s^2} \rangle \equiv \langle \mathbf{M}, \frac{\mathbf{V}'\mathbf{d}\mathbf{d}'\mathbf{V}}{s^2} \rangle \equiv \frac{1}{s}\mathbf{M}(1,1) = q-1, \end{array}$$

or in summary:

$$\begin{array}{rl} \min_{\mathbf{M},q} & \langle \mathbf{M}, \frac{\mathbf{\Sigma}}{s} \rangle \\ \text{s.t.} & \mathbf{M} \succeq \mathbf{0}, \\ & \langle \mathbf{I}, \mathbf{M} \rangle = qs, \\ & \frac{1}{s}\mathbf{M}(1,1) = q-1, \end{array} \tag{7}$$

where we made use of the fact that $\mathbf{d}'\mathbf{V} = \mathbf{1}'\mathbf{D}\mathbf{V} = \sqrt{s}\mathbf{V}(:,1)'\mathbf{D}\mathbf{V} = \begin{pmatrix} \sqrt{s} & 0 & \cdots & 0 \end{pmatrix}$. We can eliminate q from these constraints, and obtain:

$$\begin{array}{rl} \min_{\mathbf{M}} & \langle \mathbf{M}, \frac{\mathbf{\Sigma}}{s} \rangle \\ \text{s.t.} & \mathbf{M} \succeq \mathbf{0}, \\ & \langle \mathbf{I}, \mathbf{M} \rangle = \mathbf{M}(1,1) + s. \end{array} \tag{8}$$

This optimization problem can be solved by inspection: its optimal solution is given by putting $\mathbf{M}(1,1) = s(q-1)$ for any $q \geq 1$, $\mathbf{M}(2,2) = s$, and $\mathbf{M}(1,2) = \mathbf{M}(2,1) = f$ for any $f \in [-s\sqrt{q-1}, s\sqrt{q-1}]$ (to ensure that $\mathbf{M} \succeq \mathbf{0}$). All other entries of $\mathbf{M}$ should be equal to 0. This means that

$$\begin{aligned} \widehat{\mathbf{\Gamma}} &= s\mathbf{V}(:,2)\mathbf{V}(:,2)' + s(q-1)\mathbf{V}(:,1)\mathbf{V}(:,1)' + f\mathbf{V}(:,2)\mathbf{1}' + f\mathbf{1}\mathbf{V}(:,2)', \\ &= s\widetilde{\mathbf{y}}\widetilde{\mathbf{y}}' + (q-1)\mathbf{1}\mathbf{1}' + f\widetilde{\mathbf{y}}\mathbf{1}' + f\mathbf{1}\widetilde{\mathbf{y}}'. \end{aligned}$$

The value of the optimum is equal to $\mathbf{\Sigma}(2,2) = \sigma_2$, the smallest nonzero generalized eigenvalue. The value of $\widehat{\mathbf{\Gamma}}$ is essentially equivalent to the vector $\widetilde{\mathbf{y}}$ from the spectral relaxation. ■

This result shows that, while the actual choice of how to choose the matrix $\mathbf{W}$ in the relaxation cascade is basically free, for interpretability it is reasonable that $\mathbf{d}$ is within its column space (as only then all relaxations in the cascade are tighter than the spectral relaxation). Well-motivated choices for $\mathbf{W}$ exist, and we will construct one in Section 4.

2.4 Discussion

So far we have introduced a cascade of relaxations of the normalized cut problem, the loosest of which is equivalent to the spectral relaxation. For each SDP relaxation we have derived a dual version, the optimum of which is a lower bound for the primal optimum (weak duality).

In this section we go further into the duality aspects of the SDP problems. In particular, we investigate whether strong duality holds, which would imply that the primal and the dual optima are *equal* to each other. Additionally, this allows us to get a better insight in the relation between the spectral relaxation and the cascade of SDP relaxations.

2.4.1 Strong Duality

Let us investigate whether the dual optimum opt_{dual} is equal to the primal optimum $\text{opt}_{\text{primal}}$, instead of merely a lower bound as guaranteed by the weak duality. If the primal and dual optima are equal to each other, one says that *strong duality* holds. Slater's condition gives a sufficient condition for strong duality to hold.

Lemma 3 (Slater's condition) *Strong duality holds if the primal problem is convex and the primal constraints are strictly feasible. Then the primal and dual optima are equal to each other.*

Hereby, strict feasibility means that a matrix $\widehat{\mathbf{\Gamma}} \succ \mathbf{0}$ along with a value for q satisfying the equality constraints can be found. As SDP problems are convex, the first condition is certainly fulfilled. That the primal constraints in our SDP relaxations are strictly feasible can be seen by construction: choose $\widehat{\mathbf{\Gamma}} = q\mathbf{I} \succ \mathbf{0}$ and $q = 1/(1 - \frac{\mathbf{d}'\mathbf{d}}{s^2})$. Hence:

Proposition 4 *The primal optimization problems* $\mathbf{P}^{\mathbf{clust}}_{\mathbf{SDP}}$ *and* $\mathbf{P}^{\mathbf{clust}}_{\mathbf{m\text{-}SDP}}$ *are strictly feasible. I.e. the Slater condition is fulfilled, and the primal and dual optima are equal to each other:*

$$opt_{dual} = opt_{primal}.$$

If the dual constraints are also strictly feasible, duality theory teaches us that the primal optimum is achieved for a finite value of the variables (see e.g. Helmberg (2000)). However, the following remark answers negatively to this presupposition:

Remark 5 *The dual optimization problems* $\mathbf{D}^{\mathbf{clust}}_{\mathbf{SDP}}$ *and* $\mathbf{D}^{\mathbf{clust}}_{\mathbf{m\text{-}SDP}}$ *are not strictly feasible. Indeed,* $\mathbf{1}'\left(\frac{\mathbf{D}-\mathbf{A}}{s} - diag(\boldsymbol{\lambda}) + \mu\frac{\mathbf{d}\mathbf{d}'}{s^2}\right)\mathbf{1} = 0$ *for all* μ *and* $\boldsymbol{\lambda}$ *satisfying the constraint* $\mu = \mathbf{1}'\boldsymbol{\lambda}$*, and* $\mathbf{1}'\left(\frac{\mathbf{D}-\mathbf{A}}{s} - diag(\mathbf{W}\boldsymbol{\lambda}) + \mu\frac{\mathbf{d}\mathbf{d}'}{s^2}\right)\mathbf{1} = 0$ *for all* μ *and* $\boldsymbol{\lambda}$ *satisfying the constraint* $\mu = \mathbf{1}'\mathbf{W}\boldsymbol{\lambda}$*. This means that the PSD constrained matrix is never strictly positive definite. Correspondingly, the primal optimum is not achieved for a finite value of the variables.*

In particular, note that if $\widehat{\mathbf{\Gamma}}$ is a feasible point of optimization problem $\mathbf{P}^{\mathbf{clust}}_{\mathbf{SDP}}$ or of $\mathbf{P}^{\mathbf{clust}}_{\mathbf{m\text{-}SDP}}$, then also $\widehat{\mathbf{\Gamma}} + x\mathbf{1}\mathbf{1}'$ with $x \geq 0$ is a feasible point with the *same* value of the objective. The consequence is that the optimum will be achieved for matrix $\widehat{\mathbf{\Gamma}}$ with an infinitely large constant component, and hence with q infinitely large. Indeed, increasing q never increases the objective for a fixed value of $\widehat{\mathbf{\Gamma}}$ as the Remark above shows. This also means that the minimum over $\widehat{\mathbf{\Gamma}}$ can only be smaller for q larger, such that the minimum over both q and $\widehat{\mathbf{\Gamma}}$ is obtained for q unboundedly large. What does this mean? A more in-depth study of the relation between the spectral and SDP relaxations makes things clear.

2.4.2 How Much Tighter Are the SDP Relaxations?

We have already shown in Theorem 2 that the SDP relaxation from the cascade with $\mathbf{W} = \mathbf{d}$ is equivalent to the spectral relaxation. Here we prove an even stronger theorem that relates the solution of the basic SDP relaxation, which is the tightest of all, to the spectral one. Our insights gained in the previous section are of help here: the fact that the primal optimum is attained for q approaching infinity will be crucial in the proof. We sketch the proof, which follows a similar reasoning as in the proof of Theorem 2, in Appendix.

Theorem 6 *Also the solution of the basic SDP relaxation* $\mathbf{P}^{\mathbf{clust}}_{\mathbf{SDP}}$ *is essentially equivalent to the spectral relaxation. More specifically, the solution is given by:*

$$\widehat{\mathbf{\Gamma}} \;=\; s\widetilde{\mathbf{y}}\widetilde{\mathbf{y}}' + (q-1)\mathbf{1}\mathbf{1}' + \mathbf{m}\mathbf{1}' + \mathbf{1}\mathbf{m}',$$

with $q \to \infty$*, and* $\mathbf{m}$ *such that* $diag(\widehat{\mathbf{\Gamma}}) = q\mathbf{1}$ *and* $\mathbf{m}'\mathbf{d} = 0$*.*

This result is essentially equivalent with the result from the spectral relaxation, if we ignore the infinitely large constant matrix, and the two rank 2 matrix $\mathbf{m}'\mathbf{1} + \mathbf{1}'\mathbf{m}$ that merely makes the diagonal of the label matrix equal to a constant. A very similar theorem holds for the relaxations from the cascade $\mathbf{P}^{\mathbf{clust}}_{\mathbf{m\text{-}SDP}}$.

So, does this mean that none of the SDP relaxations is tighter than the spectral relaxation? Certainly not: the constraint set is clearly much tighter, as is obvious by looking at the relaxation cascade where constraints on the diagonal of $\widehat{\mathbf{\Gamma}}$ can explicitly be added

or omitted. However, all constraints except for the ones that are also present in the spectral relaxation are inactive. If additional constraints are imposed on the problem, some of the inactive constraints may become active, such that the tightness of the SDP relaxations starts paying off.

2.4.3 Additional Constraints on the SDP problems

A first approach is to introduce an upper bound on q as an additional constraint. Stated in terms of the original variables, this implies that the imbalance $\frac{s^2}{4s_+s_-}$ is upper bounded, which makes sense as this number should be finite for the unrelaxed optimum as well. (If desired, it is easy to compute the maximal value of $\frac{s^2}{4s_+s_-}$ that can be achieved in any bipartitioning of the graph, and this value can be used as a certain upper bound). Interestingly, introducing such an upper bound on q does not affect the correctness of Theorem 2. However, Theorem 6 ceases to hold if q is upper bounded, which was indeed the goal.

Perhaps a more elegant approach is based on the *transductive version* of the NCut relaxation, which we will present in detail in Section 3.1. The transductive version optimizes the same objective while respecting some given labels, and has at most the same number of variables and constraints as in the unconstrained NCut SDP relaxation. However, the dual is automatically strictly feasible as long as at least two data points are labeled differently. We can use this fact as follows. Instead of upper bounding q, one can pick two data points and specify their classes to be different from each other and subsequently solve the transductive NCut SDP relaxation from Section 3.1. It makes sense to pick the two most dissimilar points for this. If needed, the transductive NCut SDP relaxation can be solved for several pairs of data points, up to at most $n-1$ (which, if well chosen, is sufficient to guarantee that at least one of the pairwise inequivalence constraints was correct), although much less than $n-1$ pairs will usually be sufficient to guarantee that with high probability one of the pairwise constraints was correct. Then one proceeds with the solution that achieved the smallest normalized cut value.

2.4.4 Complexity Analysis

We are now ready to study the computational complexity to solve the derived SDP relaxations. The worst-case computational complexity of a pair of primal-dual strictly feasible SDP problems is known to be polynomial, which is achieved by publicly available software tools such as SeDuMi (Sturm, 1999).[3]

In particular for the basic SDP relaxation $\mathbf{P}^{\mathbf{clust}}_{\mathbf{SDP}}$, with (#vars) $= O(n)$ variables (in the dual SDP) and an SDP constraint of size (size SDP) $= O(n)$ the worst case complexity (based on a theoretical analysis of SDP problems without exploiting structure, see Vandenberghe and Boyd (1996)) is given by $O((\#\text{vars})^2(\text{size SDP})^{2.5}) = O(n^{4.5})$, hence the complexity of our basic SDP relaxation with an additional upper bound on q (for dual strict feasibility).

For the SDP cascade $\mathbf{P}^{\mathbf{clust}}_{\mathbf{m\text{-}SDP}}$, it is important to note that the number of dual variables is now only $O(m)$, reducing the worst case complexity down to $O(m^2n^{2.5})$. Hence, m is a

3. Other software tools that are in practice often faster exist, notably SDPLR, which we used for the large-scale experiments (Burer and Monteiro, 2003, 2005).

parameter trading off the tightness of the relaxation with the computational complexity, and can be adapted according to the available computing resources.

2.4.5 Estimating the Label Vector

In the context of the max-cut problem, several techniques have been proposed in literature to construct a good binary label vector based on a label matrix as found by the max-cut SDP relaxation (see e.g. Helmberg (2000) for an overview). Those techniques can be used here as well, and in this paper we use the *randomized rounding* technique.

2.4.6 Bounds on the Unrelaxed Minimum

Let us briefly discuss how the solution of the unrelaxed NCut optimization problem relates to the solution of any of the relaxations. First, since the feasible region is enlarged in the relaxed optimization problem, the relaxed minimum provides a *lower bound* for the minimum of the unrelaxed NCut problem.

On the other hand, the cost of any binary label vector provides an *upper bound* on the minimal cost over all label vectors. Hence, also the label vector as found by the randomized rounding technique will provide such an upper bound. In summary, each of our SDP relaxations allows us to both upper bound and lower bound the minimal NCut cost.

3. The Subspace Trick

In this section we discuss a simple trick that allows one to impose equivalence and inequivalence constraints on the labels in a very natural way. Furthermore, the very same trick leads to a fast approximation of the relaxed NCut optimization problem.

The idea is to reparameterize the label matrix $\widehat{\mathbf{\Gamma}}$ by $\widehat{\mathbf{\Gamma}} = \mathbf{RMR}'$, with $\mathbf{M}$ symmetric and $\mathbf{R}$ a fixed, specified matrix. In this way, we restrict the row and column space of the label matrix $\widehat{\mathbf{\Gamma}}$ to the columns of $\mathbf{R}$.

3.1 Imposing Label Constraints: Transduction and Learning with Side-Information

We first discuss the use of the subspace trick in the transduction scenario, and subsequently extend it to the general semi-supervised learning setting. Here we will use a *label constraint matrix* $\mathbf{L}$ for the matrix $\mathbf{R}$.

The approach of using such label constraint matrices has been used previously by De Bie et al. (2004) to derive a *spectral* relaxation of label-constrained normalized cut cost problems. In the experimental section, we will compare this spectral transduction method with the here derived SDP relaxations.

3.1.1 Transduction

By parameterizing $\widehat{\mathbf{\Gamma}}$ as $\widehat{\mathbf{\Gamma}} = \mathbf{LML}'$, it is straightforward to enforce label constraints in order to achieve a transductive version. Let us assume without loss of generality that the rows and columns of $\mathbf{A}$ are sorted such that the labeled (training) points occur first, with labels given by the label vector $\mathbf{y}_t$, and the unlabeled (test) points thereafter. Then we

define the *label constraint matrix* as:

$$\mathbf{L} = \begin{pmatrix} \mathbf{y}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

I.e., the first column of the matrix $\mathbf{L}$ consists of the given label vector $\mathbf{y}_t$, and zeros at positions corresponding to the n_{test} test points. The rest of the first block row contains zeros, and the lower right block is an identity matrix of size n_{test}. Then the label constraints can be imposed by observing that any valid $\widehat{\mathbf{\Gamma}}$ must satisfy:

$$\widehat{\mathbf{\Gamma}} = \mathbf{L}\mathbf{M}\mathbf{L}' = \begin{pmatrix} \mathbf{M}(1,1)\mathbf{y}_t\mathbf{y}_t' & \mathbf{y}_t\mathbf{M}(2:n_{\text{test}},1)' \\ \mathbf{M}(2:n_{\text{test}},1)\mathbf{y}_t' & \mathbf{M}(2:n_{\text{test}},2:n_{\text{test}}) \end{pmatrix}.$$

Indeed, rows (columns) corresponding to oppositely labeled training points then automatically are each other's opposite, and rows (columns) corresponding to same-labeled training points are equal to each other.

Using this parameterization we can easily derive the transductive NCut relaxation whose solution will by construction respect the constraints on the training labels:

$$\mathbf{P}^{\mathbf{trans}}_{\mathbf{SDP}} \begin{cases} \min_{\mathbf{M},q} & \langle \mathbf{M}, \mathbf{L}'\frac{\mathbf{D}-\mathbf{A}}{s}\mathbf{L}\rangle \\ \text{s.t.} & \mathbf{M} \succeq \mathbf{0}, \\ & \text{diag}(\mathbf{M}) = q\mathbf{1}, \\ & \langle \mathbf{M}, \mathbf{L}'\frac{\mathbf{d}\mathbf{d}'}{s^2}\mathbf{L}\rangle = q - 1. \end{cases} \qquad \mathbf{D}^{\mathbf{trans}}_{\mathbf{SDP}} \begin{cases} \max_{\boldsymbol{\lambda},\mu} & \mu, \\ \text{s.t.} & s\mathbf{L}'\frac{\mathbf{D}-\mathbf{A}}{s}\mathbf{L} - \text{diag}(\boldsymbol{\lambda}) \\ & \quad +\mu\mathbf{L}'\frac{\mathbf{d}\mathbf{d}'}{s^2}\mathbf{L} \succeq \mathbf{0}, \\ & \mu = \mathbf{1}'\boldsymbol{\lambda}. \end{cases}$$

Note that this is computationally even easier to solve than the unconstrained $\mathbf{P}^{\mathbf{clust}}_{\mathbf{SDP}}$ since the number of dual variables is $n_{\text{test}} + 2$, which decreases with an increasing number of labeled data points.

3.1.2 General Equivalence and Inequivalence Constraints

By using a different label constraint matrix, more general *equivalence and inequivalence constraints* can be imposed (Shental et al., 2004). An equivalence constraint between a pair of data points specifies that they belong to the same class. By extension, one can define an equivalence constraint for a set of points. On the other hand, an inequivalence constraint specifies two data points to belong to opposite classes. It is clear that the transduction scenario is a special case of the scenario where equivalence and inequivalence constraints are given. This large flexibility can be dealt with by using a label constraint matrix of the following form:

$$\mathbf{L} = \begin{pmatrix} \mathbf{1}_{s_1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ -\mathbf{1}_{s_2} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{s_3} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & -\mathbf{1}_{s_4} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1}_{s_{2p-1}} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{1}_{s_{2p}} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{1}_{s_{2p+1}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{1}_{s_{2p+2}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1}_{s_c} \end{pmatrix}.$$

Hereby, the ith row of $\mathbf{L}$ corresponds to the ith data point, in so that samples corresponding to one block row of size s_k are given to belong to the same class by an equivalence constraint (without loss of generality we assume that the samples are organized in this order in the affinity matrix $\mathbf{A}$). Inequivalence constraints are encoded by the first $2p$ block rows: for all $k \leq 2p$, samples from block row k are given to belong to a different class as samples from block row $k+1$. For the last $c-2p$ blocks no inequivalence constraints are given. These blocks will often contain only a single row, meaning that for the corresponding data point no equivalence nor inequivalence constraints are specified.

3.2 Approximating the SDP Relaxation for Speed-Up

Besides for imposing label constraints, the subspace trick can also be used to achieve a further speed-up of the SDP relaxations developed in the previous section. We discuss two different approaches. It is important to stress that both are approximations, and hence no genuine relaxations of the NCut problem anymore.

3.2.1 Using a Coarse Pre-clustering

The semi-supervised learning methodology lends itself to speed up the SDP relaxation itself. A useful approach would be to perform a coarse pre-clustering of the data. The equivalence constraints found by the pre-clustering can then be used as constraints in the constrained SDP relaxation of the NCut problem.

3.2.2 Using the Spectral Relaxation

Assuming that the spectral relaxation performs reasonably well, we know that the optimal label vector will be close to the generalized eigenvector $\widetilde{\mathbf{y}}$ belonging to the smallest nonzero eigenvalue σ_2, plus some constant vector (which is essentially the generalized eigenvector belonging to the smallest eigenvalue $\sigma_1 = 0$). In fact, it is likely that the optimal label vector is close to the space spanned by the eigenvectors corresponding to the d smallest generalized eigenvalues of $(\mathbf{D}-\mathbf{A})\mathbf{v} = \sigma\mathbf{D}\mathbf{v}$. We store these eigenvectors in the columns of the matrix $\mathbf{V}(:, 1:d) \in \Re^{n\times d}$. Then, we can approximate (the optimal value of) $\widehat{\mathbf{\Gamma}}$ by $\widehat{\mathbf{\Gamma}} \approx \mathbf{V}(:,1:d)\mathbf{M}\mathbf{V}(:,1:d)'$.

Since the label vector will only approximately lie in the column space of $\mathbf{V}(:,1:d)$, the equality constraint $\mathbf{W}'\mathrm{diag}(\mathbf{V}(:,1:d)\mathbf{M}\mathbf{V}(:,1:d)') = q\mathbf{W}'\mathbf{1}$ will be infeasible in general. Hence we relax this constraint to an inequality constraint:

$$\mathbf{W}'\mathrm{diag}(\mathbf{V}(:,1:d)\mathbf{M}\mathbf{V}(:,1:d)') \geq q\mathbf{W}'\mathbf{1}.$$

The resulting approximated relaxation then becomes:

$$\mathbf{P}^{\mathbf{appr}}_{\mathbf{SDP}} \left\{ \begin{array}{rl} \min_{\mathbf{M},q} & \langle \mathbf{M}, \frac{\mathbf{\Sigma}(1:d,1:d)}{s} \rangle, \\ \text{s.t.} & \mathbf{M} \succeq \mathbf{0}, \\ & \mathbf{W}'\mathrm{diag}(\mathbf{V}(:,1:d)\mathbf{M}\mathbf{V}(:,1:d)') \geq q\mathbf{W}'\mathbf{1}, \\ & \langle \mathbf{M}, \mathbf{V}(:,1:d)'\mathbf{d}\mathbf{d}'\mathbf{V}(:,1:d) \rangle = q-1. \end{array} \right.$$

$$\mathbf{D}^{\mathbf{appr}}_{\mathbf{SDP}} \left\{ \begin{array}{ll} \max_{\boldsymbol{\lambda},\mu} & \mu, \\ \text{s.t.} & \frac{\boldsymbol{\Sigma}(1:d,1:d)}{s} - \mathbf{V}(:,1:d)'\text{diag}(\mathbf{W}\boldsymbol{\lambda})\mathbf{V}(:,1:d) \\ & \quad +\mu\mathbf{V}(:,1:d)'\frac{\mathbf{d}\mathbf{d}'}{s^2}\mathbf{V}(:,1:d) \succeq \mathbf{0}, \\ & \mu = \mathbf{1}'\mathbf{W}\boldsymbol{\lambda}, \\ & \boldsymbol{\lambda} \geq 0. \end{array} \right.$$

Note that the number of dual variables is equal to $m+1$, and the size of the dual PSD constraint is d. Hence, the computational complexity is now reduced to $O(m^2d^{2.5} + dn^2)$, where the second term arises from the computation of the generalized eigenvectors $\mathbf{V}$ corresponding to the d smallest eigenvalues.

4. Implications Beyond the Normalized Cut

The methodology, developed in this paper in the context of (constrained) NCut bipartitioning, can be used for other combinatorial problems as well. For example, the extension of the developed techniques towards the ACut cost function is straightforward. We briefly discuss the applicability to another example, namely the well-known max-cut problem. For this problem we will also discuss a specific choice of $\mathbf{W}$ in the cascade of SDP relaxations.

4.1 The Max-Cut Problem

The SDP-relaxed max-cut problem is given by (Goemans and Williamson, 1995; Helmberg, 2000):

$$\mathbf{P}^{\mathbf{max\text{-}cut}} \left\{ \begin{array}{ll} \max_{\boldsymbol{\Gamma}} & \frac{1}{4}\langle \boldsymbol{\Gamma}, \mathbf{D}-\mathbf{A}\rangle \\ \text{s.t.} & \boldsymbol{\Gamma} \succeq \mathbf{0}, \\ & \text{diag}(\boldsymbol{\Gamma}) = \mathbf{1}. \end{array} \right. \qquad \mathbf{D}^{\mathbf{max\text{-}cut}} \left\{ \begin{array}{ll} \min_{\boldsymbol{\lambda}} & \mathbf{1}'\boldsymbol{\lambda}, \\ \text{s.t.} & -\frac{1}{4}(\mathbf{D}-\mathbf{A}) + \text{diag}(\boldsymbol{\lambda}) \succeq \mathbf{0}. \end{array} \right.$$

where again $\boldsymbol{\Gamma} \approx \mathbf{y}\mathbf{y}'$ is the label matrix, with $\mathbf{y} \in \{-1,1\}^n$. Just as for the NCut optimization problem, we can relax the constraints on the diagonal to the m constraints $\mathbf{W}'\text{diag}(\boldsymbol{\Gamma}) = \mathbf{W}'\mathbf{1}$ with $\mathbf{W} \in \Re^{n\times m}$. For $\mathbf{W} = \mathbf{1}$ (for $m=1$), the well-known spectral relaxation of max-cut is obtained:

$$\frac{1}{4}(\mathbf{D}-\mathbf{A})\mathbf{v} = \sigma\mathbf{v},$$

where the dominant eigenvector is an approximation for the maximal cut.

Also the subspace trick can readily be applied here, to give rise to label-constrained max-cut relaxations, or to approximations of the max-cut relaxation to control the computational burden. Here, let us define the matrix $\mathbf{V}$ as containing the eigenvectors of the above eigenvalue problem in order of *decreasing* eigenvalue. Then, the approximated max-cut relaxation becomes:

$$\mathbf{P}^{\mathbf{max\text{-}cut\ appr}} \left\{ \begin{array}{ll} \max_{\mathbf{M}} & \frac{1}{4}\langle \boldsymbol{\Gamma}, \mathbf{D}-\mathbf{A}\rangle \\ \text{s.t.} & \mathbf{M} \succeq \mathbf{0}, \\ & \mathbf{W}'\text{diag}(\mathbf{V}(:,1:d)\mathbf{M}\mathbf{V}(:,1:d)') \leq \mathbf{W}'\mathbf{1}. \end{array} \right.$$

$$\mathbf{D}^{\mathbf{max\text{-}cut\ appr}} \left\{ \begin{array}{ll} \min_{\boldsymbol{\lambda}} & \mathbf{1}'\boldsymbol{\lambda}, \\ \text{s.t.} & -\frac{1}{4}(\mathbf{D}-\mathbf{A}) + \text{diag}(\mathbf{W}\boldsymbol{\lambda}) \succeq \mathbf{0}, \\ & \boldsymbol{\lambda} \geq \mathbf{0}. \end{array} \right.$$

A good W for max-cut The matrix $\mathbf{W}$ can essentially be chosen freely, as long as $\mathbf{1}$ is within its column space, in order to maintain the interpretation that for $m = 1$ the spectral relaxation results, and to ensure that for $m > 1$ the relaxation is stricter than for $m = 1$. In particular, we propose to design $\mathbf{W}$ as follows. First, a partition of the data points in m subsets is made. Then, each subset of the partition corresponds to a column of $\mathbf{W}$, and the row-entries in each column that are within the corresponding subset are set equal to 1, the others are kept to 0. The result is that the constraints on the diagonal $\mathbf{W}'\text{diag}(\mathbf{V}(:, 1:d)'\boldsymbol{\lambda}\mathbf{V}(:, 1:d)) \leq \mathbf{W}'\mathbf{1}$ are effectively constraints on sums of subsets of diagonal elements. Clearly, $\mathbf{W1} = \mathbf{1}$ so $\mathbf{1}$ is in $\mathbf{W}$'s column space, as desired. In order to make these constraints as strong as possible, we use the heuristic to put points with a large value in the result of the spectral relaxation in the same subset of the partition. More specifically, we sort the entries of the relaxed label vector from the spectral relaxation, and construct the partition such that the m subsets are (roughly) equally large and such that data points in the same subset occur consecutively in this sorted ordering. This is the approach we use in the empirical results section.

5. Empirical Results

In this section we empirically evaluate the basic SDP relaxation of the NCut problem and its use for transduction. Next, we investigate the cascade of relaxations for the max-cut problem, and the subspace trick to speed up the calculations.

5.1 NCut Clustering and Transduction

In all experiments for NCut clustering and transduction, we use the randomized rounding technique (with 100 random projections) to derive a crisp label vector from the label matrix $\widehat{\mathbf{\Gamma}}$, and K-means on the relaxed label vector $\widetilde{\mathbf{y}}$ obtained from the spectral relaxation. All optimization problems related to the NCut cost function are implemented using the SeDuMi SDP solver (Sturm, 1999).

5.1.1 A Few Toy Problems

The results obtained by using the basic SDP relaxation for a few 2-dimensional clustering problems are summarized in Figure 1. A Gaussian kernel is used with kernel width equal to the average over all data points of the distance to their closest neighbor. In all these cases the resulting label matrix turned out to be indistinguishable from a perfect 1 / -1 label matrix.

5.1.2 Clustering and Transduction on Text

We use the data from De Bie and Cristianini (2004b) to evaluate the clustering and transduction performance of the basic SDP relaxation of the NCut optimization problem. The data set contains 195 articles of the Swiss constitution, each translated in 4 languages (English, French, German and Italian). The articles are grouped into so-called 'Titles', which are topics in the constitution. We use a 20-nearest neighbor affinity matrix $\mathbf{A}$ (meaning that two documents have affinity 1 if they are in the set of 20 nearest neighbors of each other, 0.5 if one is in the set of 20 nearest neighbors of the other but not vice versa, and 0

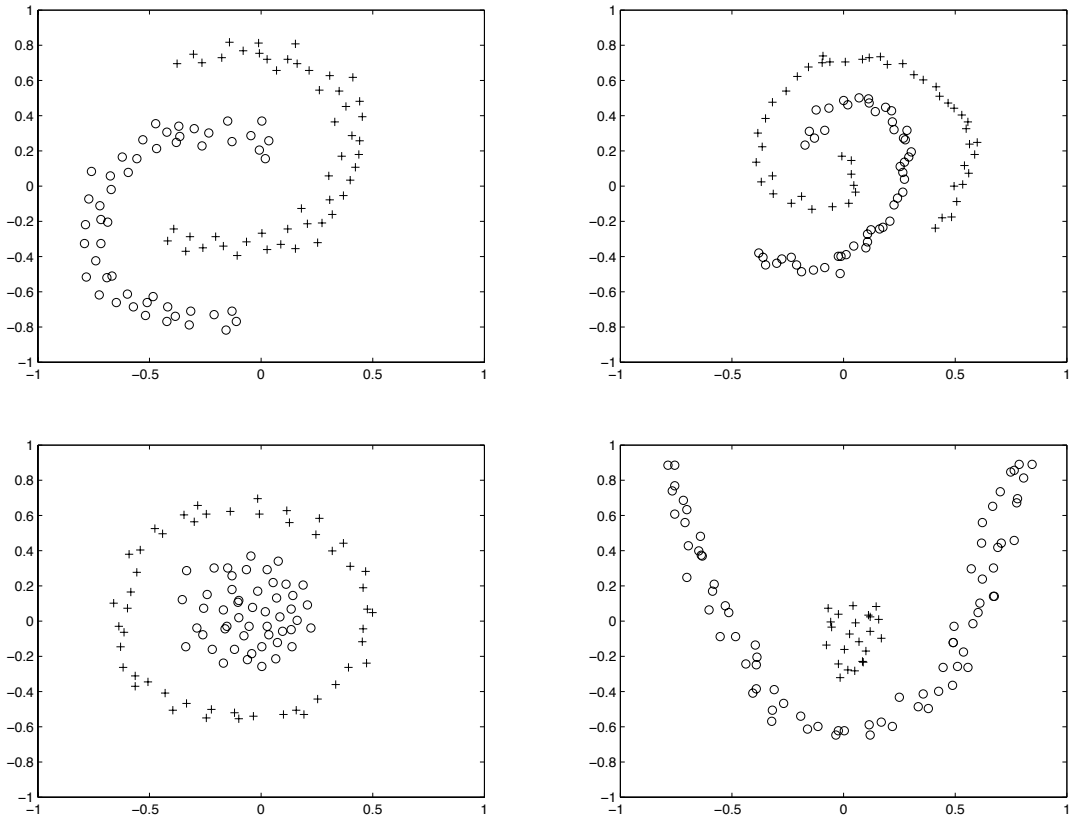


Figure 1: The labeling obtained by the SDP relaxation on 4 toy problems. All results are balanced, except for the last one.

otherwise). The distance used is the cosine distance on the bag of words representation of the documents (computed after stemming and stop word removal), i.e. 1 minus the cosine between both bag of words vectors.

We consider two reasonable divisions of the data as target clusterings. The first division clusters all articles in English with those in French, and those in German with those in Italian. The second clustering is by topic (independent of the language): it clusters those articles in the largest 'Title' together in one cluster, and the articles in all other 'Titles' in the other cluster. Clearly, considering we are using a bag of words kernel, the distinction by language is more natural. However, since there are 4 languages, several bipartitionings are likely to be more or less as natural.

Figures 2 contain the relaxed minimal cost for the transductive spectral relaxation (De Bie et al., 2004) and for the transductive SDP relaxation developed in this paper, as well as the costs corresponding to the label vectors derived from them, as a function of the fraction of data points labeled. The left graph reports the results for the (easy) clustering by language, the right one for the (harder) clustering by topic. Both graphs confirm that the lower bound on the true (unrelaxed) minimum, provided by the SDP relaxation minimum, is consistently (and significantly) tighter than the one provided by the spectral relaxation. Furthermore, the cost of the label vector derived from the spectral relaxation is consistently and significantly larger than the cost of the SDP derived solution. The leftmost points in the figures correspond to the unsupervised case (for the SDP relaxation we used the second approach explained in Section 2.4.3). Note that these unsupervised optima are considerably smaller than the value of the NCut for the true label vector, which is given by the rightmost points in both figures (100% of the data points labeled). This is especially true for the harder problem that ignores languages and clusters based on topic, which is not a surprise. In other words, both target clusterings correspond to a considerably larger cost than the optimal clustering. This result supports the conclusion of Xing and Jordan (2003) that the NCut cost function is not always a good cost function to use for clustering.

On the other hand, even a limited amount of label information seems to guide the prediction to the correct target clustering, even (although to a lesser extent) for the more unnatural clustering by topic. Consider Figure 3, where the test set accuracies for both transduction experiments are shown, again as a function of the fraction of labeled data points. On the left, the performance for the clustering by language is seen to steeply improve for a very small number of labeled data points, to saturate at a level above 0.95. On the right, we see that for the harder less natural division the improvement is less dramatic, and needs more label information, which is to be expected. Interesting to note is that for the easier problem (left figure), the spectral relaxation and the SDP relaxation perform exactly equally, while the SDP problem responds significantly better to label information than the spectral relaxation for the harder problem (right figure). This is evidence for the fact that in a transductive regime, the NCut cost function may be a good one indeed, and it is beneficial to approximate it as well as possible.

5.2 Max-Cut

We use the max-cut problem to conduct an in-depth analysis of the computational consequences of the relaxation cascade and of the subspace trick. We make use of a number of

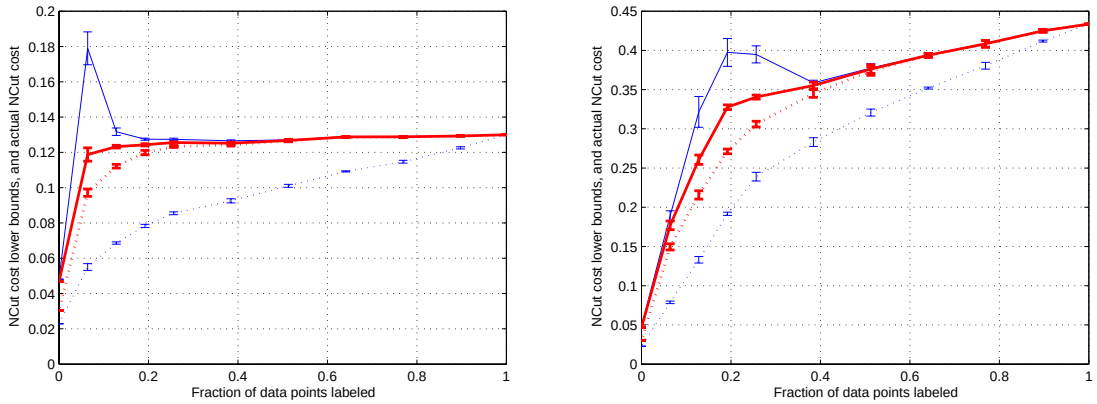


Figure 2: The costs for the best solution over 100 random roundings based on the SDP solution $\widehat{\mathbf{\Gamma}}$ (full bold line), and by performing K-means on the generalized eigenvector $\widetilde{\mathbf{y}}$ of the spectral relaxation (full faint line). This is done in the transductive scenario where the fraction of points labeled is given by the horizontal axis. Hence, the leftmost points in the graph are for the completely unsupervised scenario, and the rightmost points are equal to the cost of the target solution. The dotted lines show the lower bounds provided by the optima of both relaxed problems (bold for the SDP and faint for the spectral relaxation). The plot shows averages (and standard deviations) over 5 random selections of the training set. The left figure is for the clustering by language, the right is for the (harder) clustering by topic.

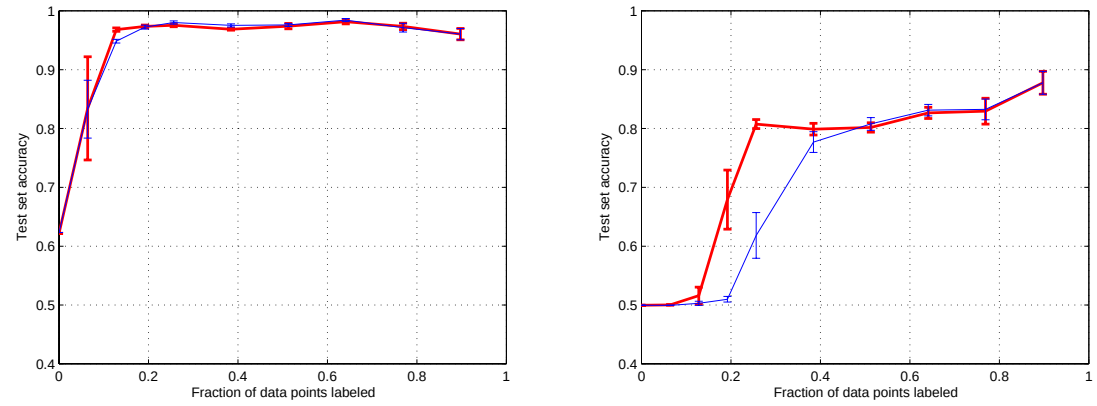


Figure 3: The test set accuracies for the best solution over 100 random roundings based on the SDP solution (bold), and by performing K-means on the generalized eigenvector of the spectral relaxation (faint). Again the horizontal axis represents the fraction of data points labeled. Averages and standard deviations over 5 random selections of the training set are shown. The left figure is for the clustering by language, the right one is for the clustering by topic.

G	$\lvert V\rvert$	$\lvert E\rvert$	density
1	800	19176	6.00
22	2000	19990	1.47
58	5000	29570	0.24
64	7000	41459	0.17
67	10000	20000	0.04
81	20000	40000	0.02

Table 2: The benchmark graphs from the Gset collection. The first column is the identifier of the graph, the second the number of vertices in the graph, the third the number of edges, and the last column shows the edge-density of the graph.

publicly available benchmark data sets from the so-called Gset collection (Helmberg and Rendl, 2000). For a summary of the graphs we used, see Table 2. For these graphs, Figure 4, shows the results of a computational analysis of the relaxation cascade and of the subspace trick as outlined in Section 4.1. In all experiments, a crisp label vector is derived from a relaxed vector (obtained using the spectral relaxation) by simple thresholding around 0, and from a relaxed label matrix by using the randomized rounding technique explained in Section 2.4.5 with 100 random projections (for the SDP relaxations). Motivated by the large size of some of the graphs in the Gset collection, we use the highly effective SDP solver SDPLR (Burer and Monteiro, 2003, 2005) called from within MATLAB in all max-cut experiments on which we report here.

For the relaxation cascade, there is only one parameter to study the effect of: the number of constraints m on the trace of the label matrix. We varied this parameter over all values $1, 2, 4, 8, 16, 32, 64, 128, 256$ and n, where for $m = 1$, the algorithm reduces to the spectral relaxation, and for $m = n$ the well-known SDP relaxation is obtained. In Figures 4, the value of the cut for each of these values of m is plotted as a function of the computation time (full line with cross markers). Average times and cuts over 10 simulations are shown, to account for the randomness of the rounding procedure and effect on the running time of the random initialization of the optimization procedure. Apparently, already a relatively small value for m and correspondingly small increase in computation time results in a significant increase of the cut found. Still, for the two largest graphs in the benchmark, our Pentium 2GHz with 1Mb RAM was unable to solve any of the SDP formulations, for memory reasons, and only one cross is plotted for $m = 1$, the spectral relaxation.

The small dots in the figures give an idea of the effect of the subspace trick, for subspace dimensionality d equal to $d = 2, 4, 8, 16, 32$, in combination with the values for m (except for 1 and n) used as above in the relaxation cascade. I.e., there are $5 \times 8 = 40$ dots in each plot. Clearly the subspace trick allows one to achieve a generally higher cut value at a significantly reduced computational cost. Using the subspace approximation, it is also possible to find a better cut than the one found using the spectral relaxation for the two most challenging problems below in the figure.

Even though the cascade of relaxations empirically appears less efficient in obtaining good approximations to the relaxed optimum, a major disadvantage of the subspace trick is

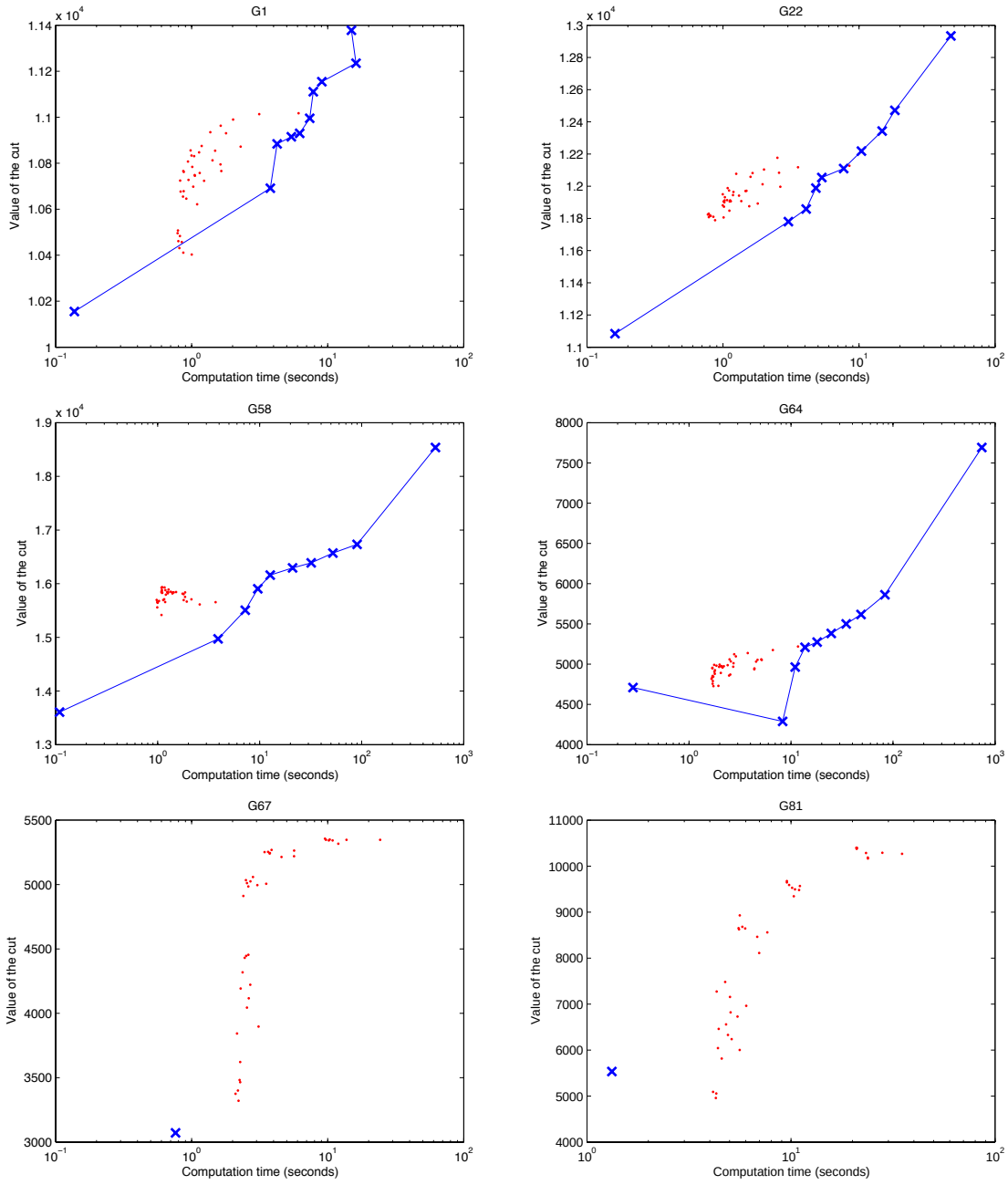


Figure 4: These plots show the value of the cut as a function of the running time for various parameter settings, each figure for another benchmark graph from the Gset collection: G1, G22, G58, G64, G67 and G81. Note the logarithmic scale on the time axis. The crosses correspond to the relaxation cascade, with $m = 1, 2, 4, \ldots, 256, n$. The small dots correspond to the use of the subspace trick for the same values of m and for various dimensionality d of the subspace: $d = 2, 4, 8, 16, 32$. For the last two graphs G67 abd G81, the relaxation cascade requires too much memory to solve on a Pentium 2GHz with 1Gb Ram and is therefore omitted (except for $d = 1$, the spectral relaxation).

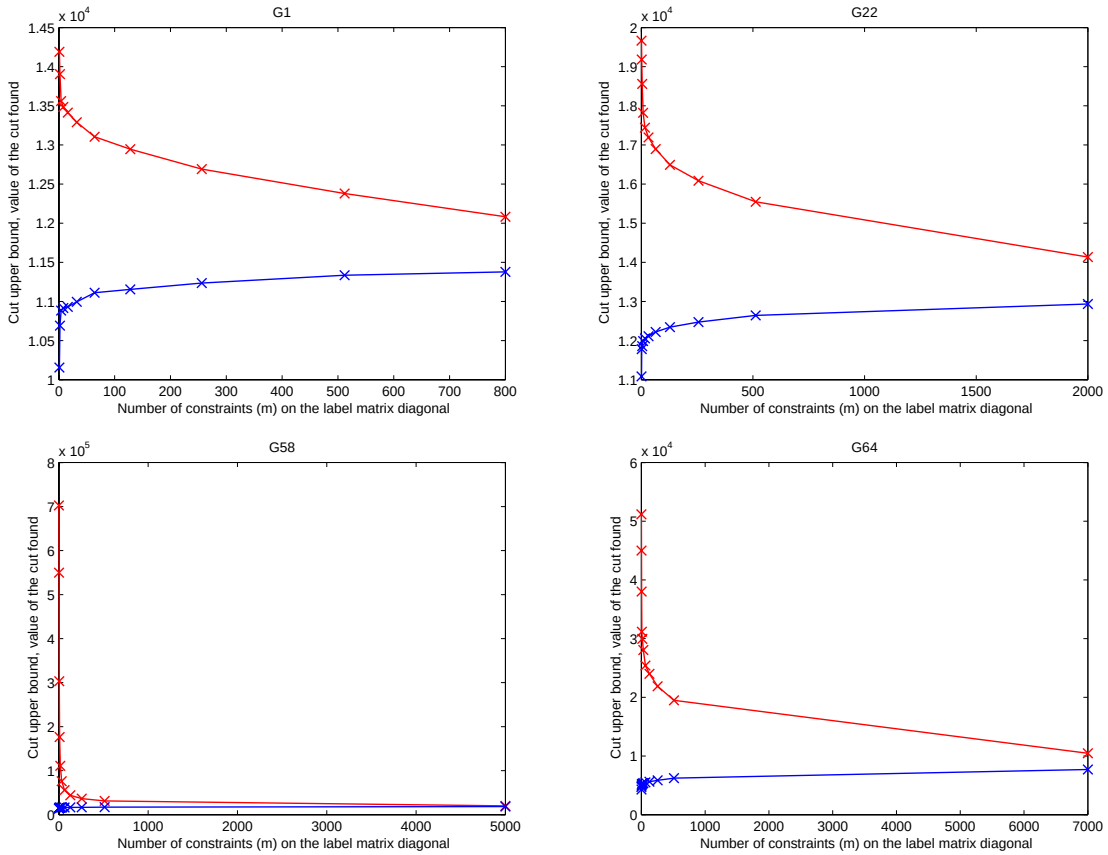


Figure 5: The upper bound on the cut as provided by the SDP relaxations (upper curves), along with the actual cut found (lower curves), for the benchmark graphs G1, G22, G58 and G64.

that it is an approximation and not a genuine relaxation. The result is that the application of the subspace trick makes the relaxed optimum useless to bound the value of the true optimum.[4] So let us now investigate to what extent the cascade of relaxations is helpful in obtaining a bound on the unrelaxed optimum in those cases where computing the full SDP relaxation is too time consuming. Figure 5 contains the value of the max-cut relaxations as a function of the number of constraints m on the diagonal of the label matrix, as well as the actual value of the cut found. One can see that the SDP upper bound on the maximum indeed decreases (that is, tightens) for increasing m. At the same time, for larger m the objective value (the cut cost) for the found label vector increases. Interestingly, it increases (and the upper bound decreases) rather steeply in the beginning and then flattens off, suggesting that a relatively low value for m may often be sufficient in practical cases.

4. Such bounds are of use for example when a Branch&Bound method is employed to find the exact optimum of the combinatorial problem.

6. Conclusions

We proposed a new cascade of SDP relaxations of the NP-complete normalized graph cut optimization problem. On both extremes of the cascade are the well-know spectral relaxation and a newly proposed SDP relaxation. The proposed relaxations directly translate into efficient machine learning algorithms for unsupervised and semi-supervised learning.

The availability of a series of relaxations with different computational complexity and tightness allows one to trade off the computational cost versus accuracy. Furthermore, we introduced the 'subspace trick', which is a simple technique that makes it possible to efficiently impose label constraints on the result of the relaxed optimization problem. Besides this, the subspace trick provides ways to obtain approximate formulations of the relaxed optimization problems with a further reduced computational cost. We believe that an interesting aspect of the paper is the fact that the techniques presented may prove useful in relaxations of other combinatorial problems as well, as witnessed by their application to the max-cut problem.

The application of these efficient approximations to machine learning algorithms might have the potential to finally fulfill the promise of SDP as a powerful new addition to the machine-learning toolbox.

We reported encouraging empirical results for the use of the NCut cost function and more in particular of its newly proposed SDP relaxation for clustering and for semi-supervised learning. Furthermore, we illustrated the use of the cascade of relaxations and of the subspace trick on the max-cut problem.

An interesting research direction opened in this paper is the question which are good and and efficiently computable choices for the matrix $\mathbf{W}$, both for the relaxation cascade and for the subspace approximation that is based on it. An answer to this question may have broad implications in the field of combinatorial optimization and relaxation theory.

An alternative avenue that can be followed to increase the scalability of SDP relaxations can be found in Lang (2004). It is based on the exploiting the ideas behind the SDPLR method (Burer and Monteiro, 2003), and works by explicitly restricting the rank of the label matrix. Further research should clarify potential relations and synergies between their method and the approaches developed in this paper.

Acknowledgments

We are grateful to Sandor Szedmak for some very useful suggestions, and to the anonymous reviewers who considerably improved the quality of this work. Furthermore, we kindly acknowledge the help of Samuel Burer in using the SDPLR method. Nello Cristianini is supported by the NIH grant No. R33HG003070-01. Tijl De Bie acknowledges support from the CoE EF/05/007 SymBioSys, and from GOA/2005/04 (both from the Research Council K.U.Leuven), and from the IST Programme of the European Community under the PASCAL Network of Excellence (IST-2002-506778).

Appendix A. Proof of Theorem 6

While this theorem would be easy to prove by plugging in the result provided in the theorem statement, for the sake of clarity we give here a prove that derives the result rather than posing it.

Proof Let us use the same notation as in the proof of Theorem 2. Then we can rewrite $\mathbf{P}^{\mathbf{clust}}_{\mathbf{SDP}}$ (in the same line as for Theorem 2):

$$\begin{array}{rl} \min_{\mathbf{M},q} & \langle \mathbf{M}, \frac{\mathbf{\Sigma}}{s} \rangle \\ \text{s.t.} & \mathbf{M} \succeq \mathbf{0}, \\ & \text{diag}(\mathbf{V}\mathbf{M}\mathbf{V}') = q\mathbf{1}, \\ & \frac{1}{s}\mathbf{M}(1,1) = q - 1. \end{array}$$

We can eliminate q from these constraints by substituting $q = 1 + \frac{1}{s}\mathbf{M}(1,1)$, and obtain:

$$\begin{array}{rl} \min_{\mathbf{M}} & \langle \mathbf{M}, \frac{\mathbf{\Sigma}}{s} \rangle \\ \text{s.t.} & \mathbf{M} \succeq \mathbf{0}, \\ & \text{diag}(\mathbf{V}\mathbf{M}\mathbf{V}') = \mathbf{1}\left(1 + \frac{1}{s}\mathbf{M}(1,1)\right). \end{array}$$

Let us decompose the left hand side of the last constraint in the following way:

$$\begin{array}{rl} & \text{diag}(\mathbf{V}(:,1)\mathbf{M}(1,1)\mathbf{V}(:,1)') + 2\text{diag}(\mathbf{V}(:,1)\mathbf{M}(1,2:n)\mathbf{V}(:,2:n)') \\ & +\text{diag}(\mathbf{V}(:,2:n)\mathbf{M}(2:n,2:n)\mathbf{V}(:,2:n)') \\ = & \mathbf{1}\frac{1}{s}\mathbf{M}(1,1) + 2\text{diag}(\mathbf{V}(:,1)\mathbf{M}(1,2:n)\mathbf{V}(:,2:n)') \\ & +\text{diag}(\mathbf{V}(:,2:n)\mathbf{M}(2:n,2:n)\mathbf{V}(:,2:n)'). \end{array}$$

We also rewrite the PSD constraint by using the *Schur complement lemma* as:

$$\mathbf{M} \succeq \mathbf{0} \quad \Leftrightarrow \quad \mathbf{M}(2:n,2:n) \succeq \frac{\mathbf{M}(1,:2:n)\mathbf{M}(1,2:n)'}{\mathbf{M}(1,1)}.$$

Besides, as $\mathbf{\Sigma}(1,1) = 0$ and $\mathbf{\Sigma}$ is diagonal, we can write

$$\langle \mathbf{M}, \frac{\mathbf{\Sigma}}{s} \rangle = \langle \mathbf{M}(2:n,2:n), \frac{\mathbf{\Sigma}(2:n,2:n)}{s} \rangle.$$

Then the optimization problem $\mathbf{P}^{\mathbf{clust}}_{\mathbf{SDP}}$ becomes:

$$\begin{array}{rl} \min_{\mathbf{M}} & \langle \mathbf{M}(2:n,2:n), \frac{\mathbf{\Sigma}(2:n,2:n)}{s} \rangle \\ \text{s.t.} & \mathbf{M}(2:n,2:n) \succeq \frac{\mathbf{M}(1,:2:n)\mathbf{M}(1,2:n)'}{\mathbf{M}(1,1)}, \\ & 2\text{diag}(\mathbf{V}(:,1)\mathbf{M}(1,2:n)\mathbf{V}(:,2:n)') + \text{diag}(\mathbf{V}(:,2:n)\mathbf{M}(2:n,2:n)\mathbf{V}(:,2:n)') = \mathbf{1}. \end{array}$$

We can see that the variable $\mathbf{M}(1,1)$ only occurs in the first constraint, and that this constraint becomes less stringent for $\mathbf{M}(1,1) \to \infty$ (note that this corresponds to $q \to \infty$, which should not be a surprise), such that the minimum will be achieved for $\mathbf{M}(1,1)$ unboundedly large. So let us already take the limit for $\mathbf{M}(1,1)$ to infinity, which gives:

$$\begin{array}{rl} \min_{\mathbf{M}} & \langle \mathbf{M}(2:n,2:n), \frac{\mathbf{\Sigma}(2:n,2:n)}{s} \rangle \\ \text{s.t.} & \mathbf{M}(2:n,2:n) \succeq \mathbf{0}, \\ & 2\text{diag}(\mathbf{V}(:,2:n)\mathbf{M}(2:n,1)\mathbf{V}(:,1)') + \text{diag}(\mathbf{V}(:,2:n)\mathbf{M}(2:n,2:n)\mathbf{V}(:,2:n)') = \mathbf{1}. \end{array}$$

The dual of this optimization problem, using Lagrange multipliers $\boldsymbol{\lambda}$ for the equality constraint on the diagonal and the symmetric matrix $\boldsymbol{\Xi}$ for the PSD constraint, is given by:

$$\begin{array}{rl} \max_{\boldsymbol{\lambda},\boldsymbol{\Xi}} & \mathbf{1}'\boldsymbol{\lambda} \\ \text{s.t.} & \boldsymbol{\Xi} = \boldsymbol{\Sigma}(2:n,2:n) - \mathbf{V}(:,2:n)'\text{diag}(\boldsymbol{\lambda})\mathbf{V}(:,2:n), \\ & \boldsymbol{\Xi} \succeq \mathbf{0}, \\ & \mathbf{V}(:,2:n)'\boldsymbol{\lambda} = 0, \end{array}$$

or equivalently:

$$\begin{array}{rl} \max_{\boldsymbol{\lambda}} & \mathbf{1}'\boldsymbol{\lambda} \\ \text{s.t.} & \boldsymbol{\Sigma}(2:n,2:n) - \mathbf{V}(:,2:n)'\text{diag}(\boldsymbol{\lambda})\mathbf{V}(:,2:n) \succeq \mathbf{0}, \\ & \mathbf{V}(:,2:n)'\boldsymbol{\lambda} = 0. \end{array}$$

From the equality constraint we can immediately see that $\boldsymbol{\lambda} \propto \mathbf{d}$, such that $\mathbf{V}(:,2:n)'\text{diag}(\boldsymbol{\lambda})\mathbf{V}(:,2:n) \propto \mathbf{V}(:,2:n)'\mathbf{D}\mathbf{V}(:,2:n) = \mathbf{I}$. Therefrom we can see that $\boldsymbol{\lambda} = \sigma_2\mathbf{d}$ and hence $\boldsymbol{\Xi} = \boldsymbol{\Sigma}(2:n,2:n) - \sigma_2\mathbf{I}$ at the optimum (where σ_2 is used to denote $\boldsymbol{\Sigma}(2,2)$, the smallest generalized eigenvalue of the spectral relaxation that is different from 0). The optimum itself is equal to $\mathbf{1}'\boldsymbol{\lambda} = \sigma_2$.

To determine the value of the primal variables at the optimum, let us now have a look at the Karush Kuhn Tucker (KKT) condition corresponding to the PSD constraint:

$$\langle \mathbf{M}(2:n,2:n), \boldsymbol{\Xi} \rangle = \langle \mathbf{M}(2:n,2:n), \boldsymbol{\Sigma}(2:n,2:n) - \sigma_2\mathbf{I} \rangle \quad = \quad 0$$

The implies that $\mathbf{M}(i,j) = 0$ for all $i \geq 2$ and $j \geq 2$ except for $i = j = 2$. The exact value of $\mathbf{M}(2,2)$ can be derived from the second KKT condition:

$$\begin{array}{rl} & \left(2\text{diag}(\mathbf{V}(:,2:n)\mathbf{M}(2:n,1)\mathbf{V}(:,1)') + \text{diag}(\mathbf{V}(:,2:n)\mathbf{M}(2:n,2:n)\mathbf{V}(:,2:n)') - \mathbf{1}\right)'\boldsymbol{\lambda} \\ = & \left(\frac{2}{\sqrt{s}}\mathbf{V}(:,2:n)\mathbf{M}(2:n,1) + \text{diag}(\mathbf{V}(:,2:n)\mathbf{M}(2:n,2:n)\mathbf{V}(:,2:n)') - \mathbf{1}\right)'\sigma_2\mathbf{d} \\ = & 0 \end{array}$$

From this follows that $\langle \mathbf{D}, \mathbf{V}(:,2:n)\mathbf{M}(2:n,2:n)\mathbf{V}(:,2:n)' \rangle = \langle \mathbf{I}, \mathbf{M}(2:n,2:n) \rangle = s$. Thus, we obtain that $\mathbf{M}(2,2) = s$.

The value of $\mathbf{M}(1,2:n)$ can straightforwardly be determined based on the equality constraints in the primal problem. The final solution is thus given by: $\mathbf{M}(1,1) = s(q-1) = \infty$, $\mathbf{M}(2,2) = s$ and $\mathbf{M}(2:n,1)$ as determined by the equality constraints of the primal problem. If we define $\mathbf{m} \in \Re^n$ as $\mathbf{m} = \frac{1}{\sqrt{s}}\mathbf{V}(:,2:n)\mathbf{M}(2:n,1)$ (satisfying $\mathbf{m}'\mathbf{d} = 0$), we can state this result conveniently in terms of the original variables:

$$\begin{array}{rcl} \widehat{\boldsymbol{\Gamma}} & = & s\mathbf{V}(:,2)\mathbf{V}(:,2)' + (q-1)\mathbf{1}\mathbf{1}' + \mathbf{m}\mathbf{1}' + \mathbf{1}\mathbf{m}', \\ & = & s\widetilde{\mathbf{y}}\widetilde{\mathbf{y}}' + (q-1)\mathbf{1}\mathbf{1}' + \mathbf{m}\mathbf{1}' + \mathbf{1}\mathbf{m}'. \end{array}$$

with $q \to \infty$. ■

References

M. F. Anjos and H. Wolkowicz. Strengthened semidefinite relaxations via a second lifting for the MAX-CUT problem. *Discrete Applied Mathematics*, 119:79–106, 2002.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. of the 18th International Conf. on Machine Learning (ICML)*, 2001.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, U.K., 2004.

S. Burer and R. D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (series B)*, 95(2):329–357, 2003.

S. Burer and R. D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming (series A)*, 103(3):427–444, 2005.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, MA, 2003.

N. Cristianini, J. Shawe-Taylor, and J. Kandola. Spectral kernel methods for clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

T. De Bie and N. Cristianini. Convex methods for transduction. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004a.

T. De Bie and N. Cristianini. Kernel methods for exploratory data analysis: a demonstration on text data. In *Proceedings of the International Workshop on Statistical Pattern Recognition (SPR2004)*. Lisbon, Portugal, August 2004b.

T. De Bie, J. Suykens, and B. De Moor. Learning from general label constraints. In *Proceedings of IAPR International Workshop on Statistical Pattern Recognition (SPR)*. 2004.

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42 (6):1115–1145, 1995.

C. Helmberg. Semidefinite programming for combinatorial optimization. Habilitationsschrift ZIB-Report ZR-00-34, TU Berlin, Konrad-Zuse-Zentrum Berlin, 2000.

C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10:673–696, 2000.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

S. D. Kamvar, D. Klein, and C. D. Manning. Spectral learning. In *IJCAI*, 2003.

K. Lang. Finding good nearly balanced cuts in power law graphs. Technical Report YRL-2004-036, Yahoo! Research, 2004.

A. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software, Special issue on Interior Point Methods (CD supplement with software)*, 11-12:625–653, 1999.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

E. P. Xing and M. I. Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical Report CSD-03-1265, Division of Computer Science, University of California, Berkeley, 2003.

Journal of Machine Learning Research 7 (2006) 1437–1466 Submitted 9/05; Revised 10/05; Published 6/06

Maximum-Gain Working Set Selection for SVMs

Tobias Glasmachers TOBIAS.GLASMACHERS@NEUROINFORMATIK.RUB.DE
Christian Igel CHRISTIAN.IGEL@NEUROINFORMATIK.RUB.DE
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

Support vector machines are trained by solving constrained quadratic optimization problems. This is usually done with an iterative decomposition algorithm operating on a small working set of variables in every iteration. The training time strongly depends on the selection of these variables. We propose the maximum-gain working set selection algorithm for large scale quadratic programming. It is based on the idea to greedily maximize the progress in each single iteration. The algorithm takes second order information from cached kernel matrix entries into account. We prove the convergence to an optimal solution of a variant termed hybrid maximum-gain working set selection. This method is empirically compared to the prominent most violating pair selection and the latest algorithm using second order information. For large training sets our new selection scheme is significantly faster.

Keywords: working set selection, sequential minimal optimization, quadratic programming, support vector machines, large scale optimization

1. Introduction

We consider 1-norm support vector machines (SVM) for classification. These classifiers are usually trained by solving convex quadratic problems with linear constraints. For large data sets, this is typically done with an iterative decomposition algorithm operating on a small working set of variables in every iteration. The selection of these variables is crucial for the training time.

Recently, a very efficient SMO-like (*sequential minimal optimization* using working sets of size 2, see Platt, 1999) decomposition algorithm was presented by Fan et al. (2005). The main idea is to consider second order information to improve the working set selection. Independent from this approach, we have developed a working set selection strategy sharing this basic idea but with a different focus, namely to minimize the number of kernel evaluations per iteration. This considerably reduces the training time of SVMs in case of large training data sets. In the following, we present our approach, analyze its convergence properties, and present experiments evaluating the performance of our algorithm. We close with a summarizing conclusion.

1.1 Support Vector Machine Learning

We consider 1-norm soft margin SVMs for classification (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002). The learning problem at hand is defined by a set of ℓ training examples $\{(x_1,y_1),\ldots,(x_\ell,y_\ell)\}$, where the x_i are points in some input space $\mathcal{X}$ with corre-

sponding class labels $y_i = \pm 1$. A positive semi-definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ ensures the existence of a feature Hilbert space $\mathcal{F}$ with inner product $\langle \cdot, \cdot \rangle$ and a mapping $\Phi : \mathcal{X} \to \mathcal{F}$ such that $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$.

The SVM algorithm constructs a real-valued, affine linear function H on the feature space. The corresponding function $h := H \circ \Phi$ on the input space can be computed in terms of the kernel k without the need for explicit computations in $\mathcal{F}$. The zero set of H is called the separating hyperplane, because the SVM uses the sign of this function for class prediction. This affine linear function is defined through maximizing the margin, that is, the desired distance of correctly classified training patterns from the hyperplane, and reducing the sum of distances by which training examples violate this margin. The trade-off between these two objectives is controlled by a regularization parameter $C > 0$.

Training a 1-norm soft margin SVM is equivalent to solving the following ℓ-dimensional convex quadratic problem with linear constraints for $\alpha \in \mathbb{R}^\ell$:

$$\mathcal{P} \quad \left\{ \begin{array}{ll} \text{maximize} & f(\alpha) = v^T \alpha - \frac{1}{2} \alpha^T Q \alpha \\ \text{subject to} & y^T \alpha = z \\ \text{and} & 0 \leq \alpha_i \leq C \ , \ \forall i \in \{1, \ldots, \ell\} \ . \end{array} \right.$$

The requirements $y^T \alpha = z$ and $0 \leq \alpha_i \leq C$ are referred to as equality constraint and box constraints, respectively. In the SVM context the constants $v \in \mathbb{R}^\ell$ and $z \in \mathbb{R}$ are fixed to $v = (1, \ldots, 1)^T$ and $z = 0$. The matrix $Q \in \mathbb{R}^{\ell \times \ell}$ is defined as $Q_{ij} := y_i y_j k(x_i, x_j)$ and is positive semi-definite as the considered kernel function k is positive semi-definite. The vector $y := (y_1, \ldots, y_\ell)^T$, $y_i \in \{+1, -1\}$ for $1 \leq i \leq \ell$, is composed of the labels of the training patterns $x_1, \ldots, x_\ell$. The set of points α fulfilling the constraints is called the feasible region $\mathcal{R}(\mathcal{P})$ of problem $\mathcal{P}$.

An optimal solution α^* of this problem defines the function $h(x) = \sum_{i=1}^{\ell} \alpha_i^* y_i k(x_i, x) + b$, where the scalar b can be derived from α^* (e.g., see Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002).

1.2 Decomposition Algorithms

Making SVM classification applicable in case of large training data sets requires an algorithm for the solution of $\mathcal{P}$ that does not presuppose the $\ell(\ell+1)/2$ independent entries of the symmetric matrix Q to fit into working memory. The methods of choice in this situation are the so called decomposition algorithms (Osuna et al., 1997). These iterative algorithms start at an arbitrary feasible point $\alpha^{(0)}$ and improve this solution in every iteration t from $\alpha^{(t-1)}$ to $\alpha^{(t)}$ until some stopping condition is satisfied. In each iteration an active set or working set $B^{(t)} \subset \{1, \ldots, \ell\}$ is chosen. Its inactive complement is denoted by $N^{(t)} := \{1, \ldots, \ell\} \setminus B^{(t)}$. The improved solution $\alpha^{(t)}$ may differ from $\alpha^{(t-1)}$ only in the components in the working set, that is, $\alpha_i^{(t-1)} = \alpha_i^{(t)}$ for all $i \in N^{(t)}$. Usually the working set $B^{(t)}$ is limited to a fixed size $|B^{(t)}| \leq q \ll \ell$. The working set must always be larger than the number of equality constraints to allow for a useful, feasible step. In general a decomposition algorithm can be formulated as follows:

Decomposition Algorithm

1 $\alpha^{(0)} \leftarrow$ feasible starting point, $t \leftarrow 1$
2 **repeat**
3 | select working set $B^{(t)}$
4 | solve QP restricted to $B^{(t)}$ resulting in $\alpha^{(t)}$
5 | $t \leftarrow t+1$
6 **until** *stopping criterion is met*

The sub-problem defined by the working set in step 4 has the same structure as the full problem $\mathcal{P}$ but with only q variables.[1] Thus, the complete problem description fits into the available working memory and is small enough to be solved by standard tools.

For the SVM problem $\mathcal{P}$ the working set must have a size of at least two. Indeed, the sequential minimal optimization (SMO) algorithm selecting working sets of size $q = 2$ is a very efficient method (Platt, 1999). The great advantage of the SMO algorithm is the possibility to solve the sub-problem analytically (cf. Platt, 1999; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002).

1.3 Working Set Selection

Step 3 is crucial as the convergence of the decomposition algorithm depends strongly on the working set selection procedure. As the selection of the working set of a given size q that gives the largest improvement in a single iteration requires the knowledge of the full matrix Q, well working heuristics for choosing the variables using less information are needed.

There exist various algorithms for this task, an overview is given in the book by Schölkopf and Smola (2002). The most prominent ones share the strategy to select pairs of variables that mostly violate the Karush-Kuhn-Tucker (KKT) conditions for optimality and can be subsumed under the *most violating pair* (MVP) approach. Popular SVM packages such as SVM^{light} by Joachims (1999) and LIBSVM 2.71 by Chang and Lin (2001) implement this technique. The idea of MVP is to select one or more pairs of variables that allow for a feasible step and most strongly violate the KKT conditions.

Here we describe the approach implemented in the LIBSVM 2.71 package. Following Keerthi and Gilbert (2002) we define the sets

$$I := \{i \in \{1,\dots,\ell\} \,|\, y_i = +1 \wedge \alpha_i^{(t-1)} < C\} \cup \{i \in \{1,\dots,\ell\} \,|\, y_i = -1 \wedge \alpha_i^{(t-1)} > 0\}$$
$$J := \{i \in \{1,\dots,\ell\} \,|\, y_i = +1 \wedge \alpha_i^{(t-1)} > 0\} \cup \{i \in \{1,\dots,\ell\} \,|\, y_i = -1 \wedge \alpha_i^{(t-1)} < C\} \ .$$

Now the MVP algorithm selects the working set $B^{(t)} = \{b_1, b_2\}$ using the rule

$$b_1 := \operatorname*{argmax}_{i \in I} \left(y_i \frac{\partial f}{\partial \alpha_i}(\alpha) \right)$$
$$b_2 := \operatorname*{argmin}_{i \in J} \left(y_i \frac{\partial f}{\partial \alpha_i}(\alpha) \right) \ .$$

1. For a sub-problem the constants $v \in \mathbb{R}^q$ and $z \in \mathbb{R}$ in general differ from $(1,\dots,1)^T$ and 0, respectively, and depend on $\alpha_i^{(t-1)}$ for $i \in N^{(t)}$.

The condition $y_{b_1}\frac{\partial f}{\partial \alpha_{b_1}}(\alpha) - y_{b_2}\frac{\partial f}{\partial \alpha_{b_2}}(\alpha) < \varepsilon$ is used as the stopping criterion. In the limit $\varepsilon \to 0$ the algorithm checks the exact KKT conditions and only stops if the solution found is optimal. The MVP algorithm is known to converge to the optimum (Lin, 2001; Keerthi and Gilbert, 2002; Takahashi and Nishi, 2005).

SVM^{light} uses essentially the same working set selection method with the important difference that it is not restricted to working sets of size 2. The default algorithm selects 10 variables by picking the five most violating pairs. In each iteration an inner optimization loop determines the solution on the 10-dimensional sub-problem up to some accuracy.

Fan et al. (2005) propose a working set selection procedure which uses second order information. The first variable b_1 is selected as in the MVP algorithm. The second variable is chosen in a way that promises the maximal value of the target function f ignoring the box constraints. The selection rule is

$$b_2 := \operatorname*{argmax}_{i \in J} \left(f(\alpha^{\max}_{\{b_1,i\}}) \right) \quad .$$

Here, $\alpha^{\max}_{\{b_1,i\}}$ is the solution of the two-dimensional sub-problem defined by the working set $\{b_1, i\}$ at position $\alpha^{(t-1)}$ considering only the equality constraint. For this second order algorithm costly kernel function evaluations may become necessary which can slow down the entire algorithm. These kernel values are cached and can be reused in the gradient update step, see equation (2) in Section 2.1. Because this algorithm is implemented in version 2.8 of LIBSVM, we will refer to it as the LIBSVM-2.8 algorithm.

The simplest feasible point one can construct is $\alpha^{(0)} = (0,\ldots,0)^T$, which has the additional advantage that the gradient $\nabla f(\alpha^{(0)}) = v = (1,\ldots,1)^T$ can be computed without kernel evaluations. It is interesting to note that in the first iteration starting from this point all components of the gradient $\nabla f(\alpha^{(0)})$ of the objective function are equal. Thus, the selection schemes presented above have a freedom of choice for the selection of the first working set. In case of LIBSVM, for b_1 simply the variable with maximum index is chosen in the beginning. Therefore, the order in which the training examples are presented is important in the first iteration and can indeed significantly influence the number of iterations and the training time in practice.

Other algorithms select a *rate certifying pair* (Hush and Scovel, 2003). The allurement of this approach results from the fact that analytical results have been derived not only about the guaranteed convergence of the algorithm, but even about the rate of convergence (Hush and Scovel, 2003; List and Simon, 2005). Unfortunately, in practice these algorithms seem to perform rather poorly.

1.4 Related Methods

Several new training algorithms for SVMs have been developed recently, which could be considered for large scale data sets. One possibility to reduce the computational cost for huge data sets is to determine only rough approximate solutions of the SVM problem. Algorithms emerged from this line of research include the Core Vector Machine by Tsang et al. (2005) and LASVM by Bordes et al. (2005), which have the additional advantage to produce even sparser solutions than the exact SVM formulation. In theory, both methods can solve the exact SVM problem with arbitrary accuracy, but their strength lies in the very fast computation of relatively rough approximate solutions. Both methods can profit from our working set selection algorithm presented below, as the Core Vector Machine uses an inner SMO loop and LASVM is basically an online version of SMO.

The SimpleSVM algorithm developed by Vishwanathan et al. (2003) provides an alternative to the decomposition technique. It can handle a wide variety of SVM formulations, but is limited in the large scale context by its extensive memory requirements. In contrast, Keerthi et al. (2000) present a geometrically inspired algorithm with modest memory requirements for the exact solution of the SVM problem. A drawback of this approach is that it is not applicable to the standard one-norm slack penalty soft margin SVM formulation, which we consider here, because it requires the classes to be linearly separable in the feature space $\mathcal{F}$.

2. Maximum-Gain Working Set Selection

Before we describe our new working set selection method, we recall how the quadratic problem restricted to a working set can be solved (cf. Platt, 1999; Cristianini and Shawe-Taylor, 2000; Chang and Lin, 2001). Then we compute the progress, the functional gain, that is achieved by solving a single sub-problem. Picking the variable pair maximizing the functional gain while minimizing kernel evaluations—by reducing cache misses when looking up rows of Q—leads to the new working set selection strategy.

2.1 Solving the Problem Restricted to the Working Set

In every iteration of the decomposition algorithm all variables indexed by the inactive set N are fixed and the problem $\mathcal{P}$ is restricted to the variables indexed by the working set $B = \{b_1, \ldots, b_q\}$. We define

$$\alpha_B = (\alpha_{b_1}, \ldots, \alpha_{b_q})^T \quad , \qquad Q_B = \begin{pmatrix} Q_{b_1 b_1} & \cdots & Q_{b_q b_1} \\ \vdots & \ddots & \vdots \\ Q_{b_1 b_q} & \cdots & Q_{b_q b_q} \end{pmatrix} \quad , \qquad y_B = (y_{b_1}, \ldots, y_{b_q})^T$$

and fix the values

$$v_B = \left(1 - \sum_{i \in N} Q_{i b_1} \alpha_i, \quad \ldots \quad , 1 - \sum_{i \in N} Q_{i b_q} \alpha_i\right)^T \in \mathbb{R}^q \quad \text{and} \quad z_B = -\sum_{i \in N} y_i \alpha_i \in \mathbb{R}$$

not depending on α_B. This results in the convex quadratic problem (see Joachims, 1999)

$$\mathcal{P}_{B,\alpha} \qquad \begin{cases} \text{maximize} & f_B(\alpha_B) = v_B^T \alpha_B - \frac{1}{2} \alpha_B^T Q_B \alpha_B \\ \text{subject to} & y_B^T \alpha_B = z_B \\ \text{and} & 0 \le \alpha_i \le C \quad \forall i \in B \ . \end{cases}$$

The value z_B can easily be determined in time linear in q, but the computation of v_B takes time linear in q and ℓ. Then $\mathcal{P}_{B,\alpha}$ can be solved using a ready-made quadratic program solver in time independent of ℓ.

We will deal with this problem under the assumption that we know the gradient vector

$$G := \nabla f_B(\alpha_B) = \left(\frac{\partial}{\partial \alpha_{b_1}} f_B(\alpha_B), \ldots, \frac{\partial}{\partial \alpha_{b_q}} f_B(\alpha_B)\right)^T \tag{1}$$

of partial derivatives of f_B with respect to all q variables $\alpha_{b_1}, \ldots, \alpha_{b_q}$ indexed by the working set. In the following, we consider SMO-like algorithms using working sets of size $q = 2$. In this case

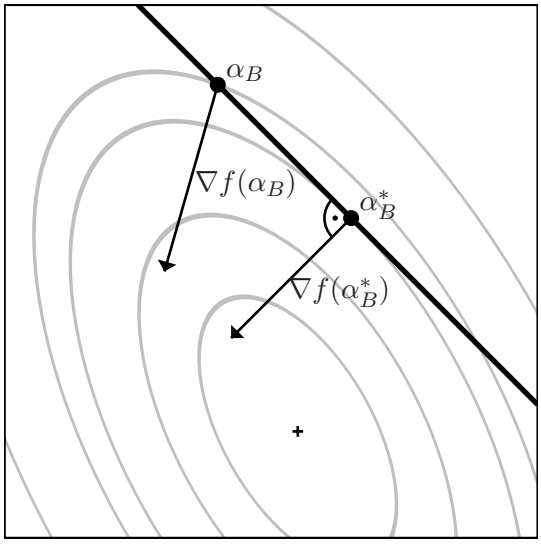


Figure 1: The 2-dimensional SMO sub-problem restricted to the equality constraint (solid 'feasible' line) and the box constraints (boundary). The point fulfilling the equality constraint with gradient orthogonal to the feasible line is a candidate for the solution of the sub-problem. If it is not feasible w.r.t. the box constraints it has to be moved along the line onto the box boundary.

the equality constraint restricts us to a line. Due to the box constraints only a bounded segment of this line is feasible (see Figure 1). To solve the restricted problem we define the vector $w_B := (1, -y_{b_1} y_{b_2})^T$ pointing along the 1-dimensional feasible hyperplane. To find the maximum on this line we look at the gradient $\nabla f_B(\alpha_B) = v_B - Q_B \alpha_B$ and compute the step $\mu_B \cdot w_B$ ($\mu_B \in \mathbb{R}$) such that the gradient $\nabla f_B(\alpha_B + \mu_B \cdot w_B)$ is orthogonal to w_B,

$$\begin{aligned} 0 &= \langle \nabla f_B(\alpha_B + \mu_B w_B), w_B \rangle \\ &= \langle v_B - Q_B \alpha_B - \mu_B Q_B w_B, w_B \rangle \\ &= \langle \nabla f_B(\alpha_B) - \mu_B Q_B w_B, w_B \rangle \ . \end{aligned}$$

Using $\nabla f_B(\alpha_B) = (G_{b_1}, G_{b_2})^T$ we get the solution

$$\mu_B^{\max} = (G_{b_1} - y_{b_1} y_{b_2} G_{b_2}) / (Q_{b_1 b_1} + Q_{b_2 b_2} - 2 y_{b_1} y_{b_2} Q_{b_1 b_2}) \ .$$

The corresponding point on the feasible line is denoted by $\alpha_B^{\max} = \alpha_B + \mu_B^{\max} w_B$. Of course, $\alpha_B^{\max}$ is not necessarily feasible. We can easily apply the box constraints to $\mu_B^{\max}$. The new solution clipped to the feasible line segment is denoted μ_B^*. The maximum of $\mathcal{P}_{B,\alpha}$ can now simply be expressed as $\alpha_B^* = \alpha_B + \mu_B^* w_B$.

After the solution of the restricted problem the new gradient

$$\nabla f(\alpha^{(t)}) = \nabla f(\alpha^{(t-1)}) - Q(\alpha^{(t)} - \alpha^{(t-1)}) \tag{2}$$

has to be computed. As the formula indicates this is done by an update of the former gradient. Because $\Delta\alpha = \alpha^{(t)} - \alpha^{(t-1)}$ differs from zero in only the b_1th and b_2th component only the corresponding two matrix rows of Q have to be known to determine the update.

2.2 Computing the Functional Gain

Expressing the target function on the feasible line by its Taylor expansion in the maximum $\alpha_B^{\max}$ we get

$$\begin{aligned}\tilde{f}_B(\xi) &:= f_B(\alpha_B^{\max} + \xi w_B) \\ &= f_B(\alpha_B^{\max}) - \frac{1}{2}(\xi w_B)^T Q_B(\xi w_B) \\ &= f_B(\alpha_B^{\max}) - \left(\frac{1}{2} w_B^T Q_B w_B\right)\xi^2 \quad .\end{aligned}$$

Now it is possible to calculate the gain as

$$\begin{aligned}f_B(\alpha_B^*) - f_B(\alpha_B) &= \tilde{f}_B(\mu_B^* - \mu_B^{\max}) - \tilde{f}_B(0 - \mu_B^{\max}) \\ &= \left(\frac{1}{2} w_B^T Q_B w_B\right)((\mu_B^{\max})^2 - (\mu_B^* - \mu_B^{\max})^2) \\ &= \left(\frac{1}{2} w_B^T Q_B w_B\right)(\mu_B^*(2\mu_B^{\max} - \mu_B^*)) \\ &= \frac{1}{2}(Q_{b_1b_1} + Q_{b_2b_2} - 2y_{b_1}y_{b_2}Q_{b_1b_2})(\mu_B^*(2\mu_B^{\max} - \mu_B^*)) \quad . \qquad (3)\end{aligned}$$

The diagonal matrix entries Q_{ii} needed for the calculation can be precomputed before the decomposition loop starts using time and memory linear in ℓ. Thus, knowing only the derivatives (1), C, y_B, and $Q_{b_1b_2}$ (and the precomputed diagonal entries) makes it possible to compute the gain in f. Usually in an SVM implementation the derivatives are already at hand because they are required for the optimality test in the stopping criterion. Of course we have access to the labels and the regularization parameter C. The only remaining quantity needed is $Q_{b_1b_2}$, which unfortunately requires evaluating the kernel function.

2.3 Maximum-Gain Working Set Selection

Now, a straightforward working set selection strategy is to look at all $\ell(\ell-1)/2$ possible variable pairs, to evaluate the gain (3) for every one of them, and to select the best, that is, the one with maximum gain. It can be expected that this greedy selection policy leads to very fast convergence measured in number of iterations needed. However, it has two major drawbacks making it advisable only for very small problems: looking at all possible pairs requires the knowledge of the complete matrix Q. As Q is in general too big to fit into the working memory, expensive kernel function evaluations become necessary. Further, the evaluation of all possible pairs scales quadratically with the number of training examples.

Fortunately, modern SVM implementations use a considerable amount of working memory as a cache for the rows of Q computed in recent iterations. In all cases, this cache contains the two rows corresponding to the working set chosen in the most recent iteration, because they were needed for

the gradient update (2). This fact leads to the following maximum-gain working pair selection (MG) algorithm:

Maximum-Gain Working Set Selection in step t

if $t = 1$ **then**
  select arbitrary working set $B^{(1)} = \{b_1, b_2\}, y_{b_1} \neq y_{b_2}$
else
  select pair $B^{(t)} \leftarrow \operatorname{argmax}_{B=\{b_1,b_2\} \mid b_1 \in B^{(t-1)}, b_2 \in \{1,\ldots,\ell\}} g_B(\alpha)$

In the first iteration, usually no cached matrix rows are available. Thus, an arbitrary working set $B^{(1)} = \{b_1, b_2\}$ fulfilling $y_{b_1} \neq y_{b_2}$ is chosen. In all following iterations, given the previous working set $B^{(t-1)} = \{b_1, b_2\}$, the gain of all combinations $\{b_1, b\}$ and $\{b_2, b\}$ ($b \in \{1, \ldots, \ell\}$) is evaluated and the best one is selected.

The complexity of the working set selection is linear in the number of training examples. It is important to note that the algorithm uses second order information from the matrix cache. These information are ignored by all existing working set selection strategies, albeit they are available for free, that is, without spending any additional computational effort. This situation is comparable to the improvement of using the gradient for the analytical solution of the sub-problem in the SMO algorithm. Although the algorithm by Fan et al. (2005) considers second order information, these are in general not available from the matrix cache.

The maximum gain working pair selection can immediately be generalized to the class of *maximum-gain working set selection* algorithms (see Section 2.5). Under this term we want to subsume all working set selection strategies choosing variables according to a greedy policy with respect to the functional gain computed using cached matrix rows. In the following, we restrict ourselves to the selection of pairs of variables as working sets.

In some SVM implementations, such as LIBSVM, the computation of the stopping condition is done using information provided during the working set selection. LIBSVM's MVP algorithm stops if the sum of the violations of the pair is less than a predefined constant ε. The simplest way to implement a roughly comparable stopping condition in MG is to stop if the value μ_B^* defining the length of the constrained step is smaller than ε.

It is worth noting that the MG algorithm does not depend on the caching strategy. The only requirement for the algorithm to efficiently profit from the kernel cache is that the cache always contains the two rows of the matrix Q that correspond to the previous working set. This should be fulfilled by every efficient caching algorithm, because recently active variables have a high probability to be in the working set again in future iterations. That is, the MG algorithm does not require a change of the caching strategy. Instead, it improves the suitability of all caching strategies that at least store the information most recently used.

2.4 Hybrid Maximum-Gain Working Set Selection

The MG algorithm can be used in combination with other methods. In order to inherit the convergence properties from MVP we introduce the *hybrid maximum gain* (HMG) working set selection algorithm. The algorithm is defined as follows:

Hybrid Maximum-Gain Working Set Selection in step t, $0 < \eta \ll 1$

```
1 if t = 1 then
2     select arbitrary working set B^(1) = {b_1, b_2}, y_{b_1} ≠ y_{b_2}
3 else
4     if ∀i ∈ B^(t-1) : α_i ≤ η·C ∨ α_i ≥ (1-η)·C then
5         select B^(t) according to MVP
6     else
7         select pair B^(t) ← argmax_{B={b_1,b_2} | b_1∈B^(t-1), b_2∈{1,...,ℓ}} g_B(α)
```

In the first iteration, usually no cached matrix rows are available. Thus, an arbitrary working set $\{b_1, b_2\}$ fulfilling $y_{b_1} \neq y_{b_2}$ is selected. If in iteration $t > 1$ both variables indexed by the previous working set $B^{(t-1)} = \{b_1, b_2\}$ are no more than $\eta \cdot C$, $0 < \eta \ll 1$, from the bounds, then the MVP algorithm is used. Otherwise the working set is selected according to MG. Figure 2 illustrates the HMG decision rule. The stopping condition tested by the decomposition algorithm is the one

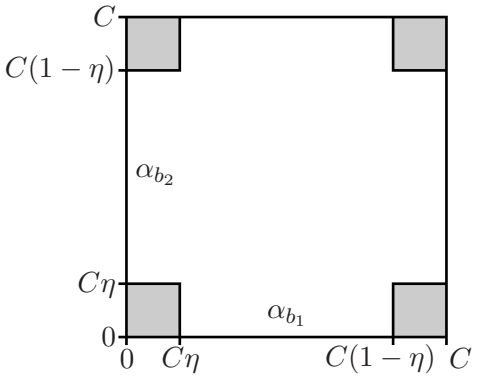


Figure 2: Illustration of the HMG algorithm. The plain defined by the previous working set $B^{(t-1)} = \{b_1, b_2\}$ is drawn. If the algorithm ended up in one of the gray corners then the MVP algorithm is used in iteration t.

from the working set selection algorithm used in the current iteration. That is, the decomposition algorithm stops if $y_{b_1} \frac{\partial f}{\partial \alpha_{b_1}}(\alpha) - y_{b_2} \frac{\partial f}{\partial \alpha_{b_2}}(\alpha)$ or μ_B^* falls below the threshold ε depending on whether MVP or MG has been selected.

The HMG algorithm is a combination of MG and MVP using MVP only in special situations. In our experiments, we set $\eta = 10^{-8}$. This choice is arbitrary and makes no difference to $\eta = 0$ in nearly all cases. In practice, in almost all iterations MG will be active. Thus, this algorithm inherits the speed of the MG algorithm. It is important to note that η is not a parameter influencing the convergence speed (as long as the parameter is small) and is therefore not subject to tuning.

The technical modification ensures the convergence of the algorithm. This is formally expressed by the following theorem.

Theorem 1 *We consider problem $\mathcal{P}$. Let $(\alpha^{(t)})_{t\in\mathbb{N}}$ be a sequence of feasible points produced by the decomposition algorithm using the HMG policy. Then, the limit point of every convergent subsequence is optimal for $\mathcal{P}$.*

The proof can be found in Section 3.

2.5 Generalization of the Algorithm

In the following, we discuss some of the potential variants of the basic MG or HMG algorithm.

It is possible to use a larger set of precomputed rows, say, 10, for the working set selection. In the extreme case we can run through all cached rows of Q. Then the working set selection algorithm becomes quite time consuming in comparison to the gradient update. As the number of iterations does not decrease accordingly, as we observed in real world applications, we recommend to use only the two rows of the matrix from the previous working set. We refer to Section 4.6 for a comparison.

A small change speeding up the working set selection is fixing one element of the working set in every iteration. When alternating the fixed position, every element is used two times successively. Only $\ell - 2$ pairs have to be evaluated in every iteration. Though leading to more iterations this policy can speed up the MG algorithm for small problems (see Section 4.6).

The algorithm can be extended to compute the gain for tuples of size $q > 2$. It is a severe disadvantage that such sub-problems can not be solved analytically and an iterative solver has to be used for the solution of the sub-problem. Note that this becomes necessary also for every gain computation during the working set selection. To keep the complexity of the working set selection linear in ℓ only one element new to the working set can be evaluated. Due to this limitation this method becomes even more cache friendly. The enlarged working set size may decrease the number of iterations required, but at the cost of the usage of an iterative solver. This should increase the speed of the SVM algorithm only on large problems with extremely complicated kernels, where the kernel matrix does not fit into the cache and the kernel evaluations in every iteration take much longer than the working set selection.

3. Convergence of the Algorithms

In this section we discuss the convergence properties of the decomposition algorithm using MG and HMG working set selection. First, we give basic definitions and prove a geometrical criterion for optimality. Then, as a motivation and a merely theoretical result, we show some properties of the gain function and prove that the greedy strategy w.r.t. the gain converges to an optimum. Returning to our algorithm, we give a counter example proving that there exist scenarios where pure MG looking at pairs of variables may stop after finitely many iterations without reaching an optimum. Finally, we prove that HMG converges to an optimum.

3.1 Prerequisites

In our convergence analysis, we consider the limit $\varepsilon \to 0$, that is, the algorithms only stop if the quantities checked in the stopping conditions vanish. We will discuss the convergence of the infinite sequence $(\alpha^{(t)})_{t\in\mathbb{N}}$ produced by the decomposition algorithm. If the decomposition algorithm stops in some iteration t_0 at $\alpha^{(t_0-1)}$, then by convention we set $\alpha^{(t)} \leftarrow \alpha^{(t_0-1)}$ for all $t \geq t_0$. The definition of convergence used here directly implies the convergence in finite time to a solution arbitrarily

close to the optimum considered in other proofs (Keerthi and Gilbert, 2002; Takahashi and Nishi, 2005).

The Bolzano-Weierstraß property states that every sequence on a compact set contains a convergent sub-sequence. Because of the compactness of $\mathcal{R}(\mathcal{P})$ the sequence $(\alpha^{(t)})_{t\in\mathbb{N}}$ always contains a convergent sub-sequence denoted $(\alpha^{(t)})_{t\in S}$ with limit point $\alpha^{(\infty)}$. From the construction of the decomposition algorithm it follows that the sequence $(f(\alpha^{(t)}))_{t\in\mathbb{N}}$ increases monotonically. The compactness of $\mathcal{R}(\mathcal{P})$ implies that it is bounded. It therefore converges and its limit $f(\alpha^{(\infty)})$ does not depend on the choice of the convergent sub-sequence. The gain sequence $g_{B^{(t)}}(\alpha^{(t-1)}) = f(\alpha^{(t)}) - f(\alpha^{(t-1)})$ is non-negative and converges to zero. It will be the aim of this section to prove that $\alpha^{(\infty)}$ is the maximizer of f within the feasible region $\mathcal{R}(\mathcal{P})$ if the decomposition algorithm is used with HMG working set selection.

List and Simon (2004) introduced the (technical) restriction that all principal 2×2 minors of Q have to be positive definite. For Theorem 4, which was shown by List and Simon (2004), and the proof of Lemma 9 (and thus of Theorem 1) we adopt this requirement. The assumption is not very restrictive because it does not require the whole matrix Q to be positive definite. If in contrast Q is indeed positive definite (for example for Gaussian kernels with distinct examples) then this property is inherited by the principal minors.[2]

If we fix any subset of variables of $\mathcal{P}$ at any feasible point $\alpha\in\mathcal{R}(\mathcal{P})$ then the resulting restricted problem is again of the form $\mathcal{P}$. By analytically solving the problem restricted to a working set B we can compute the gain $g_B(\alpha)$. The set $V_{\mathcal{P}} := \{\alpha\in\mathbb{R}^\ell \,|\, \langle y,\alpha\rangle = z\}$ is the hyperplane defined by the equality constraint. It contains the compact convex feasible region $\mathcal{R}(\mathcal{P})$. The set of possible working sets is denoted by $\mathcal{B}(\mathcal{P}) := \big\{B \,\big|\, B\subset\{1,\ldots,\ell\}, |B|=2\big\}$. We call two working sets $B_1, B_2\in\mathcal{B}(\mathcal{P})$ related if $B_1\cap B_2\neq\emptyset$. With a working set $B=\{b_1,b_2\}$, $b_1<b_2$, we associate the vector w_B with components $(w_B)_{b_1}=1$, $(w_B)_{b_2}=-y_{b_1}y_{b_2}$ and $(w_B)_i=0$ otherwise. It points into the direction in which α can be modified using the working set.

If a feasible point α is not optimal then there exists a working set B on which it can be improved. This simply follows from the fact that there are working set selection policies based on which the decomposition algorithm is known to converge (Lin, 2001; Keerthi and Gilbert, 2002; Fan et al., 2005; Takahashi and Nishi, 2005). In this case the gain $g_B(\alpha)$ is positive.

Next, we give a simple geometrically inspired criterion for the optimality of a solution.

Lemma 2 *We consider the problem $\mathcal{P}$. For a feasible point α_0 the following conditions are equivalent:*

1. *α_0 is optimal.*
2. *$\langle(\alpha-\alpha_0),\nabla f(\alpha_0)\rangle\le 0$ for all $\alpha\in\mathcal{R}(\mathcal{P})$.*
3. *$\langle\mu\cdot w_B,\nabla f(\alpha_0)\rangle\le 0$ for all $\mu\in\mathbb{R}$, $B\in\mathcal{B}(\mathcal{P})$ fulfilling $\alpha_0+\mu\cdot w_B\in\mathcal{R}(\mathcal{P})$.*

Proof The proof is organized as $(1)\Rightarrow(2)\Rightarrow(3)\Rightarrow(1)$. We consider the Taylor expansion

$$f(\alpha) = f(\alpha_0) + \langle(\alpha-\alpha_0),\nabla f(\alpha_0)\rangle - \frac{1}{2}(\alpha-\alpha_0)^T Q(\alpha-\alpha_0)$$

2. Usually there is only a zero set of training data sets that violate this condition. This is obviously true if the input space is an open subset of some $\mathbb{R}^n$ and the distribution generating the training data has a density w.r.t. the Lebesgue measure.

of f in α_0. Let us assume that (2) does not hold, that is, there exists $\alpha \in \mathcal{R}(\mathcal{P})$ such that $q := \langle(\alpha - \alpha_0), \nabla f(\alpha_0)\rangle > 0$. From the convexity of $\mathcal{R}(\mathcal{P})$ it follows that $\alpha_\mu := \mu\alpha + (1-\mu)\alpha_0 \in \mathcal{R}(\mathcal{P})$ for $\mu \in [0,1]$. We further set $r := \frac{1}{2}(\alpha - \alpha_0)^T Q(\alpha - \alpha_0) \geq 0$ and have $\langle(\alpha_\mu - \alpha_0), \nabla f(\alpha_0)\rangle = \mu q$ and $\frac{1}{2}(\alpha_\mu - \alpha_0)^T Q(\alpha_\mu - \alpha_0) = \mu^2 r$. We can chose $\mu_0 \in (0,1]$ fulfilling $\mu_0 q > \mu_0^2 r$. Then it follows

$$\begin{aligned} f(\alpha_{\mu_0}) &= f(\alpha_0) + \langle(\alpha_{\mu_0} - \alpha_0), \nabla f(\alpha_0)\rangle - \frac{1}{2}(\alpha_{\mu_0} - \alpha_0)^T Q(\alpha_{\mu_0} - \alpha_0) \\ &= f(\alpha_0) + \mu_0 q - \mu_0^2 r \\ &> f(\alpha_0) \ , \end{aligned}$$

which proves that α_0 is not optimal. Thus (1) implies (2). Of course (3) follows from (2). Now we assume α_0 is not optimal. From the fact that there are working set selection policies for which the decomposition algorithm converges to an optimum it follows that there exists a working set B on which α_0 can be improved, which means $g_B(\alpha_0) > 0$. Let α_1 denote the optimum on the feasible line segment within $\mathcal{R}(\mathcal{P})$ written in the form $\alpha_1 = \alpha_0 + \mu \cdot w_B$. Using the Taylor expansion above at α_1 and the positive semi-definiteness of Q we get

$$\begin{aligned} f(\alpha_1) = f(\alpha_0) + \langle(\alpha_1 - \alpha_0), \nabla f(\alpha_0)\rangle - \frac{1}{2}(\alpha_1 - \alpha_0)^T Q(\alpha_1 - \alpha_0) &> f(\alpha_0) \\ \Leftrightarrow \ \langle(\alpha_1 - \alpha_0), \nabla f(\alpha_0)\rangle &> \frac{1}{2}(\alpha_1 - \alpha_0)^T Q(\alpha_1 - \alpha_0) \geq 0 \\ \Rightarrow \ \langle \mu \cdot w_B, \nabla f(\alpha_0)\rangle &> 0 \end{aligned}$$

showing that (3) implies (1). ■

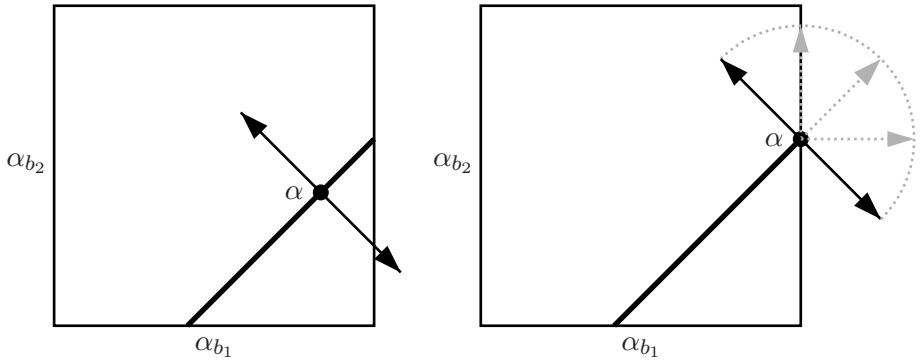


Figure 3: This figure illustrates the optimality condition given in Lemma 2 for one working set. On the left the case of two free variables α_{b_1} and α_{b_2} is shown, while on the right the variable α_{b_1} is at the bound C. The fat lines represent the feasible region for the 2-dimensional problem induced by the working set. The arrows show the possible gradient directions not violating the optimality conditions.

3.2 Convergence of the Greedy Policy

Before we look at the convergence of the MG algorithm, we use Theorem 4 by List and Simon (2004) to prove the convergence of the decomposition algorithm using the greedy policy with respect to the gain for the working set selection. For this purpose we will need the concept of a 2-sparse witness of suboptimality.

Definition 3 (2-sparse witness of suboptimality) *A family of functions* $(C_B)_{B\in\mathcal{B}(\mathcal{P})}$

$$C_B : \mathcal{R}(\mathcal{P}) \to \mathbb{R}^{\geq 0}$$

fulfilling the conditions
(C1) *C_B is continuous,*
(C2) *if α is optimal for $\mathcal{P}_{B,\alpha}$, then $C_B(\alpha) = 0$, and*
(C3) *if a feasible point α is not optimal for $\mathcal{P}$, then there exists B such that $C_B(\alpha) > 0$*
is called a 2-sparse witness of suboptimality *(List and Simon, 2004).*

Every 2-sparse witness of suboptimality induces a working set selection algorithm by

$$B^{(t)} := \operatorname*{argmax}_{B\in\mathcal{B}(\mathcal{P})} \left(C_B(\alpha^{(t-1)}) \right) \quad .$$

List and Simon (2004) call this the induced decomposition algorithm. Now we can quote a general convergence theorem for decomposition methods induced by a 2-sparse witness of suboptimality:[3]

Theorem 4 (List and Simon, 2004) *We consider the problem $\mathcal{P}$ and a 2-sparse witness of suboptimality $(C_B)_{B\in\mathcal{B}(\mathcal{P})}$. Let $(\alpha^{(t)})_{t\in\mathbb{N}}$ denote a sequence of feasible points generated by the decomposition method induced by (C_B) and $(\alpha^{(t)})_{t\in S}$ a convergent sub-sequence with limit point $\alpha^{(\infty)}$. Then, the limit point is optimal for $\mathcal{P}$.*

The following lemma allows for the application of this theorem.

Lemma 5 *The family of functions $(g_B)_{B\in\mathcal{B}(\mathcal{P})}$ is a 2-sparse witness of suboptimality.*

Proof Property **(C2)** is fulfilled directly per construction. Property **(C3)** follows from the fact that there exist working set selection strategies such that the decomposition method converges (Lin, 2001; Takahashi and Nishi, 2005; List and Simon, 2005). It is left to prove property **(C1)**. We fix a working set $B = \{b_1, b_2\}$ and the corresponding direction vector w_B. Choosing the working set B is equivalent to restricting the problem to this direction. We define the affine linear function

$$\varphi : V_{\mathcal{P}} \to \mathbb{R} \ , \qquad \alpha \mapsto \left.\frac{\partial}{\partial\mu}\right|_{\mu=0} f(\alpha + \mu w_B)$$

and the ($(\ell-2)$-dimensional) hyperplane $H := \{\alpha \in V_{\mathcal{P}} \,|\, \varphi(\alpha) = 0\}$ within the ($(\ell-1)$-dimensional) vector space $V_{\mathcal{P}}$. This set always forms a hyperplane because $Qw_B \neq 0$ is guaranteed by the assumption that all 2×2 minors or Q are positive definite. This hyperplane contains the optima of f restricted to the lines $\alpha + \mathbb{R}\cdot w_B$ considering the equality constraint but not the box constraints. We

3. Theorem 1 in List and Simon (2004) is more general as it is not restricted to working sets of size two.

introduce the map π_H projecting $V_\mathcal{P}$ onto H along w_B, that is the projection mapping the whole line $\alpha + \mathbb{R} \cdot w_B$ onto its unique intersection with H. The hyperplane H contains the compact subset

$$\tilde{H} := \left\{ \alpha \in H \,\middle|\, \alpha + \mathbb{R} \cdot w_B \cap \mathcal{R}(\mathcal{P}) \neq \emptyset \right\} = \pi_H(\mathcal{R}(\mathcal{P}))$$

on which we define the function

$$\delta : \tilde{H} \to \mathbb{R} \ , \qquad \alpha \mapsto \operatorname*{argmin}_{\{\mu \in \mathbb{R} \,|\, \alpha + \mu w_B \in \mathcal{R}(\mathcal{P})\}} |\mu| \ .$$

The term $\delta(\pi_H(\alpha))w_B$ describes the shortest vector moving $\alpha \in \tilde{H}$ to the feasible region on the line along w_B. On $\tilde{H} \setminus \mathcal{R}(\mathcal{P})$ it parameterizes the boundary of the feasible region (see Figure 4). These properties enable us to describe the optimal solution of the sub-problem induced by the working set. Starting from $\alpha \in \mathcal{R}(\mathcal{P})$ the optimum $\pi_H(\alpha)$ is found neglecting the box constraints. In case this point is not feasible it is clipped to the feasible region by moving it by $\delta(\pi_H(\alpha))w_B$. Thus, per construction it holds $g_B(\alpha) = f(\pi_H(\alpha) + \delta(\pi_H(\alpha))w_B) - f(\alpha)$. The important point here is that convexity and compactness of $\mathcal{R}(\mathcal{P})$ guarantee that δ is well-defined and continuous. We conclude that g_B is continuous as it is a concatenation of continuous functions. ■

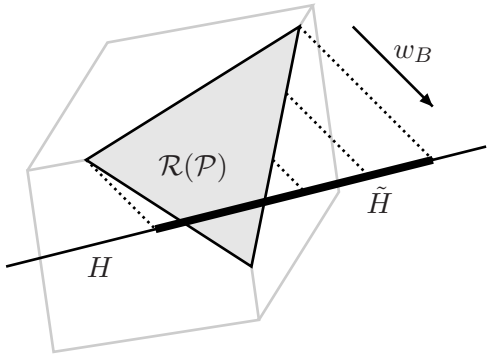


Figure 4: The feasible region $\mathcal{R}(\mathcal{P})$ within the $(\ell-1)$-dimensional vector space $V_\mathcal{P}$ is illustrated. The ℓ-dimensional box constraints are indicated in light gray. The thin line represents the hyperplane H containing the compact subset $\tilde{H}$ drawn as a fat line segment. The lengths of the dotted lines indicate the absolute values of the function δ on $\tilde{H}$. The function δ vanishes within the intersection of $\tilde{H}$ and $\mathcal{R}(\mathcal{P})$.

Corollary 6 *We consider problem $\mathcal{P}$. Let $(\alpha^{(t)})_{t\in\mathbb{N}}$ be a sequence of feasible points produced by the decomposition algorithm using the greedy working set selection policy. Then, every limit point $\alpha^{(\infty)}$ of a converging sub-sequence $(\alpha^{(t)})_{t\in S}$ is optimal for $\mathcal{P}$.*

Proof For a feasible point α and a working set B the achievable gain is computed as $g_B(\alpha)$. Thus, the decomposition method induced by the family (g_B) selects the working set resulting in the maximum gain, which is exactly the greedy policy. Lemma 5 and Theorem 4 complete the proof. ■

It is straightforward to use the more general version of the theorem from List and Simon (2004) to extend the construction for working sets limited in size to some $q > 2$.

3.3 Convergence of the MG Algorithm

Theorem 7 *Given a feasible point $\alpha^{(t)}$ for $\mathcal{P}$ and a previous working set $B^{(t-1)}$ of size 2 as a starting point. Then, in general, the MG algorithm may get stuck, that means, it may stop after finitely many iterations without reaching an optimum.*

Proof As a proof we give a counter example. The MG algorithm may get stuck before reaching the optimum because it is restricted to reselect one element of the previous working set. For $\ell < 4$ this poses no restriction. Thus, to find a counter example, we have to use some $\ell \geq 4$. Indeed, using four training examples is already sufficient. We consider $\mathcal{P}$ for $\ell = 4$ with the values

$$Q = \begin{pmatrix} 2 & \sqrt{3} & 1 & \sqrt{3} \\ \sqrt{3} & 4 & \sqrt{3} & 3 \\ 1 & \sqrt{3} & 2 & \sqrt{3} \\ \sqrt{3} & 3 & \sqrt{3} & 4 \end{pmatrix} , \qquad C = \frac{1}{10} , \qquad y = \begin{pmatrix} -1 \\ -1 \\ +1 \\ +1 \end{pmatrix} .$$

The matrix Q is positive definite with eigenvalues $(9,1,1,1)$. We assume the previous working set to be $B^{(1)} = \{1,3\}$ resulting in the point $\alpha^{(1)} = (C,0,C,0)^T$. Note that this is the result of the first iteration starting from $\alpha^{(0)} = (0,0,0,0)$ greedily choosing the working set $B^{(1)} = \{1,3\}$. It is thus possible that the decomposition algorithm reaches this state in the SVM context. We compute the gradient

$$\nabla f(\alpha^{(1)}) = \mathbf{1} - Q\alpha^{(1)} = (\frac{7}{10}, 1 - \frac{\sqrt{3}}{5}, \frac{7}{10}, 1 - \frac{\sqrt{3}}{5})^T \approx (0.7, 0.65, 0.7, 0.65)^T$$

(which is orthogonal to y). Using Lemma 2 we compute that the sub-problems defined by all working sets with exception $B = \{2,4\}$ are already optimal. The working sets $B^{(1)}$ and B have no element in common. Thus, the maximum gain working pair algorithm cannot select $B^{(2)} = B$ and gets stuck although the point $\alpha^{(1)}$ is not optimal. From Figure 5 we can see that the same example works for all points on the edge $\alpha^{(1)} = (C,\nu,C,\nu)$ for $\nu \in [0,C)$. Lemma 8 states that indeed the edges of the octahedron are the only candidates for the MG algorithm to get stuck. ■

3.4 Convergence of the HMG Algorithm

The above result makes it advisable to use a different algorithm whenever MG is endangered to get stuck. For this purpose the HMG algorithm was designed. In this section we prove that this modification indeed guarantees convergence to an optimum.

The following lemma deals with the specific property of the MG algorithm to reselect one element of the working set, that is, to select related working sets in consecutive iterations. It is a major building block in the proof of the main result stated in Theorem 1.

Lemma 8 *We consider $\mathcal{P}$, a current non-optimal feasible point α and a previous working set $B_1 = \{b_1, b_2\}$. If at least one of the variables α_{b_1} and α_{b_2} is free (not at the bounds 0 or C) then there exists a working set B_2 related to B_1 such that positive gain $g_{B_2}(\alpha) > 0$ can be achieved.*

Proof We show that no counter example exists. The idea is to reduce the number of possible scenarios to a finite number and to inspect each case individually.

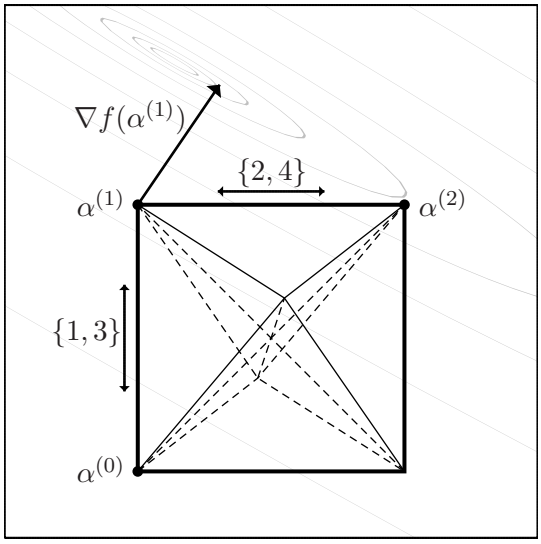


Figure 5: Illustration of the counter example. The feasible region $\mathcal{R}(\mathcal{P})$ forms an octahedron within the 3-dimensional space $V_{\mathcal{P}}$. The possible directions of movement using working sets of size 2 are parallel to the edges of the octahedron. The SVM algorithm starts in $\alpha^{(0)} = (0,0,0,0)^T$. During the first two iterations the greedy policy reaches the points $\alpha^{(1)} = (C,0,C,0)^T$ and $\alpha^{(2)} = (C,C,C,C)^T$. The MG algorithm gets stuck after the first iteration at $\alpha^{(1)}$. The plane spanned by the directions $(1,0,1,0)^T$ and $(0,1,0,1)^T$ defined by the working sets $\{1,3\}$ and $\{2,4\}$ respectively, is drawn. The gray lines are level sets of the target function f within this plane. In the point $\alpha^{(1)}$ the gradient $\nabla f(\alpha^{(1)})$ (which lies within the plane) has an angle of less than $\pi/2$ only with the horizontally drawn edge corresponding to the working set $\{2,4\}$.

For $\ell \leq 3$ the condition that B_1 and B_2 are related is no restriction and we are done. In the main part of the proof, we consider the 4-dimensional case and set $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ and $B_1 = \{1,2\}$ with free variable $\alpha_1 \in (0,C)$. In the end, we will reduce the general case to $\ell \leq 4$.

Let us have a look at potential counter examples. A feasible point α is a counter example if it is not optimal and does not allow for positive gain on any working set related to B_1. These conditions are equivalent to

$$g_B(\alpha) \begin{cases} = 0 & \text{for } B \neq \{3,4\} \\ > 0 & \text{for } B = \{3,4\} \end{cases} . \tag{4}$$

Looking at the six possible working sets B we observe from Lemma 2 that we have to distinguish three cases for sub-problems induced by the working sets:

- The current point α is at the bounds for a variable indexed by B and the points $\alpha + \mu \cdot w_B$ lie within $\mathcal{R}(\mathcal{P})$ only for $\mu \leq 0$. Then Lemma 2 states that α can only be optimal if $\langle w_B, \nabla f(\alpha) \rangle \geq 0$.

- The current point α is at the bounds for a variable indexed by B and the points $\alpha + \mu \cdot w_B$ lie within $\mathcal{R}(\mathcal{P})$ only for $\mu \geq 0$. Then Lemma 2 states that α can only be optimal if $\langle w_B, \nabla f(\alpha) \rangle \leq 0$.
- The current point α is not at the bounds for both variables indexed by B. Thus, there are positive and negative values for μ such that $\alpha + \mu \cdot w_B$ lies within $\mathcal{R}(\mathcal{P})$. From Lemma 2 it follows that α can only be optimal if $\langle w_B, \nabla f(\alpha) \rangle = 0$.

We conclude that the signs (< 0, $= 0$, or > 0) of the expressions

$$\langle w_B, \nabla f(\alpha) \rangle = \frac{\partial f}{\partial \alpha_{b'_0}}(\alpha) - y_{b'_0} y_{b'_1} \frac{\partial f}{\partial \alpha_{b'_1}}(\alpha) \quad \text{for} \quad B = \{b'_0, b'_1\} \subset \{1,2,3,4\} \tag{5}$$

and the knowledge about which variables are at which bound are sufficient for the optimality check. Further, it is not important which exact value a free variable takes. The possible combinations of vectors w_B occurring in equation (5) are generated by the label vector y. Combining these insights, we define the maps

$$\text{sign} : \mathbb{R} \to \{-1, 0, +1\}, \qquad x \mapsto \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ +1 & \text{if } x > 0 \end{cases}$$

$$\text{bound} : [0, C] \to \{0, \frac{C}{2}, C\}, \qquad x \mapsto \begin{cases} 0 & \text{if } x = 0 \\ \frac{C}{2} & \text{if } 0 < x < C \\ C & \text{if } x = C \end{cases}$$

and a mapping of all possible counter examples onto a finite number of cases

$$\Psi : \mathcal{R}(\mathcal{P}) \times \mathbb{R}^4 \times \{-1,+1\}^4 \to \{-1,0,+1\}^6 \times \{0, C/2, C\}^3 \times \{-1,+1\}^4,$$
$$\begin{pmatrix} \alpha \\ \nabla f(\alpha) \\ y \end{pmatrix} \mapsto \begin{pmatrix} \text{bound}(\alpha_i),\ i \in \{2,3,4\} \\ \text{sign}(\langle w_B, \nabla f(\alpha) \rangle),\ B \in \mathcal{B}(\mathcal{P}) \\ y \end{pmatrix} .$$

A possible counter example is fully determined by a candidate point $\alpha \in \mathcal{R}(\mathcal{P})$, the gradient $G = \nabla f(\alpha)$, and the label vector y. As the parameters of problem $\mathcal{P}$ are not fixed here, the equality constraint can be ignored, because every point fulfilling the box constraints can be made feasible by shifting the equality constraint hyperplane. The relation $\Psi(\alpha, G, y) = \Psi(\tilde{\alpha}, \tilde{G}, \tilde{y})$ divides the pre-image of Ψ into equivalence classes. For each element of one equivalence class the check of condition (4) using Lemma 2 is the same. Formally, we have

$$\begin{aligned} &\Psi(\alpha, G, y) = \Psi(\tilde{\alpha}, \tilde{G}, \tilde{y}) \\ \Rightarrow &\text{condition (4) holds for } (\alpha, G, y) \text{ if and only if condition (4) holds for } (\tilde{\alpha}, \tilde{G}, \tilde{y}) \ . \end{aligned}$$

It follows that any finite set containing representatives of all the non-empty equivalence classes is sufficient to check for the existence of a counter example in the infinite pre-image of Ψ.[4] The checking can be automated using a computer program. A suitable program can be downloaded from the online appendix

4. The function Ψ itself helps designing such a set of representatives of the non-empty classes. For every fixed α and y the set $\{-4, -3, \ldots, 3, 4\}^4 \subset \mathbb{R}^4$ is sufficient to generate all possible combinations of $\text{sign}(\langle w_B, \nabla f(\alpha) \rangle)$, $B \in \mathcal{B}(\mathcal{P})$.

`http://www.neuroinformatik.ruhr-uni-bochum.de/PEOPLE/igel/wss/ .`

The outcome of the program is that there exists no counter example.

It is left to prove the lemma for $\ell > 4$. This case can be reduced to the situations already considered. Because α is not optimal there exists a working set B_* with $g_{B_*}(\alpha) > 0$. The set $W := B_1 \cup B_*$ defines an at most 4-dimensional problem. The proof above shows that there exists a working set $B \subset W$ with the required properties. ■

Following List and Simon (2004), one can bound $\|\alpha^{(t)} - \alpha^{(t-1)}\|$ in terms of the gain:

Lemma 9 *We consider problem $\mathcal{P}$ and a sequence $(\alpha^{(t)})_{t\in\mathbb{N}}$ produced by the decomposition algorithm. Then, the sequence $\left(\|\alpha^{(t)} - \alpha^{(t-1)}\|\right)_{t\in\mathbb{N}}$ converges to* 0.

Proof The gain sequence $\left(g_{B^{(t)}}(\alpha^{(t-1)})\right)_{t\in\mathbb{N}}$ converges to zero as it is non-negative and its sum is bounded from above. The inequality

$$g_{B^{(t)}}(\alpha^{(t-1)}) \geq \frac{\sigma}{2}\|\alpha^{(t)} - \alpha^{(t-1)}\|^2 \quad \Leftrightarrow \quad \|\alpha^{(t)} - \alpha^{(t-1)}\| \leq \sqrt{\frac{2}{\sigma} g_{B^{(t)}}(\alpha^{(t-1)})}$$

holds, where σ denotes the minimal eigenvalue of the 2×2 minors of Q. By the technical assumption that all principal 2×2 minors of Q are positive definite we have $\sigma > 0$. ■

Before we can prove our main result we need the following lemma.

Lemma 10 *We consider problem $\mathcal{P}$, a sequence $(\alpha^{(t)})_{t\in\mathbb{N}}$ produced by the decomposition algorithm and the corresponding sequence of working sets $(B^{(t)})_{t\in\mathbb{N}}$. Let the index set $S \subset \mathbb{N}$ correspond to a convergent sub-sequence $(\alpha^{(t)})_{t\in S}$ with limit point $\alpha^{(\infty)}$.*

(*i*) *Let*

$$I := \left\{ B \in \mathcal{B}(\mathcal{P}) \;\middle|\; |\{t \in S \,|\, B^{(t)} = B\}| = \infty \right\}$$

denote the set of working sets selected infinitely often. Then, no gain can be achieved in the limit point $\alpha^{(\infty)}$ using working sets $B \in I$.

(*ii*) *Let*

$$R := \left\{ B \in \mathcal{B}(\mathcal{P}) \setminus I \;\middle|\; B \text{ is related to some } \tilde{B} \in I \right\} .$$

denote the set of working sets related to working sets in I. If the decomposition algorithm chooses MG working sets, no gain can be achieved in the limit point $\alpha^{(\infty)}$ using working sets $B \in R$.

This is obvious from equation (5) and the fact that these cases cover different as well as equal absolute values for all components together with all sign combinations. Hence, it is sufficient to look at these 9^4 gradient vectors, or in other words, the map from $\{-4,-3,\ldots,3,4\}^4$ to $\text{sign}(\langle w_B, \nabla f(\alpha)\rangle)$, $B \in \mathcal{B}(\mathcal{P})$ is surjective. The mapping of the 3^3 points $\alpha_1 = C/2$, $\alpha_i \in \{0, C/2, C\}$ for $i \in \{2,3,4\}$ onto $\text{bound}(\alpha_i)$, $i \in \{2,3,4\}$ is bijective. Of course the identity map $y \mapsto y$ of the 2^4 possible labels is bijective, too. Now we have constructed a set of $9^4 \cdot 3^3 \cdot 2^4 = 2,834,352$ cases. If there is no counter example among them, we know that no counter example exists in the whole infinite set.

Proof First, we prove (i). Let us assume these exists $B \in I$ on which positive gain can be achieved in the limit point $\alpha^{(\infty)}$. Then we have $\varepsilon := g_B(\alpha^{(\infty)}) > 0$. Because g_B is continuous there exists t_0 such that $g_B(\alpha^{(t)}) > \varepsilon/2$ for all $t \in S$, $t > t_0$. Because B is selected infinitely often it follows $f(\alpha^{(\infty)}) = \infty$. This is a contradiction to the fact that f is bounded on $\mathcal{R}(\mathcal{P})$.

To prove (ii) we define the index set $S_{(+1)} := \{t+1 \,|\, t \in S\}$. Using Lemma 9 we conclude that the sequence $(\alpha^{(t)})_{t \in S_{(+1)}}$ converges to $\alpha^{(\infty)}$. Let us assume that the limit point can be improved using a working set $B \in R$ resulting in $\varepsilon := g_B(\alpha^{(\infty)}) > 0$. Because g_B is continuous, there exists t_0 such that it holds $g_B(\alpha^{(t)}) > \varepsilon/2$ for all $t \in S_{(+1)}$, $t > t_0$. By the convergence of the sequence $(f(\alpha^{(t)}))_{t \in \mathbb{N}}$ we find t_1 such that for all $t > t_1$ it holds $g_{B^{(t)}}(\alpha^{(t-1)}) < \varepsilon/2$. The definition of I yields that there is a working set $\tilde{B} \in I$ related to B which is chosen in an iteration $t > \max\{t_0, t_1\}$, $t \in S$. Then in iteration $t+1 \in S_{(+1)}$ due to the MG policy the working set B (or another working set resulting in larger gain) is selected. We conclude that the gain achieved in iteration $t+1$ is greater and smaller than $\varepsilon/2$ at the same time which is a contradiction. Thus, $\alpha^{(\infty)}$ can not be improved using a working set $B \in R$. ■

Proof of Theorem 1 First we consider the case that the algorithm stops after finitely many iterations, that it, the sequence $(\alpha^{(t)})_{t \in \mathbb{N}}$ becomes stationary. We again distinguish two cases depending on the working set selection algorithm used just before the stopping condition is met. In case the MVP algorithm is used the stopping condition checks the exact KKT conditions. Thus, the point reached is optimal. Otherwise Lemma 8 asserts the optimality of the current feasible point.

For the analysis of the infinite case we distinguish two cases again. If the MG algorithm is used only finitely often then we can simply apply the convergence proof of SMO (Keerthi and Gilbert, 2002; Takahashi and Nishi, 2005). Otherwise we consider the set

$$T := \{t \in \mathbb{N} \,|\, \text{MG is used in iteration } t\}$$

of iterations in which the MG selection is used. The compactness of $\mathcal{R}(\mathcal{P})$ ensures the existence of a subset $S \subset T$ such that the sub-sequence $(\alpha^{(t)})_{t \in S}$ converges to some limit point $\alpha^{(\infty)}$. We define the sets

$$I := \left\{ B \in \mathcal{B}(\mathcal{P}) \;\middle|\; |\{t \in S \,|\, B^{(t)} = B\}| = \infty \right\}$$
$$R := \left\{ B \in \mathcal{B}(\mathcal{P}) \setminus I \;\middle|\; B \text{ is related to some } \tilde{B} \in I \right\}$$

and conclude from Lemma 10 that $\alpha^{(\infty)}$ can not be improved using working sets $B \in I \cup R$. Now let us assume that the limit point can be improved using any other working set. Then Lemma 8 states that all coordinates $\alpha_i^{(\infty)}$ for all $i \in B \in I$ are at the bounds. By the definition of the HMG algorithm this contradicts the assumption that the MG policy is used on the whole sequence $(\alpha^{(t)})_{t \in S}$. Thus, the limit point $\alpha^{(\infty)}$ is optimal for $\mathcal{P}$. From the strict increase and the convergence of the sequence $(f(\alpha^{(t)}))_{t \in \mathbb{N}}$ it follows that the limit point of every convergent sub-sequence $(\alpha^{(t)})_{t \in \tilde{S}}$ is optimal. ■

4. Experiments

The main purpose of our experiments is the comparison of different working set selection policies for large scale problems. This comparison focuses on SMO-like algorithms. The experiments were

carried out using LIBSVM (Chang and Lin, 2001). We implemented our HMG selection algorithm within LIBSVM to allow for a direct comparison. The modified source code of LIBSVM is given in the online appendix

`http://www.neuroinformatik.ruhr-uni-bochum.de/PEOPLE/igel/wss/ .`

Three SMO-like working set selection policies were compared, namely the LIBSVM-2.71 MVP algorithm, the second order LIBSVM-2.8 algorithm, and HMG working set selection.

To provide a baseline, we additionally compared these three algorithms to SVM^{light} (Joachims 1999) with a working set of size ten. In these experiments we used the same configuration and cache size as for the SMO-like algorithms. It is worth noting that neither the iteration count nor the influence of shrinking are comparable between LIBSVM and SVM^{light}. As we do not want to go into details on conceptual and implementation differences between the SVM packages, we only compared the plain runtime for the most basic case as it most likely occurs in applications. Still, as the implementations of the SMO-like algorithms and SVM^{light} differ, the results have to be interpreted with care.

We consider 1-norm soft margin SVM with radial Gaussian kernel functions

$$k_\sigma(x_i, x_j) := \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{6}$$

with kernel parameter σ and regularization parameter C. If not stated otherwise, the SVM was given 40 MB of working memory to store matrix rows. The accuracy of the stopping criterion was set to $\varepsilon = 0.001$. This value is small compared to the components of the gradient of the target function in the starting position $\alpha^{(0)}$. The shrinking heuristics for speeding up the SVM algorithm is turned on, see Section 4.4. Shrinking may cause the decomposition algorithm to require more iterations, but in most cases it considerably saves time. All of these settings correspond to the LIBSVM default configuration. If not stated otherwise, the hyperparameters C and σ were fixed to values giving well generalizing classifiers. These were determined by grid search optimizing the error on independent test data.

For the determination of the runtime of the algorithms we used a 1533 MHz AMD Athlon-XP system running Fedora Linux.

In most experiments we measured both the number of iterations needed and the runtime of the algorithm.[5] Although the runtime depends highly on implementation issues and programming skills, this quantity is in the end the most relevant in applications.

The comparison of the working set selection algorithms involves one major difficulty: The stopping criteria are different. It is in general not possible to compute the stopping criterion of one

5. We did not investigate the classification performance of the different approaches. As we are comparing algorithms converging to the exact solution of the SVM problem and the stopping criteria are chosen appropriately, we can expect that the machines trained using the different methods are equally well suited for classification. Due to the finite accuracy, the direction from which the optimum is approached, the exact path taken by the optimization, and the stopping criterion influence the value of the final solution. Thus, small differences in classification performance may occur between the algorithms. In contrast to the comparison of rough approximation methods or completely distinct types of classifiers, these effects are highly random, as they can depend on the presence of single input patterns or even on the order in which the training examples are presented. Another well known effect is that the classification accuracy measured on a test set does not necessarily increase with the solution accuracy. Thus, besides the prior knowledge that the differences are negligible, a comparisons of the classification accuracy is not meaningful in this context.

algorithm in another without additional computational effort. As the comparability of the runtime depends on an efficient implementation, each algorithm in the comparisons uses its own stopping criterion. The computation of the final value of the objective function reveals that the two stopping conditions are roughly comparable (see Section 4.2 and Table 2).

As discussed in Section 1.3, the order in which the training examples are presented influences the initial working set and thereby considerably the speed of optimization. Whether a certain ordering leads to fast or slow convergence is dependent on the working set selection method used. Therefore, we always consider the median over 10 independent trials with different initial working sets if not stated otherwise. In each trial the different algorithms started from the same working set. Whenever we claim that one algorithm requires less iterations or time these results are highly significant (two-tailed Wilcoxon rank sum test, $p < 0.001$).

Besides the overall performance of the working set selection strategies we investigated the influence of a variety of conditions. The experiments compared different values of the kernel parameter σ, the regularization parameter C, and the cache size. Further, we evaluated the performance with LIBSVM's shrinking algorithm turned on or off. Finally, we compared variants of the HMG strategy using different numbers of cached matrix rows for the gain computation.

4.1 Data Set Description

Four benchmark problems were considered. The 60,000 training examples of the MNIST handwritten digit database (LeCun et al., 1998) were split into two classes containing the digits $\{0,1,2,3,4\}$ and $\{5,6,7,8,9\}$, respectively. Every digit is represented as a 28×28 pixel array making up a 784 dimensional input space.

The next two data sets are available from the UCI repository (Blake and Merz, 1998). The spam-database contains 4,601 examples with 57 features extracted from e-mails. There are 1,813 positive examples (spam) and 2,788 negative ones. We transformed every feature to zero mean and unit variance. Because of the small training set, HMG is not likely to excel at this benchmark.

The connect-4 opening database contains 67,557 game states of the connect-4 game after 8 moves together with the labels 'win', 'loose', or 'draw'. For binary classification the 'draw' examples were removed resulting in 61,108 data points. Every situation was transformed into a 42-dimensional vector containing the entries 1, 0, or -1 for the first player, no player, or the second player occupying the corresponding field, respectively. The representation is sparse as in every vector only 8 components are non-zero. The data were split roughly into two halves making up training and test data. For the experiments only the training data were used.

The face data set contains 20,000 positive and 200,000 negative training examples. Every example originates from the comparison of two face images. Two pictures of the same person were compared to generate positive examples, while comparisons of pictures of different persons make up negative examples. The face comparison is based on 89 similarity features. These real-world data were provided by the Viisage Technology AG and are not available to the public.

Training an SVM using the large data sets face and MNIST takes very long. Therefore these two problems were not considered in all experiments.

We determined appropriate values for σ and C for each benchmark problem, see Table 1. We did coarse grid searches. The parameter combinations resulting in the smallest errors on corresponding test sets were chosen.

We want to pay special attention to the size of the kernel matrices in comparison to the cache size (see Table 1). The data sets cover a wide range of kernel matrix sizes which fit into the cache by nearly 50 % to only 0.02 %. It is a hopeless approach to adapt the cache size in order to fit larger parts of the kernel matrix into working memory. Because the space requirement for the kernel matrix grows quadratically with ℓ, large scale real world problems exceed any physically available cache.

data set	dim.	ℓ	cache	σ	C	SV	BSV
spam-database	57	4,601	47.2 %	10	50	18.5 %	11.7 %
connect-4	42	30,555	1.07 %	1.5	4.5	27.0 %	7.7 %
MNIST	784	60,000	0.28 %	3,500	50	10.5 %	4.6 %
face	89	220,000	0.02 %	3	5	2.6 %	1.2 %

Table 1: SVM parameters used in the comparison together with solution statistics. The column "dim." gives the input space dimension while ℓ is the number of training examples. The "cache"-column shows how much of the kernel matrix fits into the kernel cache. The fractions of support vectors and bounded support vectors are denoted by "SV" and "BSV". These percentage values might slightly differ between the algorithms because of the finite accuracy of the solutions.

4.2 Comparison of Working Set Selection Strategies

We trained SVMs on all data sets presented in the previous section. We monitored the number of iterations and the time until the stopping criterion was met. The results are shown in Table 2. The final target function values $f(\alpha^*)$ are also presented to prove the comparability of the stopping criteria (for the starting state it holds $f(\alpha^{(0)}) = 0$). Indeed, the final values are very close and which algorithm is most accurate depends on the problem.

It becomes clear from the experiments that the LIBSVM-2.71 algorithm performs worst. This is no surprise because it does not take second order information into account. In the following we will concentrate on the comparison of the second order algorithms LIBSVM-2.8 and HMG.

As the smallest problem considered the spam-database consists of 4,601 training examples. The matrix Q requires about 81 MB of working memory. The cache size of 40 MB should be sufficient when using the shrinking technique. The LIBSVM-2.8 algorithm profits from the fact that the kernel matrix fits into the cache after the first shrinking event. It takes less iterations and (in the mean) the same time per iteration as the HMG algorithm and is thus the fastest in the end.

In the connect-4 problem the kernel matrix does not fit into the cache even if shrinking is used to reduce the problem size. Thus, even in late iterations kernel evaluations can occur. Here, HMG outperforms the old and the new LIBSVM algorithm. This situation is even more pronounced for the MNIST data set and the face problem. Only a small fraction of Q fit into the cache making expensive kernel evaluations necessary. Note that for all of these large problems the LIBSVM-2.8 algorithm minimizes the number of iterations while HMG minimizes the training time. The HMG algorithm is the fastest on the three large scale problems, because it makes use of the kernel cache more efficiently.

data set (ℓ)	algorithm	iterations	runtime	$f(\alpha^*)$
spam-database (4,601)	LIBSVM-2.71	36,610	11.21 s	27,019.138
	LIBSVM-2.8	**9,228**	**8.44** s	27,019.140
	HMG	10,563	9.17 s	27,019.140
connect-4 (30,555)	LIBSVM-2.71	65,167	916 s	13,557.542
	LIBSVM-2.8	**45,504**	734 s	13,557.542
	HMG	50,281	**633 s**	13,557.536
MNIST (60,000)	LIBSVM-2.71	185,162	13,657 s	143,199.142
	LIBSVM-2.8	**110,441**	9,957 s	143,199.146
	HMG	152,873	**7,485 s**	143,199.160
face (220,000)	LIBSVM-2.71	37,137	14,239 s	15,812.666
	LIBSVM-2.8	**32,783**	14,025 s	15,812.666
	HMG	42,303	**11,278 s**	15,812.664

Table 2: Comparison of the number of iterations of the decomposition algorithm and training times for the different working set selection approaches. In each case the best value is highlighted. The differences are highly significant (Wilcoxon rank sum test, $p < 0.001$). Additionally, the final value of the objective function showing the comparability of the results is given.

We performed the same experiments with the SVM^{light} support vector machine implementation. The results are summarized in Table 3. We relaxed the stopping condition such that the SVM^{light} solutions are less accurate than the LIBSVM solutions. Nevertheless, the SVM^{light} algorithm is slower than the LIBSVM implementation using the SMO algorithm (see Table 2). Please note that according to the numerous implementation differences these experiments do not provide a fair comparison between SMO-like methods and decomposition algorithms using larger working sets.

data set (ℓ)	iterations	runtime	$f(\alpha^*)$
spam-database (4,601)	9,450	23.97 s	27,019,125
connect-4 (30,555)	17,315	5,589 s	13,557.520
MNIST (60,000)	42,347	282,262 s	143,175.447
face (220,000)	9,806	51,011 s	15,812.643

Table 3: Iterations, runtime and objective function value of the SVM^{light} experiments with working set size $q = 10$. Because of the enormous runtime, only one trial was conducted for the MNIST task.

4.3 Analysis of Different Parameter Regimes

The choice of the regularization parameter C and the parameter σ of the Gaussian kernel (6) influence the quadratic problem induced by the data. We analyzed this dependency using grid search on the connect-4 problem. The results are plotted in Figure 6.

All parameter configurations where LIBSVM-2.71 or LIBSVM-2.8 outperformed HMG have one important property in common, namely, that it is a bad idea to reselect an element of the previous working set. This is true when after most iterations both coordinates indexed by the working set are already optimal. This can happen for different reasons:

- For $\sigma \to 0$ the feature vectors corresponding to the training examples become more and more orthogonal and the quadratic problem $\mathcal{P}$ becomes (almost) separable.
- For increasing values of σ the example points become more and more similar in the feature space until they are hard to distinguish. This increases the quotient of the largest and smallest eigenvalue of Q. Thus, the solution of $\mathcal{P}$ is very likely to lie in a corner or on a very low dimensional edge of the box constraining the problem, that is, many of the α_i^* end up at the constraints 0 or C. This is even more likely for small values of C.

We argue that in practice parameter settings leading to those situations are not really relevant because they tend to produce degenerate solutions. Either almost all examples are selected as support vectors (and the SVM converges to nearest neighbor classification) or the information available are used inefficiently, setting most support vector coefficients to C. Both extremes are usually not intended in SVM learning. In our experiments, HMG performs best in the parameter regime giving well generalizing solutions.

4.4 Influence of the Shrinking Algorithm

A shrinking heuristics in a decomposition algorithm tries to predict whether a variable α_i will end up at the box constraint, that is, whether it will take one of the values 0 or C. In this case the variable is fixed at the boundary and the optimization problem is reduced accordingly. Of course, every heuristics can fail and thus when the stopping criterion is met these variables must be reconsidered. The temporary reduction of the problem restricts working set selection algorithms to a subset of possible choices. This may cause more iterations but has the potential to save a lot of runtime.

We repeated our experiments with the LIBSVM shrinking heuristics turned off to reveal the relation between the working set selection algorithms and shrinking, see Table 4. The experiments show that the influence of the shrinking algorithm on the different working set selection policies is highly task dependent. The time saved and even the algorithm for which more time was saved differs from problem to problem. For some problems the differences between the methods increase, for others they decrease. Compared to the experiments with shrinking turned on the results qualitatively remain the same.

4.5 Influence of the Cache Size

The speed (in terms of runtime, not iterations) of the SVM algorithm depends on the fraction of matrix rows fitting into the cache. We used the connect-4 data set to test the dependency of speed on the cache size. The full representation of the matrix Q requires nearly 3.5 GB of working memory for this problem. We trained SVMs with 20 MB (0.56% of the matrix), 40 MB (1.12%), 100 MB

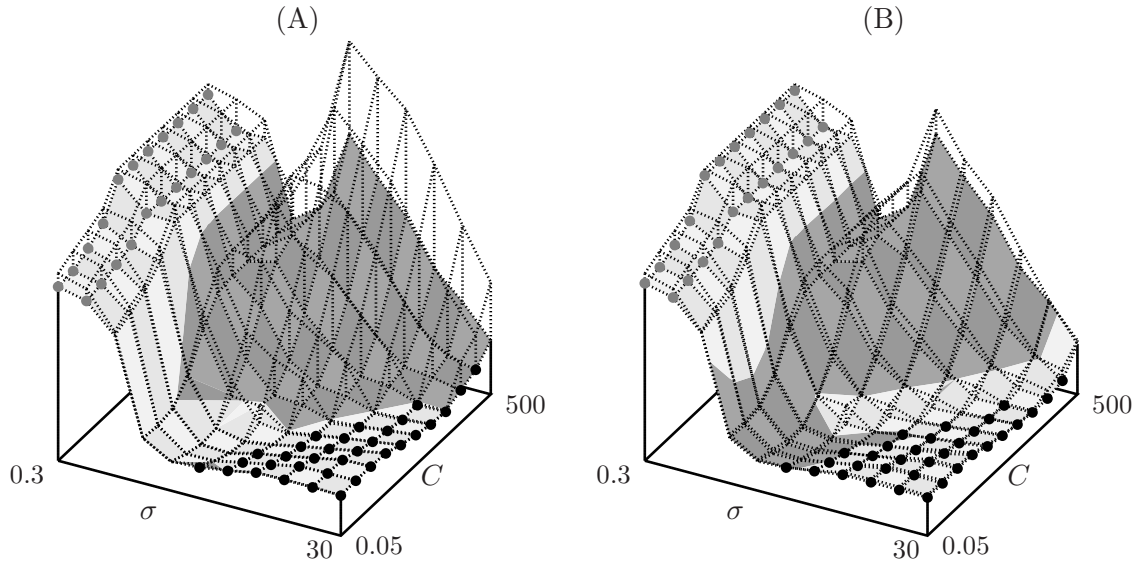


Figure 6: Influence of C and σ on the runtime for the connect-4 data set. The plots show on logarithmic scales the runtime of the SVM depending on C and σ. The comparison of HMG to LIBSVM-2.71 is plotted in (A), while plot (B) shows the comparison of HMG to LIBSVM-2.8. The colored shapes indicate the method needing less runtime, in light and dark gray for LIBSVM and HMG, respectively. Only the lower surface corresponding to the faster algorithm is drawn solid while the higher surface is indicated by the dotted grid lines. Both $\gamma = 1/(2\sigma^2)$ and C are considered in a range of factor $10,000$ containing the well generalizing parameter regime, see Table 2. The dots mark degenerate (and thus not desirable) solutions. The gray dots indicate that the solution uses at least 99 % of the training data as support vectors. If at least 99 % of the support vectors are at the upper bound C the solution is marked with a black dot.

(2.8%) and 200 MB (5.6%) cache. Because the shrinking heuristics reduces the amount of memory required for the storage of the relevant part of Q, the percentage values should be viewed with care. If all variables ending up at the box constraints are removed, the storage size of the matrix Q is about 134 MB. This matrix already fits into the 200 MB cache.

The results listed in Table 5 and plotted in Figure 7 clearly show that for small cache sizes the HMG algorithm is advantageous while for a large cache the LIBSVM-2.8 algorithm catches up.

These results can easily be explained. As long as there is a considerable chance to find a matrix row in the cache it is not necessary to use the cache friendly HMG strategy. In this case it is reasonable to minimize the number of iterations. This is best achieved by the LIBSVM-2.8 algorithm. If the cache is too small to store a relevant part of the kernel matrix it becomes advantageous to use HMG, because HMG produces at most one cache miss per iteration. We conclude that the HMG algorithm should be used for large scale problems.

data set	algorithm	iterations		runtime	
spam-database	LIBSVM-2.71	33,340	91.1 %	12.77 s	114 %
	LIBSVM-2.8	9,123	98.9 %	8.98 s	106 %
	HMG	9,342	88.4 %	11.41 s	124 %
connect-4	LIBSVM-2.71	65,735	100.9 %	2,223 s	243 %
	LIBSVM-2.8	45,466	99.9 %	1,567 s	213 %
	HMG	49,512	98.5 %	1,005 s	159 %
MNIST	LIBSVM-2.71	187,653	101.3 %	94,981 s	695 %
	LIBSVM-2.8	110,470	100.0 %	58,213 s	585 %
	HMG	155,182	101.5 %	41,097 s	549 %
face	LIBSVM-2.71	37,060	99.8 %	55,057 s	387 %
	LIBSVM-2.8	32,796	100.0 %	48,922 s	349 %
	HMG	43,066	101.8 %	33,001 s	293 %

Table 4: Iterations and time needed for solving the quadratic problem without shrinking. The percentage values refer to the corresponding results with shrinking turned on, that is, iterations and runtime of the experiments with shrinking turned on define the 100% mark. Due to the enormous runtime, for the data sets MNIST and face only one trial was conducted.

cache size	LIBSVM-2.71	LIBSVM-2.8	HMG
20 MB	958 s	766 s	656 s
40 MB	916 s	734 s	633 s
100 MB	758 s	649 s	583 s
200 MB	603 s	547 s	555 s

Table 5: The training time for the connect-4 task for the different working set selection algorithms depending on the cache size.

4.6 Number of Matrix Rows Considered

In the definition of the HMG algorithm we restrict ourselves to computing the gain using the two cached matrix rows corresponding to the previous working set. This seems to be an arbitrary restriction. To determine the influence of the number of rows considered we compared the HMG algorithm to two modified versions.

We computed the gain using only one matrix row corresponding to one element of the working set. The element chosen was alternated in every iteration such that a selected variable was used in exactly two successive iterations. This policy reduces the time required for the working set selection by about 50%. It can be considered as the minimal strategy avoiding asymmetries between the variables. The results comparing the usage of one and two rows are shown in Table 6.

Although the stopping criteria used are the same we get different final target function values. This happens due to the reduced number of pairs over which the maximum is taken in the one-row strategy. The values listed indicate that the experiments are roughly comparable, but the one-row strategy produces less accurate solutions in general.

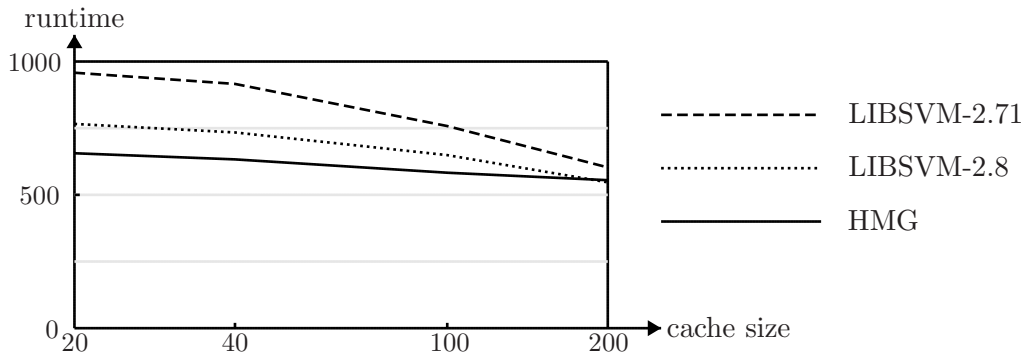


Figure 7: The training time in seconds for the connect-4 data set is plotted over the cache size in MB.

	two rows			one row		
data set	iterations	runtime	$f(\alpha^*)$	iterations	runtime	$f(\alpha^*)$
spam-database	**10,563**	**9.17 s**	27,019.140	14,023	10.99 s	27,019.134
connect-4	**50,281**	**633 s**	13,557.536	83,305	4,967 s	13,557.529

Table 6: Comparison of the one-row and the two-row strategy. The better values are printed in bold face. The differences are highly significant (Wilcoxon rank sum test, $p < 0.001$). The final target function values are lower using the one-row strategy.

The two-row strategy performed clearly better. The reasons for the poor performance of the one-row strategy are the higher number of iterations and, what is worse, the far higher number of unshrinking events. Because of the reduced amount of pairs considered this strategy is endangered to wrongly detect optimality on the shrunk problem causing a costly unshrinking process. Simple experiments indicated that the one-row strategy can compete if the problem is small. However, in this case both HMG strategies were outperformed by the LIBSVM algorithm.

The other strategy tested is to compute the gain for every cached matrix element available. Of course this algorithm is extremely time consuming and thus not practical for applications. This test gives us the minimum number of iterations the HMG working pair algorithm can achieve, as it is the strategy using all information available. It thus provides a bound on the performance of possible extensions of the algorithm using more than two cached matrix rows to determine the working pair. In the case where the whole matrix Q fits into the cache and all rows have already been computed, the strategy coincides with the exact greedy policy w.r.t. the gain. We compared the results to the the two-row strategy, see Table 7.

Again it is difficult to compare the experiments because the algorithm using all cached rows available generally stops later than the two-row strategy. We will nevertheless interpret the results, although this difficulty indicates that the bound on the possible performance of HMG algorithms using more than two rows may not be tight.

The behavior is clearly problem specific. On the connect-4 task both strategies nearly showed the same performance. This reveals that on some real world problems the two-row strategy cannot

	two rows		whole cache		iterations
data set	iterations	$f(\alpha^*)$	iterations	$f(\alpha^*)$	saved
spam-database	10,563	27,019.140	7,280	27,019.143	31 %
connect-4	50,281	13,557.536	51,285	13,557.538	0 %

Table 7: Comparison of the strategies using two matrix rows and the whole matrix cache available.

be outperformed by HMG strategies using more than two matrix rows. In contrast for the spam-database, the whole-cache strategy saved 31 percent of the iterations. This is a considerable amount which was bought dearly using all cached matrix rows for the working set selection. In practice one would like to use a faster method, which, for example, looks at a small fixed number of matrix rows. The reduction of the number of iterations will presumably be less for such strategies. Additionally, the non-trivial question for the row selection policy arises. Thus, for simplicity as well as performance we recommend to stick to the two-row HMG algorithm.

5. Conclusion

The time needed by a decomposition algorithm to solve the support vector machine (SVM) optimization problem up to a given accuracy depends highly on the working set selection. In our experiments with large data sets, that is, when training time really matters, our new hybrid maximum-gain working set selection (HMG) saved a lot of time compared to the latest second order selection algorithm. This speed-up is achieved by the avoidance of cache misses in the decomposition algorithm. In contrast, for small problems the LIBSVM-2.8 algorithm is faster. This result suggest a mixed strategy which switches between the algorithms depending on cache and problem size.

The main advantage of the HMG algorithm is its efficient usage of the matrix cache. It reselects almost always one element of the previous working set. Therefore, at most one matrix row needs to be computed in every iteration. The new algorithm obtains strong theoretical support as it is known to converge to an optimum under weak prerequisites, see Section 3.

The HMG algorithm is especially efficient for appropriate kernel and regularization parameter settings leading to well-generalizing solutions. Thus, it is the method of choice when parameters suiting the problem at hand are *roughly* known. It is for example a good idea to find out well working parameters using a small subset of the data and then train the SVM with the HMG algorithm using the whole data set.

Although LIBSVM-2.8 and HMG both select working sets using second order information, different target functions and variable sets are considered. It is an issue of future work to investigate the performance and the convergence properties of possible combinations of methods. In particular, an elaborate cooperation between the kernel cache strategy and the working set selection algorithm is promising to increase the efficiency of future algorithms.

Acknowledgment

We would like to thank Nikolas List for fruitful discussions and hints. We acknowledge support from Viisage Technology AG under contract “SecureFaceCheck”.

References

C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

A. Bordes, S. Ertekin, J. Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 5:1579–1619, 2005.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

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

D. Hush and C. Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51:51–71, 2003.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 169–184. MIT Press, 1999.

S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46:351–360, 2002.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, 2000.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:1288–1298, 2001.

N. List and H. U. Simon. A general convergence theorem for the decomposition method. In John Shawe-Taylor and Yoram Singer, editors, *Proceedings of the 17th Annual Conference on Learning Theory, COLT 2004*, volume 3120 of *LNCS*, pages 363–377. Springer-Verlag, 2004.

N. List and H. U. Simon. General polynomial time decomposition algorithms. In Peter Auer and Ron Meir, editors, *Proceedings of the 18th Annual Conference on Learning Theory, COLT 2005*, volume 3559 of *LNCS*, pages 308–322. Springer-Verlag, 2005.

E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In J. Principe, L. Giles, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII*, pages 276–285. IEEE Press, 1997.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12, pages 185–208. MIT Press, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

N. Takahashi and T. Nishi. Rigorous proof of termination of SMO algorithm for support vector machines. *IEEE Transaction on Neural Networks*, 16(3):774–776, 2005.

I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New-York, 1998.

S. V. N. Vishwanathan, A. J. Smola, and M. N. Murty. SimpleSVM. In T. Fawcett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pages 760–767. AAAI Press, 2003.

Journal of Machine Learning Research 7 (2006) 1467–1492 Submitted 10/05; Revised 2/06; Published 7/06

Parallel Software for Training Large Scale Support Vector Machines on Multiprocessor Systems*

Luca Zanni ZANNI.LUCA@UNIMO.IT
Thomas Serafini SERAFINI.THOMAS@UNIMO.IT
Department of Mathematics
University of Modena and Reggio Emilia
via Campi 213/B, Modena, 41100, Italy

Gaetano Zanghirati G.ZANGHIRATI@UNIFE.IT
Department of Mathematics
University of Ferrara
Scientific-Technological Campus, Block B
via Saragat 1, Ferrara, 44100, Italy

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

Parallel software for solving the quadratic program arising in training *support vector machines* for classification problems is introduced. The software implements an iterative decomposition technique and exploits both the storage and the computing resources available on multiprocessor systems, by distributing the heaviest computational tasks of each decomposition iteration. Based on a wide range of recent theoretical advances, relevant decomposition issues, such as the quadratic subproblem solution, the gradient updating, the working set selection, are systematically described and their careful combination to get an effective parallel tool is discussed. A comparison with state-of-the-art packages on benchmark problems demonstrates the good accuracy and the remarkable time saving achieved by the proposed software. Furthermore, challenging experiments on real-world data sets with millions training samples highlight how the software makes large scale standard nonlinear support vector machines effectively tractable on common multiprocessor systems. This feature is not shown by any of the available codes.

Keywords: support vector machines, large scale quadratic programs, decomposition techniques, gradient projection methods, parallel computation

1. Introduction

Training support vector machines (SVM) for binary classification requires to solve the following convex quadratic programming (QP) problem (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000)

$$\begin{array}{rl} \min & \mathcal{F}(\alpha) = \frac{1}{2}\alpha^T G\alpha - \sum_{i=1}^{n}\alpha_i \\ \text{sub. to} & \sum_{i=1}^{n} y_i\alpha_i = 0, \\ & 0 \le \alpha_i \le C, \qquad i = 1,\ldots,n, \end{array} \tag{1}$$

*. This work was supported by the Italian Education, University and Research Ministry via the FIRB Projects *"Statistical Learning: Theory, Algorithms and Applications"* (grant RBAU01877P) and *"Parallel Algorithms and Numerical Nonlinear Optimization"* (grant RBAU01JYPN).

whose size n is equal to the number of examples in the given training set

$$D = \left\{(x_i, y_i),\ i = 1,\ldots,n, \quad x_i \in \mathbb{R}^M,\ y_i \in \{-1,1\}\right\},$$

and the entries of G are defined by

$$G_{ij} = y_i y_j K(x_i, x_j), \quad i,j = 1,2,\ldots,n,$$

where $K : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ denotes the kernel function. The main features of this problem are the density of the quadratic form and the special feasible region defined by box constraints and a single linear equality constraint. In many practical SVM applications, standard QP solvers based on the explicit storage of the Hessian matrix G may be very inefficient or, in the case of large data sets, even not applicable due to excessive memory requirements. For these reasons in recent years a lot of attention has been dedicated to this problem and several *ad hoc* strategies have been developed, which are able to solve the problem with G out of memory. Among these strategies, the decomposition techniques have been the most investigated approaches and have given rise to the state-of-the-art software for the SVM QP problem. The idea behind the decomposition techniques consists in splitting the problem into a sequence of smaller QP subproblems, sized $n_{\rm sp}$ say, that can be stored in the available memory and efficiently solved (Boser et al., 1992; Chang and Lin, 2001; Collobert and Bengio, 2001; Joachims, 1998; Osuna et al., 1997; Platt, 1998). At each decomposition step, a subset of the variables, usually called *working set*, is optimized through the solution of the subproblem in order to obtain a progress towards the minimum of the objective function $\mathcal{F}(\alpha)$. Effective implementations of this simple idea involve important theoretical and practical issues. From the theoretical point of view, the policy for updating the working set plays a crucial role since it can guarantee the strict decrease of the objective function at each step (Hush and Scovel, 2003). The most used working set selections rely on the violations of the Karush-Kuhn-Tucker (KKT) first order optimality conditions. In case of working sets of minimal size, that is sized 2, a proper selection via the *maximal-violating pair* principle (or related criteria) is sufficient to ensure asymptotic convergence of the decomposition scheme (Lin, 2002; Chen et al., 2005). For larger working sets, convergence proofs are available under a further condition which ensures that the distance between two successive approximations tends to zero (Lin, 2001a; Palagi and Sciandrone, 2005). Furthermore, based on these working set selections and further assumptions, the linear convergence rate can be also proved (Lin, 2001b). For the practical efficiency of a decomposition technique, the fast convergence and the low computational cost per iteration seem the most important features. Unfortunately, these goals are conflicting since the strategies to improve the convergence rate (as the use of large working sets or the selections based on second order information) usually increase the cost per iteration. Examples of good trade-offs between the two goals are given by the most widely used decomposition packages: LIBSVM (Chang and Lin, 2001) and SVM^{light} (Joachims, 1998).

The LIBSVM software is developed for working sets sized 2, hence it tends to minimize the computational cost per iteration. In fact, in this case the inner QP subproblem can be analytically solved without requiring a numerical QP solver and the updating of the objective gradient only involves the two Hessian columns corresponding to the updated variables. On the other hand, if only few components are updated per iteration, slow convergence is generally implied. In the last LIBSVM release (ver. 2.8) this drawback is attenuated by a new working set selection that partially exploits the second order information, thus getting only a moderate increase of the computational cost with respect to the standard selections (Fan et al., 2005).

The SVM^{light} algorithm uses a more general decomposition strategy, in the sense that it can also exploit working sets of size larger than 2. By updating more variables per iteration, such an approach is well suited for a faster convergence, but it introduces additional difficulties and costs. A generalized maximal-violating pair policy for the working set selection and a numerical solver for the inner QP subproblems are needed; furthermore, we must recall that the more variables are changed per iteration, the more expensive is the objective gradient updating. Even if SVM^{light} can run with any working set size, numerical experiences show that it effectively faces the above difficulties only in case of small sized working sets ($n_{\text{sp}} = O(10)$), where it often exhibits comparable performance with LIBSVM.

Following the SVM^{light} decomposition framework, another attempt to reach a good trade-off between convergence rate and cost per iteration was introduced by Zanghirati and Zanni (2003). This was the first approach suited for an effective implementation on multiprocessors systems. Unlike SVM^{light}, it is designed to manage medium-to-large sized working sets ($n_{\text{sp}} = O(10^2)$ or $n_{\text{sp}} = O(10^3)$), that allow the scheme to converge in very few iterations, whose most expensive tasks (subproblem solving and gradient updating) can be easily and fruitfully distributed among the available processors. Of course, several issues must be addressed to achieve good performance, such as limiting the overhead for kernel evaluations and, also important, choosing a suitable inner QP solver. Zanghirati and Zanni (2003) obtained an efficient subproblem solution by a gradient projection-type method: it exploits the simple structure of the constraints, exhibits good convergence rate and is well suited for a parallel implementation. The promising results given by this parallel scheme can be now further improved thanks to some recent studies on both the gradient projection QP solvers (Serafini et al., 2005; Dai and Fletcher, 2006) and the selection rules for large sized working sets (Serafini and Zanni, 2005). On the basis of these studies a new *parallel gradient projection-based decomposition technique* (PGPDT) is developed and implemented in software available at `http://www.dm.unife/gpdt`.

Other parallel approaches to SVMs have been recently proposed, by splitting the training data into subsets and distributing them among the processors. Some of these approaches, such as those by Collobert et al. (2002) and by Dong et al. (2003), do not aim to solve the problem (1) and then perform non-standard SVM training. Collobert et al. (2002) presented a mixture of multiple SVMs where, cyclically, single SVMs are trained on subsets of the training set and a neural network is used to assign samples to different subsets. Dong et al. (2003) used a block-diagonal approximation of the kernel matrix to derive independent SVMs and filter out the examples which are estimated to be non-support vectors; then a new serial SVM is trained on the collected support vectors. The idea to combine asynchronously-trained SVMs is revisited also by the *cascade algorithm* introduced by Graf et al. (2005). The support vectors given by the SVMs of a cascade layer are combined to form the training sets of the next layer. At the end, the global KKT conditions are checked and the process is eventually restarted from the beginning, re-inserting the computed support vectors in the training subsets of the first layer. The authors prove that this feedback loop allows the algorithm to converge to a solution of (1) and consequently to perform a standard training. Unfortunately, for all these approaches no parallel code is available yet.

This work deals with the PGPDT software and its practical behaviour. First, we give the reader an exhaustive self-contained description of the PGPDT algorithm by showing how the crucial subtasks, singly developed in very recent works, are combined with appropriate load balancing strategies newly designed to get an effective parallel tool. Second, we show how, by exploiting the resources of common multiprocessor systems, PGPDT achieves good time speedup in comparison

with state-of-the-art serial packages and makes nonlinear SVMs tractable even on millions training samples.

The paper is organized as follows: Section 2 states the decomposition framework and describes its parallelization, Section 3 compares the PGPDT with SVM^{light} and LIBSVM on medium-to-large benchmark data sets and also faces some $O(10^6)$ real-world problems, Section 4 draws the main conclusions and future developments.

2. The Decomposition Framework and its Parallelization

To describe in detail the decomposition technique implemented by PGPDT we need some basic notations. At each decomposition iteration, the indices of the variables α_i, $i = 1,\ldots,n$, are split into the set $\mathcal{B}$ of *basic* variables, usually called the *working set*, and the set $\mathcal{N} = \{1,2,\ldots,n\} \setminus \mathcal{B}$ of *nonbasic* variables. As a consequence, the kernel matrix G and the vectors $\alpha = (\alpha_1,\ldots,\alpha_n)^T$ and $y = (y_1,\ldots,y_n)^T$ can be arranged with respect to $\mathcal{B}$ and $\mathcal{N}$ as follows:

$$G = \begin{bmatrix} G_{\mathcal{B}\mathcal{B}} & G_{\mathcal{B}\mathcal{N}} \\ G_{\mathcal{N}\mathcal{B}} & G_{\mathcal{N}\mathcal{N}} \end{bmatrix}, \qquad \alpha = \begin{bmatrix} \alpha_{\mathcal{B}} \\ \alpha_{\mathcal{N}} \end{bmatrix}, \qquad y = \begin{bmatrix} y_{\mathcal{B}} \\ y_{\mathcal{N}} \end{bmatrix}.$$

Furthermore, we denote by n_{sp} the size of the working set ($n_{\text{sp}} = \#\mathcal{B}$) and by α^* a solution of (1). Finally, suppose a distributed-memory multiprocessor system equipped with N_{P} processors is available for solving the problem (1) and that each processor has a local copy of the training set.

The decomposition strategy used by the PGPDT falls within the general scheme stated in Algorithm PDT. Here, we denote by the label "Distributed task" the steps where the N_{P} processors cooperate to perform the required computation; in these steps communications and synchronization are needed. In the other steps, the processors asynchronously perform the same computations on the same input data to obtain a local copy of the expected output data.

It must be observed that algorithm PDT essentially follows the SVM^{light} decomposition scheme proposed by Joachims (1998), but it allows to distribute among the available processors the subproblem solution in step A2 and the gradient updating in step A3. Thus, two important implications can be remarked: from the theoretical viewpoint, the PDT algorithm satisfies the same convergence properties of the SVM^{light} algorithm, but, in practice, it requires new implementation strategies in order to effectively exploit the resources of a multiprocessor system. Here, we state the main convergence results of the PDT algorithm and we will describe in the next subsections how its steps have been implemented in the PGPDT software.

The convergence properties of the sequence $\{\alpha^{(k)}\}$ generated by the PDT algorithm are mainly based on the special rule (4) for the working set selection. The rule was originally introduced by Joachims (1998) following an idea similar to the Zoutendijk's feasible direction approach, to define basic variables that make possible a rapid decrease of the objective function. The asymptotic convergence of the decomposition schemes based on this working set selection was first proved by Lin (2001a) by relating the selection rule with the violation of the KKT conditions and by assuming the following strict block-wise convexity assumption on $\mathcal{F}(\alpha)$:

$$\min_{\mathcal{J}}(\lambda_{\min}(G_{\mathcal{J}\mathcal{J}})) > 0\,, \tag{5}$$

ALGORITHM PDT Parallel decomposition technique

A1. *Initialization*. Set $\alpha^{(1)} = 0$ and let $n_{\rm sp}$ and n_c be two integer values such that $n \geq n_{\rm sp} \geq n_c > 0$, n_c even. Choose $n_{\rm sp}$ indices for the working set $\mathcal{B}$ and set $k = 1$.

A2. *QP subproblem solution*. **[Distributed task]** Compute the solution $\alpha_{\mathcal{B}}^{(k+1)}$ of

$$\begin{array}{rl} \min & \frac{1}{2}\alpha_{\mathcal{B}}^T G_{\mathcal{B}\mathcal{B}}\alpha_{\mathcal{B}} + \left(G_{\mathcal{B}\mathcal{N}}\alpha_{\mathcal{N}}^{(k)} - 1_{\mathcal{B}}\right)^T \alpha_{\mathcal{B}} \\ \text{sub. to} & \sum_{i\in\mathcal{B}} y_i\alpha_i = -\sum_{i\in\mathcal{N}} y_i\alpha_i^{(k)} \ , \\ & 0 \leq \alpha_i \leq C, \quad \forall\, i \in \mathcal{B} \ , \end{array} \tag{2}$$

where $1_{\mathcal{B}}$ is the $n_{\rm sp}$-vector of all one; set $\alpha^{(k+1)} = \left(\begin{array}{cc} \alpha_{\mathcal{B}}^{(k+1)^T}, & \alpha_{\mathcal{N}}^{(k)^T} \end{array} \right)^T$.

A3. *Gradient updating*. **[Distributed task]** Update the gradient

$$\nabla\mathcal{F}\left(\alpha^{(k+1)}\right) = \nabla\mathcal{F}\left(\alpha^{(k)}\right) + \left[\begin{array}{c} G_{\mathcal{B}\mathcal{B}} \\ G_{\mathcal{N}\mathcal{B}} \end{array} \right] \left(\alpha_{\mathcal{B}}^{(k+1)} - \alpha_{\mathcal{B}}^{(k)}\right) \tag{3}$$

and terminate if $\alpha^{(k+1)}$ satisfies the KKT conditions.

A4. *Working set updating*. Update $\mathcal{B}$ by the following selection rule:

A4.1. Find the indices corresponding to the nonzero components of the solution of

$$\begin{array}{rl} \min & \nabla\mathcal{F}\left(\alpha^{(k+1)}\right)^T d \\ \text{sub. to} & y^T d = 0 \ , \\ & d_i \geq 0 \quad \text{for } i \text{ such that } \alpha_i^{(k+1)} = 0 \ , \\ & d_i \leq 0 \quad \text{for } i \text{ such that } \alpha_i^{(k+1)} = C \ , \\ & -1 \leq d_i \leq 1 \ , \\ & \#\{d_i \mid d_i \neq 0\} \leq n_c \ . \end{array} \tag{4}$$

Let $\bar{\mathcal{B}}$ be the set of these indices.

A4.2. Fill $\bar{\mathcal{B}}$ up to $n_{\rm sp}$ entries with indices $j \in \mathcal{B}$. Set $\mathcal{B} = \bar{\mathcal{B}}$, $k \leftarrow k+1$ and go to A2.

where $\mathcal{J}$ is any subset of $\{1,\ldots,n\}$ with $\#\mathcal{J} \leq n_{sp}$ and $\lambda_{\min}(G_{\mathcal{J}\mathcal{J}})$ denotes the smallest eigenvalue of $G_{\mathcal{J}\mathcal{J}}$. This condition is used to prove that there exists $\tau > 0$ such that

$$\mathcal{F}(\alpha^{(k+1)}) \leq \mathcal{F}(\alpha^{(k)}) - \frac{\tau}{2}\|\alpha^{(k+1)} - \alpha^{(k)}\|^2 \qquad \forall k\,, \tag{6}$$

from which the important property $\lim_{k\to\infty}\|\alpha^{(k+1)} - \alpha^{(k)}\| = 0$ can be derived. Of course, the assumption (5) is satisfied when G is positive definite (for example, when the Gaussian kernel is used and all the training examples are distinct), but it may not hold in other instances of the problem (1). Convergence results that do not require the condition (5) are given by Lin (2002) and Palagi and Sciandrone (2005).

For the special case $n_{sp} = 2$, where the selection rule (4) gives only the two indices corresponding to the maximal violation of the KKT conditions (the *maximal-violating pair*), Lin (2002) has shown that the assumption (5) is not necessary to ensure the convergence.

For any working set size, Palagi and Sciandrone (2005) have shown that the condition (6) is ensured by solving at each iteration the following proximal point modification of the subproblem (2):

$$\begin{array}{rl} \min & \frac{1}{2}\alpha_{\mathcal{B}}^T G_{\mathcal{B}\mathcal{B}}\alpha_{\mathcal{B}} + \left(G_{\mathcal{B}\mathcal{N}}\alpha_{\mathcal{N}}^{(k)} - 1_{\mathcal{B}}\right)^T \alpha_{\mathcal{B}} + \frac{\tau}{2}\|\alpha_{\mathcal{B}} - \alpha_{\mathcal{B}}^{(k)}\|^2 \\ \text{sub. to} & \sum_{i\in\mathcal{B}} y_i\alpha_i = -\sum_{i\in\mathcal{N}} y_i\alpha_i^{(k)}\,, \\ & 0 \leq \alpha_i \leq C, \quad \forall\, i \in \mathcal{B}\,. \end{array} \tag{7}$$

Unfortunately, this modification affects the behaviour of standard decomposition schemes in a way which is not completely understood yet. Our preliminary experiences suggest that sufficiently large values of τ can easily allow a better performance of the inner QP solvers, but those values often imply a dangerous decrease in the convergence rate of the decomposition technique. On the other hand, too small values for τ do not produce essential differences with respect to the schemes where the subproblem (2) is solved.

In the PGPDT software, besides the default setting which implements the standard PDT algorithm, two different ways to generate a sequence satisfying the condition (6) are available by user selection: (i) solving the subproblem (7) in place of (2) at each iteration or (ii) solving (7) only as emergency step, when $\alpha_{\mathcal{B}}^{(k+1)}$ obtained via (2) fails to satisfy (6). All the computational experiments of Section 3 are carried out with the default setting that generally yields the best performance. For what concerns the practical rule used in the PGPDT to stop the iterative procedure, the fulfilment of the KKT conditions within a prefixed tolerance is checked (with the equality constraint multiplier computed as suggested by Joachims, 1998). The default tolerance is 10^{-3}, as it is usual in SVM packages, but different values can be selected by the user.

Before to describe the PGPDT implementation in detail, it must be recalled that this software is designed to be effective in case of sufficiently large n_{sp}, i.e., when iterations with well parallelizable tasks are generated. For this reason, in the sequel the reader may assume n_{sp} to be of medium-to-large size.

2.1 Parallel Gradient Projection Methods for the Subproblems

The inner QP subproblems (2) and (7) can fit into the following general form:

$$\min_{w\in\Omega} \quad f(w) = \frac{1}{2}w^T A w + b^T w \tag{8}$$

where $A \in \mathbb{R}^{n_{sp} \times n_{sp}}$ is dense, symmetric and positive semidefinite, $w, b \in \mathbb{R}^{n_{sp}}$ and the feasible region Ω is defined by

$$\Omega = \{w \in \mathbb{R}^{n_{sp}}, \quad \ell \leq w \leq u, \quad c^T w = \gamma\}, \qquad \ell,\, u,\, c \in \mathbb{R}^{n_{sp}},\ \ell < u. \tag{9}$$

We recall that the size n_{sp} is such that A can fit into the available memory.

Since subproblem (8) appears at each decomposition iteration, an effective inner solver becomes a crucial tool for the performance of a decomposition technique. The standard library QP solvers can be successfully applied only in the small size case ($n_{sp} = O(10)$), since their computational cost easily degrades the performance of a decomposition technique based on medium-to-large n_{sp}. For such kind of decomposition schemes, it is essential to design more efficient inner solvers able to exploit the special features of (8) and, for the PGPDT purposes, with the additional property to be easily parallelizable. To this end, the gradient projection methods are very appealing approaches (Bertsekas, 1999). They consist in a sequence of projections onto the feasible region, that are nonexpensive operations in the case of the special constraints (9). In fact, the projection onto Ω (denoted by $P_\Omega(\cdot)$) can be performed in $O(n_{sp})$ operations by efficient algorithms, like those by Pardalos and Kovoor (1990) and by Dai and Fletcher (2006). Furthermore, the single iteration core consists essentially in an n_{sp}-dimensional matrix-vector product that is suited to be optimized (by exploiting the vector sparsity) and also to be parallelized. Thus, the simple and nonexpensive iteration motivates the interest for these approaches as possible alternative to standard solvers based on expensive factorizations (usually requiring $O(n_{sp}^3)$ operations). A general parallel gradient projection scheme for (8) is depicted in Algorithm PGPM. As in the classical gradient projection methods, at each iteration a feasible descent direction $d^{(k)}$ is obtained by projecting onto Ω a point derived by taking a steepest descent step of length ρ_k from the current $w^{(k)}$. A linesearch procedure is then applied along the direction $d^{(k)}$ to decide the step size λ_k able to ensure the global convergence. The parallelization of this iterative scheme is simply obtained by a block row-wise distribution of A and by a parallel computation of the heaviest task of each iteration: the matrix-vector product $Ad^{(k)}$.

Concerning the convergence rate, that is the key element for the PGPM performance, the choices of both the steplength ρ_k and the linesearch parameter λ_k play a crucial role. Recent works have shown that appropriate selection rules for these parameters can significantly improve the typical slow convergence rate of the traditional gradient projection approaches (refer to Ruggiero and Zanni, 2000b, for the R-linear convergence of PGPM-like schemes). From the steplength viewpoint, very promising results are actually obtained with selection strategies based on the Barzilai-Borwein (BB) rules (Barzilai and Borwein, 1988):

$$\rho_{k+1}^{\mathrm{BB1}} = \frac{{d^{(k)}}^T d^{(k)}}{{d^{(k)}}^T A d^{(k)}}, \qquad \rho_{k+1}^{\mathrm{BB2}} = \frac{{d^{(k)}}^T A d^{(k)}}{{d^{(k)}}^T A^2 d^{(k)}}.$$

The importance of these rules has been observed in combination with both monotone and nonmonotone linesearch strategies (Birgin et al., 2000; Dai and Fletcher, 2005; Ruggiero and Zanni, 2000a). In particular, for the SVM applications, the special BB steplength selections proposed by Serafini et al. (2005), for the monotone scheme, and by Dai and Fletcher (2006), for the nonmonotone method, seem very efficient.

The *generalized variable projection method* (GVPM) by Serafini et al. (2005) uses a standard limited minimization rule as linesearch technique and an adaptive alternation of the two BB formulae. It outperforms the monotone gradient projection scheme used by Zanghirati and Zanni (2003),

ALGORITHM PGPM Parallel gradient projection method for step A2 of Algorithm PDT.

B1. *Initialization.*

[Data distribution] $\forall p = 1,\ldots,N_{\mathrm{P}}$: allocate a row-wise slice $A_p = (a_{ij})_{i\in I_p,\ j=1,\ldots,n_{\mathrm{sp}}}$ of A, where I_p is the subset of row indices belonging to processor p:

$$I_p \subset \{1,\ldots,n\}, \quad \bigcup_{p=1}^{N_{\mathrm{P}}} I_p = \{1,\ldots,n\}\,, \quad I_i \cap I_j = \emptyset \ \text{ for } i \neq j.$$

Furthermore, allocate local copies of all the other input data.

Initialize the parameters for the steplength selection rule and for the linesearch strategy.

Set $w^{(0)} \in \Omega\,, \quad 0 < \rho_{\min} < \rho_{\max}\,, \quad \rho_0 \in [\rho_{\min}, \rho_{\max}]\,, \quad k = 0.$

[Distributed task] $\forall p = 1,\ldots,N_{\mathrm{P}}$: compute the local slice $t_p^{(0)} = A_p w^{(0)}$ and send it to all the other processors; assemble a local copy of the full $t^{(0)} = Aw^{(0)}$ vector.

Set $\quad g^{(0)} = \nabla f(w^{(0)}) = Aw^{(0)} + b = t^{(0)} + b\,.$

B2. *Projection.*

Terminate if $w^{(k)}$ satisfies a stopping criterion; otherwise compute the descent direction

$$d^{(k)} = P_\Omega\left(w^{(k)} - \rho_k g^{(k)}\right) - w^{(k)}\,.$$

B3. *Matrix-vector product.*

[Distributed task] $\forall p = 1,\ldots,N_{\mathrm{P}}$: compute the local slice $z_p^{(k)} = A_p d^{(k)}$ and send it to all the other processors; assemble a local copy of the full $z^{(k)} = Ad^{(k)}$ vector.

B4. *Linesearch.*

Compute the linesearch step λ_k and $w^{(k+1)} = w^{(k)} + \lambda_k d^{(k)}$.

B5. *Update.*

Compute

$$t^{(k+1)} = Aw^{(k+1)} = t^{(k)} + \lambda_k A d^{(k)} = t^{(k)} + \lambda_k z^{(k)},$$
$$g^{(k+1)} = \nabla f(w^{(k+1)}) = Aw^{(k+1)} + b = t^{(k+1)} + b,$$

and a new steplength ρ_{k+1}.

Update the parameters for the linesearch strategy, set $k \leftarrow k+1$ and go to step B2.

Initialization (step B1). Set $i_\rho = 2,\ n_{\min} = 3,\ n_{\max} = 10,\ \lambda_\ell = 0.1,\ \lambda_u = 5,\ n_\rho = 1$.

Linesearch (step B4). Compute $\lambda_k = \arg\min_{\lambda\in[0,1]} f(w^{(k)} + \lambda d^{(k)})$.

Update (step B5).

If $d^{(k)^T} A d^{(k)} = 0$ then

set $\rho_{k+1} = \rho_{\max}$

else

compute $\rho_{k+1}^{\mathrm{BB1}}$, $\rho_{k+1}^{\mathrm{BB2}}$, $\lambda_{\mathrm{opt}} = \arg\min_{\lambda} f(w^{(k)} + \lambda d^{(k)})$.

If $(n_\rho \geq n_{\min})$ and $\Big[(n_\rho \geq n_{\max})$ or $(\rho_{k+1}^{\mathrm{BB2}} \leq \rho_k \leq \rho_{k+1}^{\mathrm{BB1}})$

or $\Big((\lambda_{\mathrm{opt}} < \lambda_\ell$ and $\rho_k = \rho_k^{\mathrm{BB1}})$ or $(\lambda_{\mathrm{opt}} > \lambda_u$ and $\rho_k = \rho_k^{\mathrm{BB2}})\Big)\Big]$ then

set $i_\rho \leftarrow \mathrm{mod}(i_\rho, 2) + 1,\quad n_\rho = 0$;

end.

end.

Compute $\rho_{k+1} = \min\Big\{\rho_{\max}, \max\Big\{\rho_{\min}, \rho_{k+1}^{\mathrm{BB}i_\rho}\Big\}\Big\}$ and set $n_\rho \leftarrow n_\rho + 1$.

Figure 1: linesearch and steplength rule for the GVPM method.

that was simply based on an alternation of the BB rules every three iterations. Furthermore, the numerical experiments reported by (Serafini et al., 2005) show that the GVPM is much more efficient than the pr_LOQO (Smola, 1997) and MINOS (Murtagh and Saunders, 1998) solvers, two softwares widely used within the machine learning community. GVPM steplength selection and linesearch are described in Figure 1.

The Dai-Fletcher scheme is based on the following steplength selection:

$$\rho_{k+1}^{\mathrm{DF}} = \frac{\sum_{i=0}^{m-1} s^{(k-i)^T} s^{(k-i)}}{\sum_{i=0}^{m-1} s^{(k-i)^T} v^{(k-i)}}\ , \qquad m \geq 1\ , \tag{10}$$

where $s^{(j)} = w^{(j+1)} - w^{(j)}$ and $v^{(j)} = g^{(j+1)} - g^{(j)}$, $(g^{(j)} = \nabla f(w^{(j)}))$, $j = 0, 1, \ldots$. Observe that the case $m = 1$ reduces to the standard BB rule $\rho_{k+1}^{\mathrm{BB1}}$. In order to frequently accept the full step $w^{(k+1)} = w^{(k)} + d^{(k)}$ generated with the above steplength, a special nonmonotone linesearch is used. Figure 2 describes the version of the Dai-Fletcher method corresponding to the parameters setting suggested by Zanni (2006) for the SVM applications. It may be observed that the linesearch parameter $\lambda_k = \arg\min_{\lambda\in[0,1]} f(w^{(k)} + \lambda d^{(k)})$ is used only if $f(w^{(k)} + d^{(k)}) \geq f_{\mathrm{ref}}$ and not at each iteration, as in the GVPM. The steplength selection corresponds to the rule (10) with $m = 2$ and, for what concerns the iteration cost, no significant additional tasks are required in comparison to the GVPM ($g^{(k+1)}$ is already available in step B5).

The PGPDT software can run the PGPM with either the GVPM or the Dai-Fletcher scheme, the latter being the default due to better experimental convergence rate (Zanni, 2006).

We end this subsection with some further details about the PGPM implementation used within the PGPDT software. The starting point $w^{(0)}$ is $P_\Omega(\alpha_{\mathcal{B}}^{(k)})$ if the stopping rule of the decomposition procedure is nearly satisfied, otherwise $w^{(0)} = P_\Omega(0)$ is used. This aims to start the PGPM with

Initialization (step B1). Set $L = 2$, $f_{\text{ref}} = \infty$, $f_{\text{best}} = f_c = f(w^{(0)})$, $h = 0$, $k = 0$, $s^{(k-1)} = v^{(k-1)} = 0$.

Linesearch (step B4).

If $\left(k = 0 \text{ and } f(w^{(k)} + d^{(k)}) \geq f(w^{(k)})\right)$ or $\left(k > 0 \text{ and } f(w^{(k)} + d^{(k)}) \geq f_{\text{ref}}\right)$ then

$$w^{(k+1)} = w^{(k)} + \lambda_k d^{(k)} \qquad \text{with} \qquad \lambda_k = \underset{\lambda \in [0,1]}{\arg\min} f(w^{(k)} + \lambda d^{(k)})$$

else

$$w^{(k+1)} = w^{(k)} + d^{(k)}$$

end.

Update (step B5). Compute $s^{(k)} = w^{(k+1)} - w^{(k)}$; $v^{(k)} = g^{(k+1)} - g^{(k)}$.

If $s^{(k)^T} v^{(k)} = 0$ then

set $\rho_{k+1} = \rho_{\max}$

else

If $s^{(k-1)^T} v^{(k-1)} = 0$ then

set $\rho_{k+1} = \min\left\{\rho_{\max}, \max\left\{\rho_{\min}, \dfrac{s^{(k)^T} s^{(k)}}{s^{(k)^T} v^{(k)}}\right\}\right\}$

else

set $\rho_{k+1} = \min\left\{\rho_{\max}, \max\left\{\rho_{\min}, \dfrac{s^{(k)^T} s^{(k)} + s^{(k-1)^T} s^{(k-1)}}{s^{(k)^T} v^{(k)} + s^{(k-1)^T} v^{(k-1)}}\right\}\right\}$

end.

end.

If $f(w^{(k+1)}) < f_{\text{best}}$ then

set $f_{\text{best}} = f(w^{(k+1)})$, $f_c = f(w^{(k+1)})$, $h = 0$;

else

set $f_c = \max\left\{f_c, f(w^{(k+1)})\right\}$, $h = h + 1$;

If $h = L$ then

set $f_{\text{ref}} = f_c$, $f_c = f(w^{(k+1)})$, $h = 0$;

end.

end.

Figure 2: linesearch and steplength rule for the Dai-Fletcher method.

sparse vectors in the first decomposition steps, and to save inner solver iterations at the end of the decomposition, where slight changes in $\alpha_{\mathcal{B}}^{(k)}$ are expected. At the beginning we also set $\rho_{\min} = 10^{-10}$, $\rho_{\max} = 10^{10}$ and $\rho_0 = \min\{\rho_{\max}, \max\{\rho_{\min}, \bar{\rho}_0\}\}$, where $\bar{\rho}_0 = \|P_\Omega(w^{(0)} - (Aw^{(0)} + b)) - w^{(0)}\|_\infty^{-1}$. For the computation of $P_\Omega(\cdot)$ in step B2, the default is the following: if $n_{\text{sp}} \leq 20$ the bisection-like method described by Pardalos and Kovoor (1990) is used, else the secant-based algorithm proposed by Dai and Fletcher (2006) is chosen, that usually is faster for large size. However, the user can select one of the two projectors. Finally, we remark that the PGPM stopping rule is the same used

for the decomposition technique: the fulfilment of the KKT conditions within a prefixed tolerance. In the PGPDT, the tolerance required to the inner solver depends on the quality of the outer iterate $\alpha^{(k)}$: in the first iterations the same tolerance as the decomposition scheme is used, while a progressively lower tolerance is imposed when $\alpha^{(k)}$ nearly satisfies the outer stopping criterion. In our experience, a more accurate inner solution just from the beginning doesn't imply remarkable increase of the overall performance.

2.2 Parallel Gradient Updating

The gradient updating in step A3 is usually the most expensive task of a decomposition iteration. Since the matrix G is assumed to be out of memory, in order to obtain $\nabla \mathcal{F}(\alpha^{(k+1)})$ some entries of G need to be computed and, consequently, some kernel evaluations are involved that can be very expensive in case of large sized input space and not much sparse training examples. Thus, any strategy able to save kernel evaluations or to optimize their computation is crucial for minimizing the time consumption for updating the gradient. The updating formula (3) allows to get $\nabla \mathcal{F}(\alpha^{(k+1)})$ by involving only the columns of G corresponding to the indices for which $(\alpha_i^{(k+1)} - \alpha_i^{(k)}) \neq 0,\ i \in \mathcal{B}$. Further improvements in the number of kernel evaluations can be obtained by introducing a caching strategy, consisting in using an area of the available memory to store some elements of G to avoid their recomputation in subsequent iterations. PGPDT fills the caching area with the columns of G involved in (3); when the cache is full, the current columns substitute those that have not been used for the largest number of iterations. This simple trick seems to well combine with the working set selection used in step A4, which forces some indices of the current $\mathcal{B}$ to remain in the new working set (see the next section for more details), and remarkable reduction of the kernel evaluations are often observed. Nevertheless, the improvements implied by a caching strategy are obviously dependent on the size of the caching area. To this regard, the large amount of memory available on modern multiprocessor systems is an appealing resource for improving the performance of a decomposition technique. One of the innovative features of PGPDT is to implement a parallel gradient updating where both the matrix-vector multiplication and the caching strategy are distributed among the processors. This is done by asking each processor to perform a part of the column combinations required in (3) and to make available its local memory for caching the columns of G. In this way, the gradient updating benefits not only from a computations distribution, but also from a reduction of the kernel evaluations due to much larger caching areas. Of course, these features are not shared by standard decomposition packages, designed to exploit the resources of only one processor. The main steps of the above parallel updating procedure are summarized in Algorithm PGU.

Concerning the reduction of the kernel evaluations, it is worth to recall that the entries of G stored in the caching area can be used also for $G_{\mathcal{B}\mathcal{B}}$ in step A2. Moreover, for the computation of the linear term in (2), the equality

$$G_{\mathcal{B}\mathcal{N}}\alpha_{\mathcal{N}}^{(k)} - 1_{\mathcal{B}} = \nabla \mathcal{F}_{\mathcal{B}}(\alpha^{(k)}) - G_{\mathcal{B}\mathcal{B}}\alpha_{\mathcal{B}}^{(k)}$$

can avoid additional kernel evaluations by exploiting already computed quantities.

The gradient updating overhead within each decomposition iteration can be further reduced by optimizing the kernel computation. Even if a caching strategy can limit the number of kernel evaluations, large problems often require millions of them and their optimization becomes a need. PGPDT uses sparse vector representation of the training examples and exploits the sparseness in the dot products required by the kernel evaluations. Three kernels are available: linear, polynomial

ALGORITHM PGU Parallel gradient updating in step A3 of Algorithm PDT

i) Denote by $W_p,\ p = 1,2,\ldots,N_{\mathrm{P}}$, the caching area of the processor p and by G_i the i-th column of G. Let

$$\mathcal{B}_1 = \left\{ i \in \mathcal{B} \quad | \quad \alpha_i^{(k+1)} - \alpha_i^{(k)} \neq 0 \right\},$$
$$\mathcal{B}_n = \left\{ i \in \mathcal{B}_1 \quad | \quad G_i \notin W_p,\ \ p = 1,2\ldots,N_{\mathrm{P}} \right\}, \qquad \mathcal{B}_c = \mathcal{B}_1 \setminus \mathcal{B}_n.$$

Distribute among the processors the sets $\mathcal{B}_c$ and $\mathcal{B}_n$ and denote by $\mathcal{B}_{c,p}$ and $\mathcal{B}_{n,p}$ the sets of indices assigned to processor p. Make the distribution in such a way that

$$\mathcal{B}_c = \bigcup_{i=1}^{N_{\mathrm{P}}} \mathcal{B}_{c,i}, \qquad \mathcal{B}_{c,i} \cap \mathcal{B}_{c,j} = \emptyset \ \text{ for } \ i \neq j, \qquad \forall i \in \mathcal{B}_{c,p} \ \Rightarrow \ G_i \in W_p,$$
$$\mathcal{B}_n = \bigcup_{i=1}^{N_{\mathrm{P}}} \mathcal{B}_{n,i}, \qquad \mathcal{B}_{n,i} \cap \mathcal{B}_{n,j} = \emptyset \ \text{ for } \ i \neq j$$

and by trying to obtain a well balanced workload among the processors.

ii) $\forall p = 1,2,\ldots,N_{\mathrm{P}}$: use the columns $G_i \in W_p$, $i \in \mathcal{B}_{c,p}$, to compute

$$r_p = \begin{bmatrix} G_{\mathcal{B}\,\mathcal{B}_{c,p}} \\ G_{\mathcal{N}\,\mathcal{B}_{c,p}} \end{bmatrix} \left(\alpha_{\mathcal{B}_{c,p}}^{(k+1)} - \alpha_{\mathcal{B}_{c,p}}^{(k)} \right),$$

then compute the columns $G_i,\ i \in \mathcal{B}_{n,p}$, necessary to obtain

$$r_p \leftarrow r_p + \begin{bmatrix} G_{\mathcal{B}\,\mathcal{B}_{n,p}} \\ G_{\mathcal{N}\,\mathcal{B}_{n,p}} \end{bmatrix} \left(\alpha_{\mathcal{B}_{n,p}}^{(k+1)} - \alpha_{\mathcal{B}_{n,p}}^{(k)} \right)$$

and store in W_p as much as possible of these columns, eventually by substituting those less recently used.

iii) $\forall p = 1,2,\ldots,N_{\mathrm{P}}$: send r_p to all the other processors and assemble a local copy of

$$\nabla \mathcal{F}\left(\alpha^{(k+1)}\right) = \nabla \mathcal{F}\left(\alpha^{(k)}\right) + \sum_{i=1}^{N_{\mathrm{P}}} r_i.$$

ALGORITHM SP1 Selection procedure for step A4.1 of algorithm PDT.

i) Sort the indices of the variables according to $y_i \nabla \mathcal{F}(\alpha^{(k+1)})_i$ in decreasing order and let $I \equiv (i_1, i_2, \ldots, i_n)^T$ be the sorted list (i.e., $y_{i_1} \nabla \mathcal{F}(\alpha^{(k+1)})_{i_1} \geq y_{i_2} \nabla \mathcal{F}(\alpha^{(k+1)})_{i_2} \geq \ldots \geq y_{i_n} \nabla \mathcal{F}(\alpha^{(k+1)})_{i_n}$).

ii) Repeat the selection of a pair $(i_t, i_b) \in I \times I$, with $t < b$, as follows:

- moving down from the top of the sorted list, choose $i_t \in I_{\text{top}}(\alpha^{(k+1)})$,
- moving up from the bottom of the sorted list, choose $i_b \in I_{\text{bot}}(\alpha^{(k+1)})$,

until n_c indices are selected or a pair with the above properties cannot be found.

iii) Let $\bar{\mathcal{B}}$ be the set of the selected indices.

and Gaussian. The interested reader is referred to the available code for more details on their practical implementation. We end this section by remarking that, in case of linear kernel, the updating formula (3) can be simplified in

$$t = \sum_{i \in \mathcal{B}_1} y_i x_i \left(\alpha_i^{(k+1)} - \alpha_i^{(k)} \right), \qquad \mathcal{B}_1 = \left\{ i \in \mathcal{B} \quad | \quad \alpha_i^{(k+1)} - \alpha_i^{(k)} \neq 0 \right\},$$

$$\nabla \mathcal{F}(\alpha^{(k+1)})_j = \nabla \mathcal{F}(\alpha^{(k)})_j + y_j x_j^T t, \qquad j = 1, 2, \ldots, n, \tag{11}$$

and the importance of a caching strategy is generally negligible. Consequently, PGPDT faces linear SVMs without any caching strategy and performs the gradient updating by simply distributing the n tasks (11) among the processors.

2.3 Working Set Selection

In this section we describe how the working set updating in step A4 of the PDT algorithm is implemented within PGPDT. It consists in two phases: in the first phase at most n_c indices are chosen for the new working set by solving the problem (4), while in the second phase at least $n_{\text{sp}} - n_c$ entries are selected from the current $\mathcal{B}$ to complete the new working set. The selection procedure in step A4.1 was first introduced by Joachims (1998) and then rigorously justified by Lin (2001a). In short, by using the notation

$$I_{\text{top}}(\alpha) \equiv \left\{ i \mid (\alpha_i < C \text{ and } y_i = -1) \text{ or } (\alpha_i > 0 \text{ and } y_i = 1) \right\},$$
$$I_{\text{bot}}(\alpha) \equiv \left\{ j \mid (\alpha_j > 0 \text{ and } y_j = -1) \text{ or } (\alpha_j < C \text{ and } y_j = 1) \right\},$$

this procedure can be stated as in Algorithm SP1.

It is interesting to recall how this selection procedure is related to the violation of the first order optimality conditions. For the convex problem (1) the KKT conditions can also be written as

$$\text{a feasible } \alpha^* \text{ is optimal} \quad \Longleftrightarrow \quad \max_{i \in I_{\text{top}}(\alpha^*)} y_i \nabla \mathcal{F}(\alpha^*)_i \leq \min_{j \in I_{\text{bot}}(\alpha^*)} y_j \nabla \mathcal{F}(\alpha^*)_j .$$

ALGORITHM SP2 Selection procedure for step A4.2 of algorithm PDT.

i) Let $\bar{\mathcal{B}}$ be the set of indices selected in step A4.1.

ii) Fill $\bar{\mathcal{B}}$ up to $n_{\rm sp}$ entries by adding the most recent indices[†] $j \in \mathcal{B}$ satisfying $0 < \alpha_j^{(k+1)} < C$; if these indices are not enough, then add the most recent indices $j \in \mathcal{B}$ such that $\alpha_j^{(k+1)} = 0$ and, eventually, the most recent indices $j \in \mathcal{B}$ satisfying $\alpha_j^{(k+1)} = C$.

iii) Set $n_c = \min\{ n_c, \max\{10, J, n_{\rm new}\}\}$, where J is the largest even integer such that $J \leq \frac{n_{\rm sp}}{10}$ and $n_{\rm new}$ is the largest even integer such that $n_{\rm new} \leq \#\{ j,\ j \in \bar{\mathcal{B}} \setminus \mathcal{B}\}$;
set $\mathcal{B} = \bar{\mathcal{B}}$, $k \leftarrow k+1$ and go to step A2.

[†]We mean the indices that are in the working set $\mathcal{B}$ since the lowest number of consecutive iterations.

It means that, given a non-optimal feasible α, there exists at least a pair $(i, j) \in I_{\rm top}(\alpha) \times I_{\rm bot}(\alpha)$ satisfying

$$y_i \nabla \mathcal{F}(\alpha)_i > y_j \nabla \mathcal{F}(\alpha)_j \ .$$

Following Keerthi and Gilbert (2002), these pairs are called *KKT-violating pairs* and, from this point of view, the above selection procedure chooses indices $(i, j) \in I_{\rm top}(\alpha^{(k+1)}) \times I_{\rm bot}(\alpha^{(k+1)})$ by giving priority to those pairs which most violate the optimality conditions. In particular, at each iteration the *maximal-violating pair* is included in the working set: this property is crucial for the asymptotic convergence of a decomposition technique.

From the practical viewpoint, the indices selected via problem (4) identify steepest-like feasible descent directions: this is aimed to get a quick decrease of the objective function $\mathcal{F}(\alpha)$. Nevertheless, for fast convergence, both n_c and the updating phase in step A4.2 have a key relevance. In fact, as it is experimentally shown by Serafini and Zanni (2005), values of n_c equal or close to $n_{\rm sp}$ often yield a dangerous *zigzagging* phenomenon (i.e., some variables enter and leave the working set many times), which can heavily degrade the convergence rate especially for large $n_{\rm sp}$. This drawback suggests to set n_c sufficiently smaller than $n_{\rm sp}$ and then it opens the problem of how to select the remaining indices to fill up the new working set. The studies available in literature on this topic (see Hsu and Lin, 2002; Serafini and Zanni, 2005; Zanghirati and Zanni, 2003, and also the SVM^{light} code) suggest that an efficient approach consists in selecting these indices from the current working set. We recall in Algorithm SP2 the filling strategy recently proposed in (Serafini and Zanni, 2005) and used by the PGPDT software.

The selection policy used by Algorithm SP2 is based on two criteria: the first accords priority to the free variables over the variables at either the lower or the upper bound, the second takes into account *how long* (i.e., how many consecutive decomposition iterations) a variable has been into the working set. Roughly speaking, both the criteria aim to preserve into the working set the variables which are likely to need further optimization. The interested reader can find in the papers by Hsu and Lin (2002) and by Serafini and Zanni (2005) a deeper discussion on these criteria and the computational evidence of their benefits in terms of convergence rate. Finally, Algorithm SP2 also introduces an adaptive reduction of the parameter n_c, useful in case of large sized working sets. This trick allows the decomposition technique to start with n_c close to $n_{\rm sp}$, in order to optimize many new variables in the very first iterations, and avoids zigzagging through the progressive reduction of

n_c. The reduction takes place only if n_c is larger than an empirical threshold and it is controlled via the number of those new indices selected in step A4.1 that do not belong to the current working set.

3. Computational Experiments

The aim of this computational study is to analyse the PGPDT performance. To this end, it is also worth to show that the serial version of the proposed software (called GPDT) can train SVMs with effectiveness comparable to that of the state-of-the-art softwares LIBSVM (ver. 2.8) and SVM^{light} (ver. 6.01). Since there are no other parallel software currently available for comparison, the PGPDT will be evaluated in terms of scaling properties with respect to the serial packages.

Our implementation is an object oriented C^{++} code and its parallel version uses standard MPI communication routines (Message Passing Interface Forum, 1995), hence it is easily portable on many multiprocessor systems. Most of the experiments are carried out on an IBM SP5, which is an IBM SP Cluster 1600 equipped with 64 nodes p5-575 interconnected by a high performance switch (HPS). Each node owns 8 IBM SMP Power5 processors at 1.9GHz and 16GB of RAM (2GB per CPU). The serial packages run on this computer by exploiting only a single CPU. PGPDT has been tested also on different parallel architectures and, for completeness, we report the results obtained on a system where less memory than in the IBM SP5 is available for each CPU: the IBM CLX/1024 Linux Cluster, that owns 512 nodes equipped with two Intel Xeon processors at 3.0GHz and 1GB of RAM per CPU. Both the systems are available at the CINECA Supercomputing center (Bologna, Italy, `http://www.cineca.it`).

The considered softwares are compared on several medium, large and very large test problems generated from well known benchmark data sets, described in the next subsection.

3.1 Test Problems

We trained Gaussian and polynomial SVMs with kernel functions $K(x_i,x_j) = \exp\left(-\|x_i - x_j\|^2/(2\sigma^2)\right)$ and $K(x_i,x_j) = \left(s({x_i}^T x_j)+1\right)^d$, respectively[1].

In what follows we give some details on the databases used for the generation of the training sets, as well as on the SVM parameters we have chosen. Error rates are given as the percentage of misclassifications.

The UCI Adult data set (at `http://www.research.microsoft.com/~jplatt/smo.html`) allows to train an SVM to predict whether a household has an income greater than \$50000. The inputs are 123-dimensional binary sparse vectors with sparsity level $\approx 89\%$. We use the largest version of the data set, sized 32561. We train a Gaussian SVM with training parameters chosen accordingly to the database technical documentation, i.e., $C = 1$ and $\sigma = \sqrt{10}$, that are indicated as those maximizing the performance on a (unavailable) validation set.

The Web data set (available at `http://www.research.microsoft.com/~jplatt/smo.html`) concerns a web page classification problem with a binary representation based on 300 keyword features. On average, the sparsity level of the examples is about 96%. We use the largest version of the data set, sized 49749. We train a Gaussian SVM with the parameters suggested in the data set documentation: $C = 5$ and $\sigma = \sqrt{10}$. As before, these values are claimed to give the best performance on a (unavailable) validation set.

1. Here the notation has the usual meaning: σ is the Gaussian's variance, s is the polynomial scaling parameter and d is the polynomial degree.

The MNIST database of handwritten digits (`http://yann.lecun.com/exdb/mnist`) contains 784-dimensional nonbinary sparse vectors; the data set size is 60000 and the data sparsity is $\approx$ 81%. The provided test set is sized 10000. We train two SVM classifiers for the digit "8" with the following parameters: $C = 10$, $\sigma = 1800$ for the Gaussian kernel and $C = 3000$, $d = 4$, $s = 3 \cdot 10^{-9}$ for the polynomial kernel. This setting gives the following error rates on the test set: 0.55% for the Gaussian kernel and 0.60% for the polynomial kernel.

The Forest Cover Type data set[2] has 581012 samples with 54 attributes, distributed in 8 classes. The average sparsity level of the samples is about 78%. We train some SVM classifiers for separating class 2 from the other classes. The training sets, sized up to 300000, are generated by randomly sampling the data set. We use a Gaussian kernel with $\sigma^2 = 2.5 \cdot 10^4$, $C = 10$. For the largest training set the error rate is about 3.6% on the test set given by the remaining 281012 examples.

The KDDCUP-99 Intrusion Detection data set[3] consists in binary TCP dump data from seven weeks of network traffic. Each original pattern has 34 continuous features and 7 symbolic features. As suggested by Tsang et al. (2005), we normalize each continuous feature to the range $[0,1]$ and transform each symbolic feature to multiple binary features. In this way, the inputs are 122-dimensional sparse vectors with sparsity level $\approx 90\%$. We work with the whole training set sized 4898431 and with some smaller subsets obtained by randomly sampling the original database. We use a Gaussian kernel with parameters $\sigma^2 = (1.2)^{-1}$, $C = 2$. This choice yields error rates of about 7% on the test set of 311029 examples available in the database.

3.2 Serial Behaviour

In the first experiments set, we analyse the behaviour of the serial code on the test problems just described. In Table 1 we report the time in seconds (sec.), the decomposition iteration count (it.) and the number of kernel evaluations in millions (MKernel) required for each one of the considered SVM training packages. The values we use for the working set parameters n_{sp} and n_c are also reported: as mentioned, the LIBSVM software works only with $n_{\text{sp}} = n_c = 2$, whilst both SVM^{light} and GPDT accept larger values. For these two softwares, meaningful ranges of parameters were explored: we report the results corresponding to the pairs that gave the best training time and to the default setting ($n_{\text{sp}} = n_c = 10$ for SVM^{light}, $n_{\text{sp}} = 400$, $n_c = \lfloor n_{\text{sp}}/3 \rfloor = 132$ for GPDT). SVM^{light} is run with several values of n_{sp} in the range $[2,80]$ with both its inner solvers: the Hildreth-D'Esopo and the pr_LOQO. The best training time is obtained by using the Hildreth-D'Esopo solver with n_{sp} small and $n_c = n_{\text{sp}}/2$, generally observing a significant performance decrease for $n_{\text{sp}} > 40$.

We run the codes assigning to the caching area 512MB for the MNIST test problems and 768MB in the other cases; the default threshold $\varepsilon = 10^{-3}$ for the termination criterion is used, except for the two largest Cover Type and KDDCUP-99 test problems, where the stopping tolerance ε is set to 10^{-2}. All the other parameters are assigned default values. This means that both LIBSVM and SVM^{light} benefit from the *shrinking* (Joachims, 1998) strategy that is not implemented in the current release of GPDT.

Table 1 well emphasizes the different approach of the three softwares. In particular we see how GPDT, by exploiting large working sets, converges in far less iterations than the other softwares, but its iterations are much heavier. Looking at the computational time, GPDT seems to be very competitive with respect to both LIBSVM and SVM^{light}. Furthermore, the kernel column highlights

2. Available at `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/covtype`.
3. Available at `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

Data set	n	n_{sp}	n_c	sec.	it.	MKernel
		GPDT				
UCI Adult	32561	400	132	94.1	162	494.2
		400	200	93.6	129	498.5
MNIST (poly)	60000	400	132	379.6	598	424.6
		600	200	345.3	221	324.4
MNIST (Gauss)	60000	400	132	359.2	136	504.8
		2000	300	341.2	22	396.4
Web Pages	49749	400	132	69.6	228	285.5
		600	200	62.2	101	252.9
Cover Type	300000	400	132	24365.5	3730	120846.5
		500	80	21561.4	5018	99880.0
KDDCUP-99	400000	400	132	10239.0	1149	56548.3
		180	60	9190.3	2248	51336.7
		LIBSVM				
UCI Adult	32561	2	2	165.9	15388	452.1
MNIST (poly)	60000	2	2	2154.4	452836	792.0
MNIST (Gauss)	60000	2	2	1081.8	20533	409.4
Web Pages	49749	2	2	64.0	13237	170.3
Cover Type	300000	2	2	17271.7	274092	53152.6
KDDCUP-99	400000	2	2	11220.8	40767	50773.8
		SVM^{light}				
UCI Adult	32561	10	10	216.7	10448	405.1
		20	10	201.1	4317	393.5
		40	20	203.8	2565	410.3
MNIST (poly)	60000	10	10	6454.1	380743	1943.8
		4	2	3090.2	420038	859.8
		8	4	3124.0	238609	905.6
MNIST (Gauss)	60000	10	10	795.6	10262	278.3
		4	2	570.3	18401	204.1
		16	8	562.8	4970	203.8
Web Pages	49749	10	10	108.6	8728	208.5
		4	2	93.8	12195	166.9
		16	8	92.7	4444	188.2
Cover Type	300000	10	10	82892.6	266632	146053.2
		8	4	29902.3	151762	44791.4
		16	8	28585.5	78026	48864.9
KDDCUP-99	400000	10	10	11356.4	21950	23941.3
		8	4	10141.8	28254	21663.6
		20	10	12308.4	20654	24966.0

Table 1: performance of the serial packages on different test problems.

Solver	SV	BSV	$\mathcal{F}_{\text{opt}}$	b	test error
MNIST (poly) test problem					
GPDT	2712	640	−2555033.8	3.54283	0.63%
LIBSVM	2715	640	−2555033.6	3.54231	0.63%
SVM^{light}	2714	640	−2555033.0	3.54213	0.62%
Cover Type test problem					
GPDT	50853	32683	−299399.7	0.22083	3.62%
LIBSVM	51131	32573	−299396.0	0.22110	3.63%
SVM^{light}	51326	32511	−299393.9	0.22149	3.62%

Table 2: accuracy of the serial solvers.

how GPDT benefits from a good optimization of the execution time for the kernel computation: compare, for instance, the results for the MNIST Gaussian test, where the kernel evaluations are very expensive. Here, in front of a number of kernel evaluations similar to LIBSVM and larger than SVM^{light}, a significant lower training time is exhibited. The same consideration holds true for the MNIST polynomial test; however in this case the good GPDT performance is also due to a lower number of kernel evaluations.

The next experiments are intended to underline how the good training time given by GPDT is accompanied by scaling and accuracy properties very similar to the other packages. From the accuracy viewpoint, this is shown for two of the considered test problems by reporting in Table 2 the number of support vectors (SV) and bound support vectors (BSV), the computed optimal value $\mathcal{F}_{\text{opt}}$ of the objective function, the bias b of the separating surface expression[4] (Cristianini and Shawe-Taylor, 2000) and the error rate on the test set.

For what concerns the scaling, Figure 3a shows, for the Cover Type test problem (the worst case for GPDT), the training time with respect to the problem size. All the packages exhibit almost the same dependence that, for this particular data set, seems between quadratic and cubic with respect to the number of examples. For completeness, the number of support vectors of these test problems is also reported in Figure 3b.

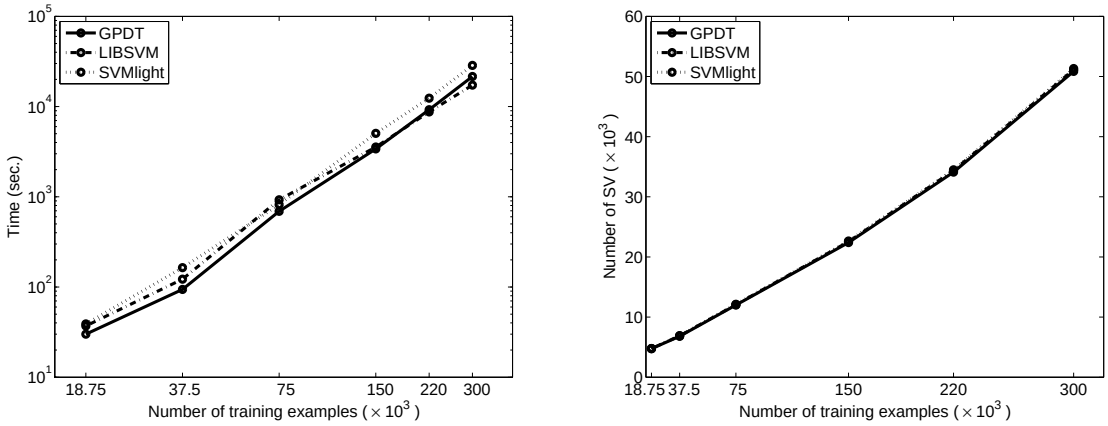
3.3 Parallel Behaviour

The second experiments set concerns with the behaviour of PGPDT. We evaluate PGPDT on the previous four largest problems and some very large problems sized $O(10^6)$ derived from the KDDCUP-99 data set.

3.3.1 LARGE TEST PROBLEMS

For a meaningful comparison against the serial version, PGPDT is run on the MNIST, Cover Type and KDDCUP-99 ($n = 400000$) test problems with the same n_{sp}, n_c and ε parameters as in the previous experiments; furthermore, the same amount of caching area (768MB) is now allocated on each CPU of the IBM SP5. Default values are assigned to the other parameters.

4. The *support vectors* are those samples in the training set corresponding to $\alpha_i^* > 0$; the samples with $\alpha_i^* = C$ are called *bound support vectors*. Roughly speaking, the support vectors are the samples characterizing the hypersurface separating the two classes and the bias b is its displacement.



(a) dependence on n.

(b) number of support vectors.

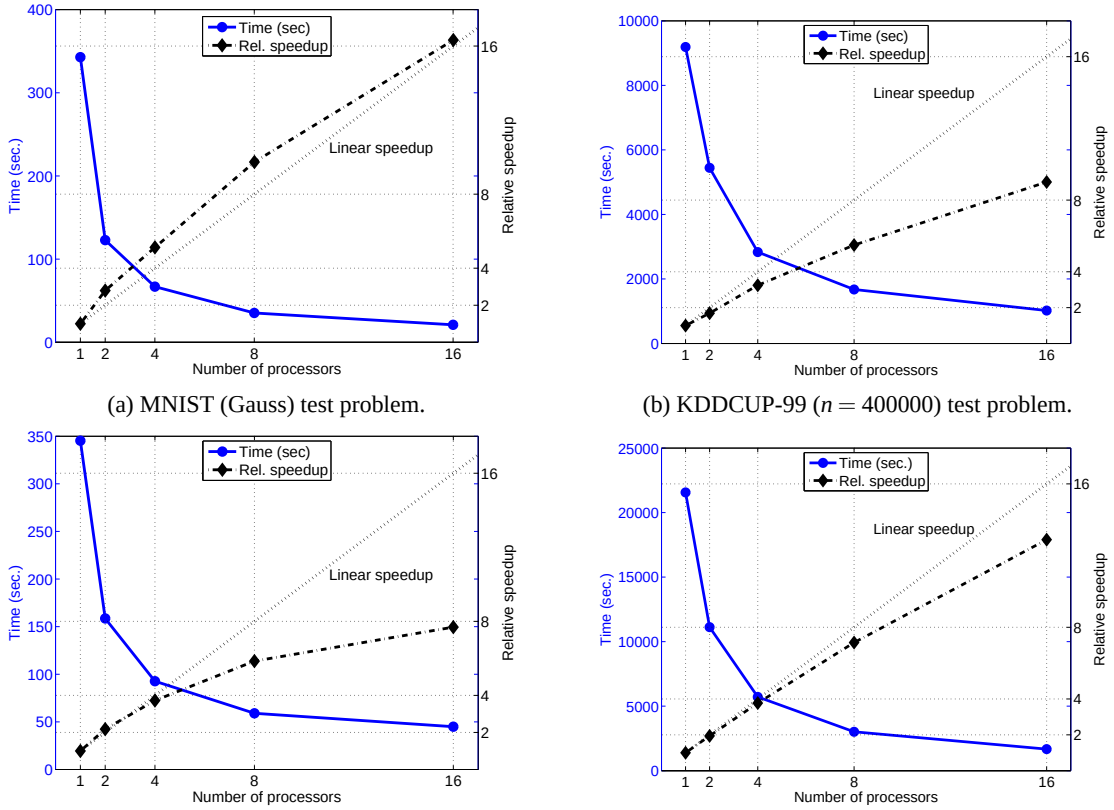
Figure 3: scaling of the serial solvers on test problems from the Cover Type data set.

N_P	sec.	sp_r	it.	MKernel	SV	BSV	$\mathcal{F}_{opt}$
	MNIST (poly) test problem						
1	345.3		221	324.2	2712	640	−2555033.8
2	158.6	2.18	212	249.2	2710	640	−2555033.8
4	100.5	3.44	214	253.9	2711	641	−2555033.8
8	59.7	5.78	212	259.8	2711	641	−2555033.7
16	47.3	7.30	217	271.4	2711	641	−2555033.8
	Cover Type test problem						
1	21561		5018	99880	50853	32683	−299399.7
2	11123	1.94	5047	98925	50786	32673	−299399.8
4	5715	3.77	5059	93597	50786	32668	−299399.9
8	3016	7.15	5086	82853	50832	32664	−299399.9
16	1673	12.89	5029	59439	50826	32697	−299399.9

Table 3: PGPDT scaling on the IBM SP5 system.

Table 3 and Figure 4 summarize the results obtained by running PGPDT on different numbers of processors. We evaluate the parallel performance by the *relative speedup*, defined as $sp_r = T_{\text{serial}}/T_{\text{parallel}}$, where T_{serial} is the training time spent on a single processor, while T_{parallel} denotes the training time on N_P processors.

Seeking clearness, in Table 3 we also report additional information on the overall PGPDT behaviour. In particular, we can see an essentially constant number of decomposition iterations (recall that only the computational burden within the decomposition iteration is distributed) and the same solution accuracy as the serial run (compare the numbers in SV, BSV and $\mathcal{F}_{opt}$ columns). Moreover, remark the lower number of total kernel evaluations needed by the parallel version, due to the growing amount of global caching memory available, which our parallel caching strategy is able to exploit. This is the motivation of the superlinear speedup observed in some situations like the MNIST (Gaussian) test problem (Figure 4a). Unfortunately, there may be cases where the bene-



(a) MNIST (Gauss) test problem.

(b) KDDCUP-99 ($n = 400000$) test problem.

(c) MNIST (poly) test problem.

(d) Cover Type test problem.

Figure 4: PGPDT scaling on the IBM SP5 system.

fits due to the parallel caching strategy are not sufficient to ensure optimal speedups. For instance, sometimes the n_{sp} values that give satisfactory serial performance are not suited for good PGPDT scaling. This is the case of the KDDCUP-99 test problem (Figure 4b), where the small working sets sized $n_{sp} = 180$ imply many decomposition iterations and consequently the fixed costs of the non-distributed tasks (working set selection and stopping rule) become very heavy. Another example is provided by the MNIST (polynomial) test problem (Figure 4c): here the subproblem solution is a dominant task in comparison to the gradient updating and the suboptimal scaling of the PGPM solver on 16 processors leads to poor speedups. However, also in these cases remarkable time reductions are observed in comparison with the serial softwares (see Table 1).

We further remark that all these considerations are quite dependent on the underlying parallel architecture. In particular, on multiprocessor systems where less memory than in the SP5 platform is available for each CPU, even better speedups can be expected due to the effectiveness of the parallel caching strategy. For instance, we report in Figure 5 what we get for the KDDCUP-99 test problem on the IBM CLX/1024 Linux Cluster, where only 400MB of caching area can be allocated on each CPU. Due to both the worse performance of this machine and the reduced caching area, larger training time is required, but an optimal PGPDT speedup is now observed up to 16 processors.

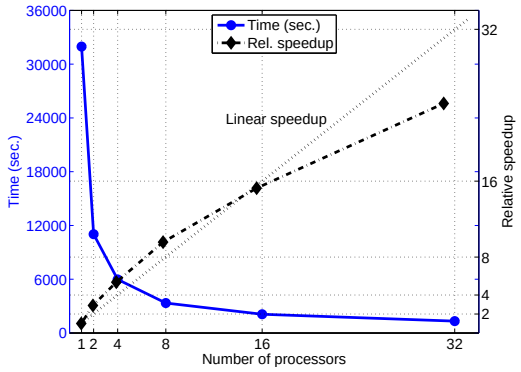
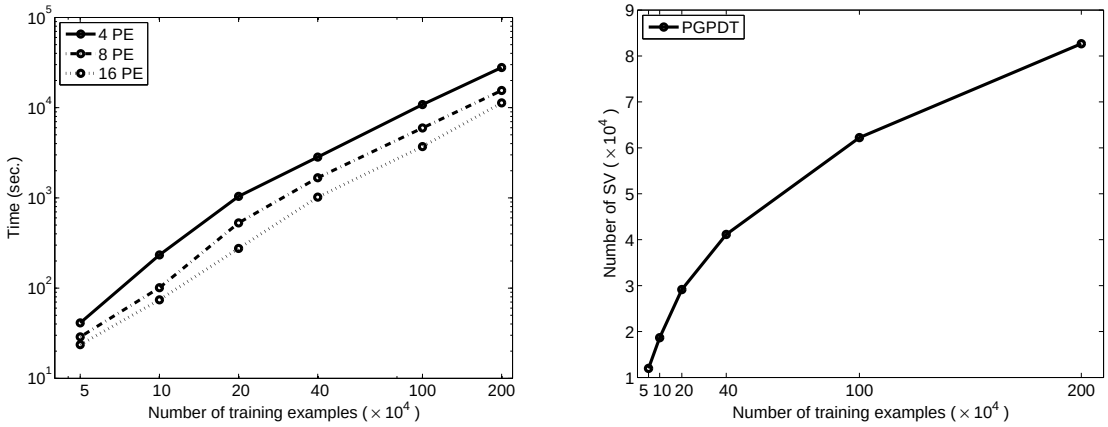


Figure 5: PGPDT scaling on the CLX/1024 system for the KDDCUP-99 ($n = 4 \cdot 10^5$) test problem.



(a) dependence on n.

(b) number of support vectors.

Figure 6: Parallel training time for different sizes of the KDDCUP-99 test probelms

3.3.2 VERY LARGE TEST PROBLEMS

In this section we present the behavior of the PGPDT code on very large test problems. In particular we considered three test problems from the KDDCUP-99 data set of size $n = 10^6$, $2 \cdot 10^6$ and 4898431, the latter being the full data set size. The test problems are obtained by training Gaussian SVMs with the parameters setting previously used for this data set.

In the two larger cases a different setting for the n_{sp}, n_c and caching area have been used. In particular, for the case $n = 2 \cdot 10^6$ we used $n_{\text{sp}} = 150$, $n_c = 40$ and 600Mb of caching area; for the full case we used $n_{\text{sp}} = 90$, $n_c = 30$ and 250Mb of caching area. The reason for reducing the caching area is that every processor can allocate no more that 1.7Gb of memory and, when the data set size increases, most of the memory is used for storing the training data and cannot be used for caching.

These $O(10^6)$ test problems are firstly used to study how the PGPDT time complexity scales with the size of the data sets. In Figure 6a the training time is reported for 4, 8 and 16 processors. Figure 6b shows the growth rate of the support vectors for these test problems. It can be observed that the scaling is close to $O(n^2)$, as often exhibited by the serial state-of-the-art decom-

position packages (Collobert and Bengio, 2001; Joachims, 1998). This result is quite natural if we remember that PGPDT is based on a parallelization of each iteration of a standard decomposition technique. Concerning the subquadratic scaling exhibited for increasing sizes, it can be motivated by the sublinear growth of the support vectors observed on these experiments; however, in different situations it may be expected a training time complexity that scales at least quadratically (see, for instance, the experiments on the Cover Type data set described in Figure 3).

Table 4 shows the PGPDT performance in terms of training time and accuracy for different number of processors. Here, the time is measured in hours and minutes and the kernel evaluations are expressed in billions. For the test problem sized $n = 2 \cdot 10^6$, the serial results concern only the GPDT because LIBSVM exceeded the time limit of 60 hours and SVM^{light} stopped without a valid solution after relaxing the KKT conditions. Due to the very large size of the problem, the amount of 600MB for the caching area seems not sufficient to prevent a huge number of kernel evaluations in the serial run. Again, this drawback is reduced in the multiprocessor runs, due to increased memory for caching. Thus, analogously to some previous experiments (see Figures 4a, 5), superlinear speedup is exhibited, in this case up to about 20 processors. The largest test problem, with size about 5 millions and more than 10^5 support vectors, can be faced in a reasonable time only with the parallel version. In this case the overall remark is that, on the considered architecture, few processors allow to train the Gaussian SVM in less than one day while few tens of processors can be exploited to reduce the training time to about 10 hours.

Finally, by observing in Table 4 the column of the objective function values, we may confirm that also in these experiments the training time saving ensured by PGPDT is obtained without damaging the solution accuracy.

These results show that PGPDT is able to exploit the resources of today multiprocessor systems to overcome the limits of the serial SVM implementations in solving $O(10^6)$ problems (see also the training time in Figure 6a). As already mentioned, there is no other available parallel software to perform a fair comparison on the same architecture and the same data; however, an indirect comparison with the results reported by Graf et al. (2005) for the cascade algorithm suggests that PGPDT could be really competitive. Furthermore, since the cascade algorithm and PGPDT exploit very different parallelization ideas (recall that the former is based on the distribution of smaller independent SVMs), promising improvements could be achieved by an appropriate combination of the two approaches.

4. Conclusions and Future Work

Parallel software to train linear and nonlinear SVMs for classification problems is presented, which is suitable for distributed memory multiprocessors systems. It implements an iterative decomposition technique based on a gradient projection solver for the inner subproblems. At each decomposition iteration, the heaviest tasks, i.e., solving the subproblem and updating the gradient, are distributed among the available processors. Furthermore, a parallel caching strategy allows to effectively exploit as much memory as available to avoid expensive kernel recomputations. Numerical comparisons with the state-of-the-art softwares LIBSVM and SVM^{light} on benchmark problems show the significant speedup that the proposed parallel package can achieve in training large scale SVMs. In short, experiments on $O(10^6)$ data sets show that nonlinear SVMs with $O(10^5)$ support vectors can be trained in few hours by exploiting some tens of processors. Thus, this parallel package, available at `http://dm.unife.it/gpdt`, can be a useful tool for overcoming the limits of the

N_P	time	it.	GKernel	SV	BSV	$\mathcal{F}_{opt}$
	$n = 2 \cdot 10^6$					
1	$54^h\ 59^m$	6192	1135.8	82521	466	−9625.9
2	$14^h\ 22^m$	6077	468.5	84565	463	−9625.8
4	$7^h\ 44^m$	6005	458.1	82193	464	−9625.7
8	$4^h\ 18^m$	6064	462.9	82723	462	−9625.8
16	$3^h\ 08^m$	6116	467.0	84100	460	−9625.9
24	$2^h\ 47^m$	6202	473.0	83626	464	−9626.0
	$n = 4898431$					
8	$19^h\ 08^m$	12300	1752.9	131041	1021	−14479.6
16	$12^h\ 16^m$	12295	1739.7	130918	1046	−14479.6
32	$9^h\ 22^m$	12310	1742.9	131736	1017	−14479.6

Table 4: PGPDT scaling on very large test problems from the KDDCUP-99 data set.

serial SVM implementation currently available. The main improvements will concern: (i) the optimization/distribution of the tasks which are not currently parallelized, to improve the scalability; (ii) the introduction of a shrinking strategy, for further reducing the number of kernel evaluations; (iii) the inner solver robustness, to better face the subproblems arising from badly scaled training data. Furthermore, work is in progress to include in a new PGPDT release a suitable data distribution and the extension to regression problems.

Acknowledgments

The authors are grateful to the staff of the CINECA Supercomputing Center (Bologna, Italy) for supporting us with their valuable expertise. The authors also thank the anonymous referees for their helpful comments.

References

Jonathan Barzilai and Jonathan M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, second edition, 1999.

Ernesto G. Birgin, José Mario Martínez, and Marcos Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

Bernhard E. Boser, Isabelle Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In David Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, Pittsburgh, PA, 1992.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001. URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Pai-Hsuen Chen, Rong-En Fan, and Chih-Jen Lin. A study on SMO-type decomposition methods for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2005. To appear on IEEE Transaction on Neural Networks, 2006.

Ronan Collobert and Samy Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5):1105–1114, 2002.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

Yu-Hong Dai and Roger Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming*, 106(3):403–421, 2006.

Yu-Hong Dai and Roger Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100(1):21–47, 2005.

Jian-Xiong Dong, Adam Krzyzak, and Ching Y. Suen. A fast parallel optimization for training support vector machine. In P. Perner and A. Rosenfeld, editors, *Proceedings of 3rd International Conference on Machine Learning and Data Mining*, volume 17, pages 96–105. Springer Lecture Notes in Artificial Intelligence, Leipzig, Germany, 2003.

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

Hans Peter Graf, Eric Cosatto, Léon Bottou, Igor Dourdanovic, and Vladimir N. Vapnik. Parallel support vector machines: the Cascade SVM. In Lawrence Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.

Chih-Wei Hsu and Chih-Jen Lin. A simple decomposition method for support vector machines. *Machine Learning*, 46:291–314, 2002.

Don Hush and Clint Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51:51–71, 2003.

Throstem Joachims. Making large-scale SVM learning practical. In Bernard Schölkopf, C.J.C. Burges, and Alex Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, Cambridge, MA, 1998.

S. Sathiya Keerthi and Elmer G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46:351–360, 2002.

Chih-Jen Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:1288–1298, 2001a.

Chih-Jen Lin. Linear convergence of a decomposition method for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001b.

Chih-Jen Lin. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Transactions on Neural Networks*, 13:248–250, 2002.

Message Passing Interface Forum. MPI: A message-passing interface standard (version 1.2). *International Journal of Supercomputing Applications*, 8(3/4), 1995. URL `http://www.mpi-forum.org`. Also available as Technical Report CS-94-230, Computer Science Dept., University of Tennesse, Knoxville, TN.

Bruce A. Murtagh and Michael A. Saunders. MINOS 5.5 user's guide. Technical report, Department of Operation Research, Stanford University, Stanford CA, 1998.

Edgar Osuna, Robert Freund, and Girosi Federico. Training support vector machines: an application to face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR97)*, pages 130–136. IEEE Computer Society, New York, 1997.

Laura Palagi and Marco Sciandrone. On the convergence of a modified version of SVM^{light} algorithm. *Optimization Methods and Software*, 20:317–334, 2005.

Panos M. Pardalos and Naina Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46:321–328, 1990.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernard Schölkopf, C.J.C. Burges, and Alex Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, Cambridge, MA, 1998.

Valeria Ruggiero and Luca Zanni. A modified projection algorithm for large strictly convex quadratic programs. *Journal of Optimization Theory and Applications*, 104(2):281–299, 2000a.

Valeria Ruggiero and Luca Zanni. Variable projection methods for large convex quadratic programs. In Donato Trigiante, editor, *Recent Trends in Numerical Analysis*, volume 3 of *Advances in the Theory of Computational Mathematics*, pages 299–313. Nova Science Publisher, 2000b.

Thomas Serafini and Luca Zanni. On the working set selection in gradient projection-based decomposition techniques for support vector machines. *Optimization Methods and Software*, 20: 583–596, 2005.

Thomas Serafini, Gaetano Zanghirati, and Luca Zanni. Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20:353–378, 2005.

Alex J. Smola. pr_LOQO optimizer, 1997. URL `http://www.kernel-machines.org/code/prloqo.tar.gz`.

Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6(4):363–392, 2005.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

Gaetano Zanghirati and Luca Zanni. A parallel solver for large quadratic programs in training support vector machines. *Parallel Computing*, 29:535–551, 2003.

Luca Zanni. An improved gradient projection-based decomposition technique for support vector machines. *Computational Management Science*, 3(2):131–145, 2006.

Journal of Machine Learning Research 7 (2006) 1493–1515 Submitted 10/05; Revised 3/06; Published 7/06

Building Support Vector Machines with Reduced Classifier Complexity

S. Sathiya Keerthi SELVARAK@YAHOO-INC.COM
Yahoo! Research
3333 Empire Avenue, Building 4
Burbank, CA 91504, USA

Olivier Chapelle CHAPELLE@TUEBINGEN.MPG.DE
MPI for Biological Cybernetics
72076 Tübingen, Germany

Dennis DeCoste DECOSTED@YAHOO-INC.COM
Yahoo! Research
3333 Empire Avenue, Building 4
Burbank, CA 91504, USA

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

Support vector machines (SVMs), though accurate, are not preferred in applications requiring great classification speed, due to the number of support vectors being large. To overcome this problem we devise a primal method with the following properties: (1) it decouples the idea of basis functions from the concept of support vectors; (2) it greedily finds a set of kernel basis functions of a specified maximum size ($d_{\max}$) to approximate the SVM primal cost function well; (3) it is efficient and roughly scales as $O(nd_{\max}^2)$ where n is the number of training examples; and, (4) the number of basis functions it requires to achieve an accuracy close to the SVM accuracy is usually far less than the number of SVM support vectors.

Keywords: SVMs, classification, sparse design

1. Introduction

Support Vector Machines (SVMs) are modern learning systems that deliver state-of-the-art performance in real world pattern recognition and data mining applications such as text categorization, hand-written character recognition, image classification and bioinformatics. Even though they yield very accurate solutions, they are not preferred in online applications where classification has to be done in great speed. This is due to the fact that a large set of basis functions is usually needed to form the SVM classifier, making it complex and expensive. In this paper we devise a method to overcome this problem. Our method incrementally finds basis functions to maximize accuracy. The process of adding new basis functions can be stopped when the classifier has reached some limiting level of complexity. In many cases, our method efficiently forms classifiers which have an order of magnitude smaller number of basis functions compared to the full SVM, while achieving nearly the same level of accuracy.

SVM solution and post-processing simplification Given a training set $\{(x_i, y_i)\}_{i=1}^n$, $y_i \in \{1, -1\}$, the Support Vector Machine (SVM) algorithm with an L_2 penalization of the training errors consists

of solving the following primal problem

$$\min \frac{\lambda}{2}\|w\|^2 + \frac{1}{2}\sum_{i=1}^{n} \max(0, 1 - y_i w \cdot \phi(x_i))^2. \tag{1}$$

Computations involving ϕ are handled using the kernel function, $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. For convenience the bias term has not been included, but the analysis presented in this paper can be extended in a straightforward way to include it. The quadratic penalization of the errors makes the primal objective function continuously differentiable. This is a great advantage and becomes necessary for developing a primal algorithm, as we will see below.

The standard way to train an SVM is to introduce Lagrange multipliers α_i and optimize them by solving a dual problem. The classifier function for a new input x is then given by the sign of $\sum_i \alpha_i y_i k(x, x_i)$. Because there is a flat part in the loss function, the vector α is usually sparse. The x_i for which $\alpha_i \neq 0$ are called *support vectors (SVs)*. Let n_{SV} denote the number of SVs for a given problem. A recent theoretical result by Steinwart (Steinwart, 2004) shows that n_{SV} grows as a linear function of n. Thus, for large problems, this number can be large and the training and testing complexities might become prohibitive since they are respectively, $O(n\, n_{\mathrm{SV}} + {n_{\mathrm{SV}}}^3)$ and $O(n_{\mathrm{SV}})$.

Several methods have been proposed for reducing the number of support vectors. Burges and Schölkopf (1997) apply nonlinear optimization methods to seek sparse representations after building the SVM classifier. Along similar lines, Schölkopf et al. (1999) use L_1 regularization on β to obtain sparse approximations. These methods are expensive since they involve the solution of hard non-convex optimization problems. They also become impractical for large problems. Downs et al. (2001) give an exact algorithm to prune the support vector set after the SVM classifier is built. Thies and Weber (2004) give special ideas for the quadratic kernel. Since these methods operate as a post-processing step, an expensive standard SVM training is still required.

Direct simplification via basis functions and primal Instead of finding the SVM solution by maximizing the dual problem, one approach is to *directly minimize the primal form* after invoking the representer theorem to represent w as

$$w = \sum_{i=1}^{n} \beta_i \phi(x_i). \tag{2}$$

If we allow $\beta_i \neq 0$ for all i, substitute (2) in (1) and solve for the β_i's then (assuming uniqueness of solution) we will get $\beta_i = y_i \alpha_i$ and thus we will precisely retrieve the SVM solution (Chapelle, 2005). But our aim is to obtain approximate solutions that have as few non-zero β_i's as possible. For many classification problems there exists a small subset of the basis functions[1] suited to the complexity of the problem being solved, irrespective of the training size growth, that will yield pretty much the same accuracy as the SVM classifier. The evidence for this comes from the empirical performance of other sparse kernel classifiers: the Relevance Vector Machine (Tipping, 2001), Informative Vector Machine (Lawrence et al., 2003) are probabilistic models in a Bayesian setting; and Kernel Matching Pursuit (Vincent and Bengio, 2002) is a discriminative method that is mainly developed for the least squares loss function. These recent non-SVM works have laid the claim that they can match the accuracy of SVMs, while also bringing down considerably, the number of basis functions as well as the training cost. Work on simplifying SVM solution has not caught up well

1. Each $k(x, x_i)$ will be referred to as a basis function.

with those works in related kernel fields. The method outlined in this paper makes a contribution to fill this gap.

We deliberately use the variable name, β_i in (2) so as to interpret it as a basis weight as opposed to viewing it as $y_i\alpha_i$ where α_i is the Lagrange multiplier associated with the i-th primal slack constraint. While the two are (usually) one and the same at exact optimality, they can be very different when we talk of sub-optimal primal solutions. There is a lot of freedom when we simply think of the β_i's as basis weights that yield a good suboptimal w for (1). First, we do not have to put any bounds on the β_i. Second, we do not have to think of a β_i corresponding to a particular location relative to the margin planes to have a certain value. Going even one more step further, we do not even have to restrict the basis functions to be a subset of the training set examples.

Osuna and Girosi (1998) consider such an approach. They achieve sparsity by including the L_1 regularizer, $\lambda_1\|\beta\|_1$ in the primal objective. But they do not develop an algorithm (for solving the modified primal formulation and for choosing the right λ_1) that scales efficiently to large problems.

Wu et al. (2005) write w as

$$w = \sum_{i=1}^{l} \beta_i \phi(\tilde{x}_i)$$

where l is a chosen small number and optimize the primal objective with the β_i as well as the $\tilde{x}_i$ as variables. But the optimization can become unwieldy if l is not small, especially since the optimization of the $\tilde{x}_i$ is a hard non-convex problem.

In the RSVM algorithm (Lee and Mangasarian, 2001; Lin and Lin, 2003) a random subset of the training set is chosen to be the $\tilde{x}_i$ and then only the β_i are optimized.[2] Because basis functions are chosen randomly, this method requires many more basis functions than needed in order to achieve a level of accuracy close to the full SVM solution; see Section 3.

A principled alternative to RSVM is to use a greedy approach for the selection of the subset of the training set for forming the representation. Such an approach has been popular in Gaussian processes (Smola and Bartlett, 2001; Seeger et al., 2003; Keerthi and Chu, 2006). Greedy methods of basis selection also exist in the boosting literature (Friedman, 2001; Rätsch, 2001). These methods entail selection from a continuum of basis functions using either gradient descent or linear programming column generation. Bennett et al. (2002) and Bi et al. (2004) give modified ideas for kernel methods that employ a set of basis functions fixed at the training points.

Particularly relevant to the work in this paper are the kernel matching pursuit (KMP) algorithm of Vincent and Bengio (2002) and the growing support vector classifier (GSVC) algorithm of Parrado-Hernández et al. (2003). KMP is an effective greedy discriminative approach that is mainly developed for least squares problems. GSVC is an efficient method that is developed for SVMs and uses a heuristic criterion for greedy selection of basis functions.

Our approach *The main aim of this paper is to give an effective greedy method SVMs which uses a basis selection criterion that is directly related to the training cost function and is also very efficient.* The basic theme of the method is forward selection. It starts with an empty set of basis functions and greedily chooses new basis functions (from the training set) to improve the primal objective function. We develop efficient schemes for both, the greedy selection of a new basis function, as well as the optimization of the β_i for a given selection of basis functions. For choosing upto $d_{\max}$ basis functions, the overall compuational cost of our method is $O(nd_{\max}^2)$. The different

2. For convenience, in the RSVM method, the SVM regularizer is replaced by a simple L_2 regularizer on β.

	SpSVM-2		SVM	
Data Set	TestErate	#Basis	TestErate	n_{SV}
Banana	10.87 (1.74)	17.3 (7.3)	10.54 (0.68)	221.7 (66.98)
Breast	29.22 (2.11)	12.1 (5.6)	28.18 (3.00)	185.8 (16.44)
Diabetis	23.47 (1.36)	13.8 (5.6)	23.73 (1.24)	426.3 (26.91)
Flare	33.90 (1.10)	8.4 (1.2)	33.98 (1.26)	629.4 (29.43)
German	24.90 (1.50)	14.0 (7.3)	24.47 (1.97)	630.4 (22.48)
Heart	15.50 (1.10)	4.3 (2.6)	15.80 (2.20)	166.6 (8.75)
Ringnorm	1.97 (0.57)	12.9 (2.0)	1.68 (0.24)	334.9 (108.54)
Thyroid	5.47 (0.78)	10.6 (2.3)	4.93 (2.18)	57.80 (39.61)
Titanic	22.68 (1.88)	3.3 (0.9)	22.35 (0.67)	150.0 (0.0)
Twonorm	2.96 (0.82)	8.7 (3.7)	2.42 (0.24)	330.30 (137.02)
Waveform	10.66 (0.99)	14.4 (3.3)	10.04 (0.67)	246.9 (57.80)

Table 1: Comparison of *SpSVM-2* and *SVM* on benchmark data sets from (Rätsch). For TestErate, #Basis and n_{SV}, the values are means over ten different training/test splits and the values in parantheses are the standard deviations.

components of the method that we develop in this paper are not new in themselves and are inspired from the above mentioned papers. However, from a practical point of view, it is not obvious how to combine and tune them in order to get a very efficient SVM training algorithm. That is what we achieved in this paper through numerous and careful experiments that validated the techniques employed.

Table 1 gives a preview of the performance of our method (called *SpSVM-2* in the table) in comparison with SVM on several UCI data sets. As can be seen there, our method gives a competing generalization performance while reducing the number of basis functions very significantly. (More specifics concerning Table 1 will be discussed in Section 4.)

The paper is organized as follows. We discuss the details of the efficient optimization of the primal objective function in Section 2. The key issue of selecting basis functions is taken up in Section 3. Sections 4-7 discuss other important practical issues and give computational results that demonstrate the value of our method. Section 8 gives some concluding remarks. The appendix gives details of all the data sets used for the experiments in this paper.

2. The Basic Optimization

Let $J \subset \{1,\ldots,n\}$ be a given index set of basis functions that form a subset of the training set. We consider the problem of minimizing the objective function in (1) over the set of vectors w of the form[3]

$$w = \sum_{j \in J} \beta_j \phi(x_j). \tag{3}$$

3. More generally, one can consider expansion on points which do not belong to the training set.

2.1 Newton Optimization

Let $K_{ij} = k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ denote the generic element of the $n \times n$ kernel matrix K. The notation K_{IJ} refers to the submatrix of K made of the rows indexed by I and the columns indexed by J. Also, for a n-dimensional vector p, let p_J denote the $|J|$ dimensional vector containing $\{p_j : j \in J\}$.

Let $d = |J|$. With w restricted to (3), the primal problem (1) becomes the d dimensional minimization problem of finding β_J that solves

$$\min_{\beta_J} f(\beta_J) = \frac{\lambda}{2}\beta_J^\top K_{JJ}\beta_J + \frac{1}{2}\sum_{i=1}^{n}\max(0, 1 - y_i o_i)^2 \tag{4}$$

where $o_i = K_{i,J}\beta_J$. Except for the regularizer being more general, i.e., $\beta_J^\top K_{JJ}\beta_J$ (as opposed to the simple regularizer, $\|\beta_J\|^2$), the problem in (4) is very much the same as in a linear SVM design. Thus, the Newton method and its modification that are developed for linear SVMs (Mangasarian, 2002; Keerthi and DeCoste, 2005) can be used to solve (4) and obtain the solution β_J.

Newton Method

1. Choose a suitable starting vector, β_J^0. Set $k = 0$.
2. If β_J^k is the optimal solution of (4), stop.
3. Let $I = \{i : 1 - y_i o_i \geq 0\}$ where $o_i = K_{i,J}\beta_J^k$ is the output of the i-th example. Obtain $\bar{\beta}_J$ as the result of a Newton step or equivalently as the solution of the regularized least squares problem,
$$\min_{\beta_J} \frac{\lambda}{2}\beta_J^\top K_{JJ}\beta_J + \frac{1}{2}\sum_{i\in I}(1 - y_i K_{i,J}\beta_J)^2. \tag{5}$$
4. Take β_J^{k+1} to be the minimizer of f on L, the line joining β_J^k and $\bar{\beta}_J$. Set $k := k + 1$ and go back to step 2 for another iteration.

The solution of (5) is given by

$$\bar{\beta}_J = \beta_J^k - P^{-1}g, \quad \text{where} \quad P = \lambda K_{JJ} + K_{JI}K_{JI}^\top \quad \text{and} \quad g = \lambda K_{JJ}\beta_J - K_{JI}(y_I - o_I). \tag{6}$$

P and g are also the (generalized) Hessian and gradient of the objective function (4).

Because the loss function is piecewise quadratic, Newton method converges in a finite number of iterations. The number of iterations required to converge to the exact solution of (4) is usually very small (less than 5). Some Matlab code is available online at `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/primal`.

2.2 Updating the Hessian

As already pointed out in Section 1, we will mainly need to solve (4) in an incremental mode:[4] with the solution β_J of (4) already available, solve (4) again, but with one more basis function added, i.e., J incremented by one. Keerthi and DeCoste (2005) show that the Newton method is very efficient

4. In our method basis functions are added one at a time.

for such seeding situations. Since the kernel matrix is dense, we maintain and update a Cholesky factorization of P, the Hessian defined in (6). Even with J fixed, during the course of solving (4) via the Newton method, P will undergo changes due to changes in I. Efficient rank one schemes can be used to do the updating of the Cholesky factorization (Seeger, 2004). The updatings of the factorization of P that need to be done because of changes in I are not going to be expensive because such changes mostly occur when J is small; when J is large, I usually undergoes very small changes since the set of training errors is rather well identified by that stage. Of course P and its factorization will also undergo changes (their dimensions increase by one) each time an element is added to J. This is a routine updating operation that is present in most forward selection methods.

2.3 Computational Complexity

It is useful to ask: what is the complexity of the incremental computations needed to solve (4) when its solution is available for some J, at which point one more basis element is included in it and we want to re-solve (4)? In the best case, when the support vector set I does not change, the cost is mainly the following: computing the new row and column of K_{JJ} ($d+1$ kernel evaluations); computing the new row of K_{JI} (n kernel computations);[5] computing the new elements of P ($O(nd)$ cost); and the updating of the factorization of P ($O(d^2)$ cost). Thus the cost can be summarized as: $(n+d+1)$ kernel evaluations and $O(nd)$ cost. Even when I does change and so the cost is more, it is reasonable to take the above mentioned cost summary as a good estimate of the cost of the incremental work. Adding up these costs till $d_{\max}$ basis functions are selected, we get a complexity of $O(nd_{\max}^2)$. Note that this is the basic cost given that we already know the sequence of $d_{\max}$ basis functions that are to be used. Thus, $O(nd_{\max}^2)$ is also the complexity of the method in which basis functions are chosen randomly. In the next section we discuss the problem of selecting the basis functions systematically and efficiently.

3. Selection of New Basis Element

Suppose we have solved (4) and obtained the minimizer β_J. Obviously, the minimum value of the objective function in (4) (call it f_J) is greater than or equal to $f^\star$, the optimal value of (1). If the difference between them is large we would like to continue on and include another basis function. Take one $j \notin J$. How do we judge its value of inclusion? The best scoring mechanism is the following one.

3.1 Basis Selection Method 1

Include j in J, optimize (4) fully using (β_J, β_j), and find the improved value of the objective function; call it $\tilde{f}_j$. Choose the j that gives the least value of $\tilde{f}_j$. We already analyzed in the earlier section that the cost of doing one basis element inclusion is $O(nd)$. So, if we want to try all elements outside J, the cost is $O(n^2 d)$; the overall cost of such a method of selecting $d_{\max}$ basis functions is $O(n^2 d_{\max}^2)$, which is much higher than the basic cost, $O(nd_{\max}^2)$ mentioned in the previous section. Instead, if we work only with a random subset of size κ chosen from outside J, then the cost in one basis selection step comes down to $O(\kappa nd)$, and the overall cost is limited to $O(\kappa nd_{\max}^2)$. Smola and Bartlett (2001) have successfully tried such random subset choices for Gaussian process regression, using $\kappa = 59$. However, note that, even with this scheme, the cost of new basis selection ($O(\kappa nd)$)

5. In fact this is not n but the size of I. Since we do not know this size, we upper bound it by n.

is still disproportionately higher (by κ times) than the cost of actually including the newly selected basis function ($O(nd)$). Thus we would like to go for cheaper methods.

3.2 Basis Selection Method 2

This method computes a score for a new element j in $O(n)$ time. The idea has a parallel in Vincent and Bengio's work on Kernel Matching Pursuit (Vincent and Bengio, 2002) for least squares loss functions. They have two methods called *prefitting* and *backfitting*; see equations (7), (3) and (6) of Vincent and Bengio (2002).[6] Their *prefitting* is parallel to *Basis Selection Method 1* that we described earlier. The cheaper method that we suggest below is parallel to their *backfitting* idea. Suppose β_J is the solution of (4). Including a new element j and its corresponding variable, β_j yields the problem of minimizing

$$\frac{\lambda}{2}(\beta_J^\top \ \beta_j)\begin{pmatrix} K_{JJ} & K_{Jj} \\ K_{jJ} & K_{jj} \end{pmatrix}\begin{pmatrix} \beta_J \\ \beta_j \end{pmatrix} + \frac{1}{2}\sum_{i=1}^{n}\max(0, 1 - y_i(K_{iJ}\beta_J + K_{ij}\beta_j)^2, \tag{7}$$

We fix β_J and optimize (7) using only the new variable β_j and see how much improvement in the objective function is possible in order to define the score for the new element j.

This one dimensional function is piecewise quadratic and can be minimized exactly in $O(n \log n)$ time by a dichotomy search on the different breakpoints. But, a very precise calculation of the scoring function is usually unnecessary. So, for practical solution we can simply do a few Newton-Raphson-type iterations on the derivative of the function and get a near optimal solution in $O(n)$ time. Note that we also need to compute the vector K_{Jj}, which requires d kernel evaluations. Though this cost is subsumed in $O(n)$, it is a factor to remember if kernel evaluations are expensive.

If all $j \notin J$ are tried, then the complexity of selecting a new basis function is $O(n^2)$, which is disproportionately large compared to the cost of including the chosen basis function, which is $O(nd)$. Like in *Basis Selection Method 1*, we can simply choose κ random basis functions to try. If $d_{\max}$ is specified, one can choose $\kappa = O(d_{\max})$ without increasing the overall complexity beyond $O(nd_{\max}^2)$. More complex schemes incorporating a kernel cache can also be tried.

3.3 Kernel Caching

For upto medium size problems, say $n < 15,000$, it is a good idea to have cache for the entire kernel matrix. If additional memory space is available and, say a Gaussian kernel is employed, then the values of $\|x_i - x_j\|^2$ can also be cached; this will help significantly reduce the time associated with the tuning of hyperparameters. For larger problems, depending on memory space available, it is a good idea to cache as many as possible, full kernel rows corresponding to j that get tried, but do not get chosen for inclusion. It is possible that they get called in a later stage of the algorithm, at which time, this cache can be useful. It is also possible to think of variations of the method in which full kernel rows corresponding to a large set (as much that can fit into memory) of randomly chosen training basis is pre-computed and only these basis functions are considered for selection.

3.4 Shrinking

As basis functions get added, the SVM solution w and the margin planes start stabilizing. If the number of support vectors form a small fraction of the training set, then, for a large fraction of

6. For least squares problems, Adler et al. (1996) had given the same ideas as Vincent and Bengio in earlier work.

(well-classified) training examples, we can easily conclude that they will probably never come into the active set I. Such training examples can be left out of the calculations without causing any undue harm. This idea of shrinking has been effectively used to speed-up SVM training (Joachims, 1999; Platt, 1998).

3.5 Experimental Evaluation

We now evaluate the performance of basis selection methods 1 and 2 (we will call them as *SpSVM-1*, *SpSVM-2*) on some sizable benchmark data sets. A full description of these data sets and the kernel functions used is given in the appendix. The value of $\kappa = 59$ is used. To have a baseline, we also consider the method, *Random* in which the basis functions are chosen randomly. This is almost the same as the RSVM method (Lee and Mangasarian, 2001; Lin and Lin, 2003), the only difference being the regularizer ($\beta_J^\top K_{J,J}\beta_J$ in (4) versus $\|\beta_J\|^2$ in RSVM). For another baseline we consider the (more systematic) unsupervised learning method in which an incomplete Cholesky factorization with pivoting (Meijerink and van der Vorst, 1977; Bach and Jordan, 2005) is used to choose basis functions.[7] For comparison we also include the GSVC method of Parrado-Hernández et al. (2003). This method, originally given for SVM hinge loss, uses the following heuristic criterion to select the next basis function $j^* \notin J$:

$$j^* = \arg \min_{j \in I, j \notin J} \max_{l \in J} |K_{jl}| \tag{8}$$

with the aim of encouraging new basis functions that are far from the basis functions that are already chosen; also, j is restricted only to the support vector indices (I in (5)). For a clean comparison with our methods, we implemented GSVC for SVMs using quadratic penalization, $\max(0, 1-y_i o_i)^2$. We also tried another criterion, suggested to us by Alex Smola, that is more complex than (8):

$$j^* = \arg \max_{j \in I, j \notin J} (1 - y_j o_j)^2 d_j^2 \tag{9}$$

where d_j is the distance (in feature space) of the j-th training point from the subspace spanned by the elements of J. This criterion is based on an upper bound on the improvement to the training cost function obtained by including the j-th basis function. It also makes sense intuitively as it selects basis functions that are both not well approximated by the others (large d_j) and for which the error incurred is large.[8] Below, we will refer to this criterion as *BH*. It is worth noting that both (8) and (9) can be computed very efficiently.

Figures 1 and 2 compare the six methods on six data sets.[9] Overall, *SpSVM-1* and *SpSVM-2* give the best performance in terms of achieving good reduction of test error rate with respect to the number of basis functions. Although *SpSVM-2* slightly lags *SpSVM-1* in terms of performance in the early stages, it does equally well as more basis functions are added. Since *SpSVM-2* is significantly less expensive, it is the best method to use. Since *SpSVM-1* is quite cheap in the early stages, it is also appropriate to think of a hybrid method in which *SpSVM-1* is used in the early stages and, when it becomes expensive, switch to *SpSVM-2*. The other methods sometimes do well, but, overall, they are inferior in comparison to *SpSVM-1* and *SpSVM-2*. Interestingly, on the *IJCNN* and *Vehicle* data

7. We also tried the method of Bach and Jordan (2005) which uses the training labels, but we noticed little improvement.
8. Note that when the set of basis functions is not restricted, the optimal β satisfies $\lambda\beta_i y_i = \max(0, 1 - y_i o_i)$.
9. Most figures given in this paper appear in pairs of two plots. One plot gives test error rate as a function of the number of basis functions, to see how effective the compression is. The other plot gives the test error rate as a function of CPU time, and is used to indicate the efficiency of the method.

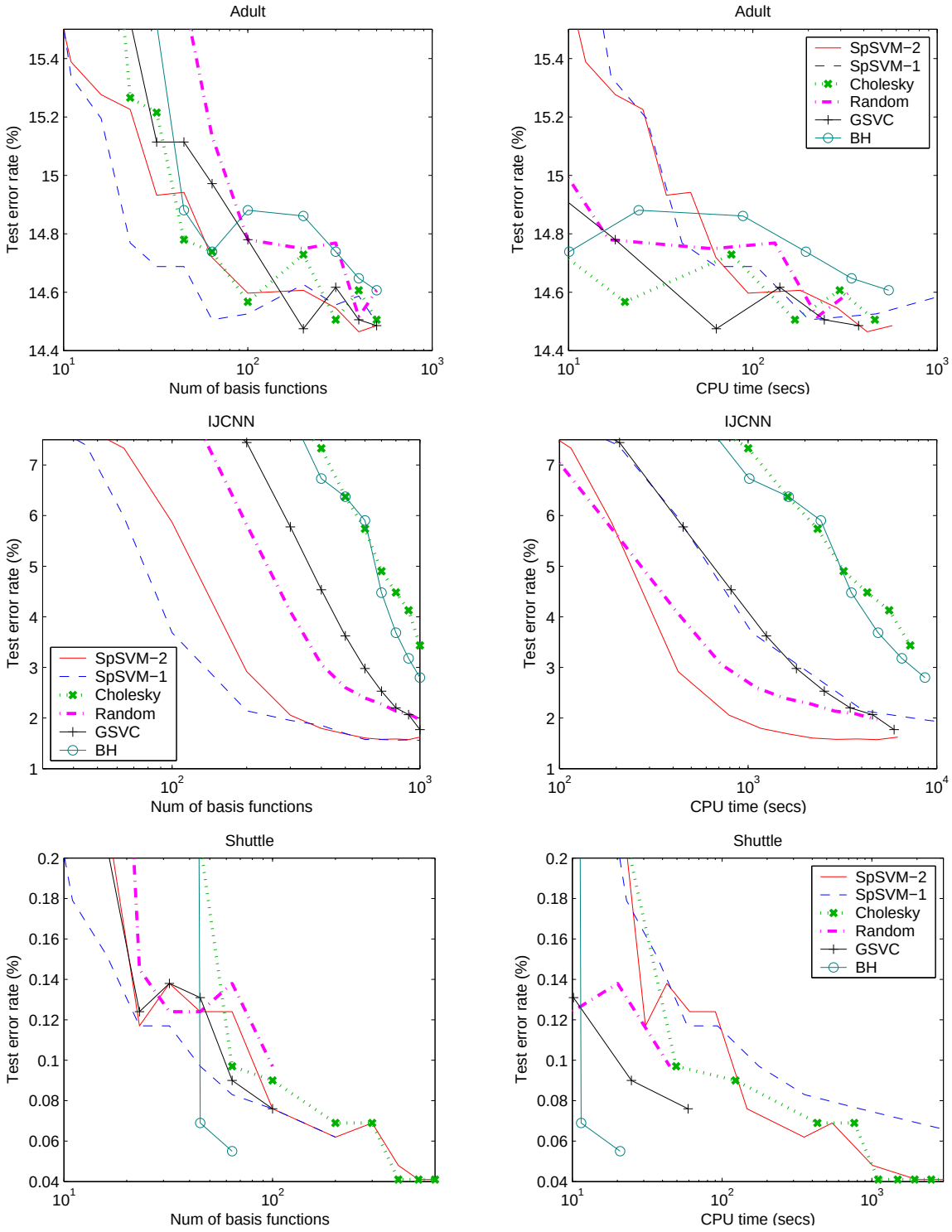


Figure 1: Comparison of basis selection methods on *Adult*, *IJCNN* & *Shuttle*. On *Shuttle* some methods were terminated because of ill-conditioning in the matrix P in (6).

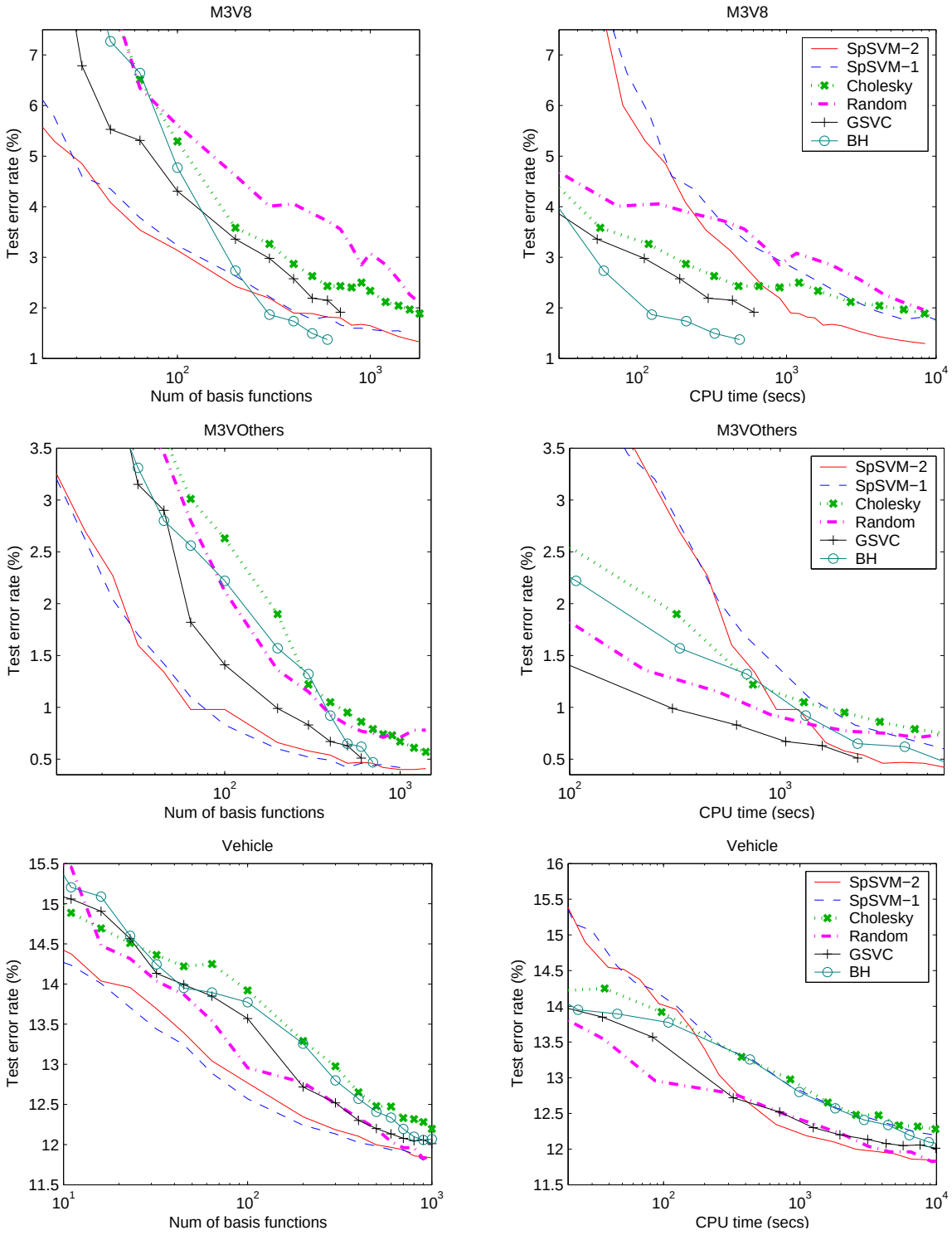


Figure 2: Comparison of basis selection methods on *M3V8*, *M3VOthers* & *Vehicle*.

sets, *Cholesky*, *GSVC* and *BH* are even inferior to *Random*. A possible explanation is as follows: these methods give preference to points that are furthest away in feature space from the points already selected. Thus, they are likely to select points which are outliers (far from the rest of the training points); but outliers are probably unsuitable points for expanding the decision function.

As we mentioned in Section 1, there also exist other greedy methods of kernel basis selection that are motivated by ideas from boosting. These methods are usually given in a setting different from that we consider: a set of (kernel) basis functions is given and a regularizer (such as $\|\beta\|_1$) is directly specified on the multiplier vector β. The method of Bennett et al. (2002) called MARK is given for least squares problems. It is close to the kernel matching pursuit method. We compare *SpSVM-2* with kernel matching pursuit and discuss MARK in Section 5. The method of Bi et al. (2004) uses column generation ideas from linear and quadratic programming to select new basis functions and so it requires the solution of, both, the primal and dual problems.[10] Thus, the basis selection process is based on the sensitivity of the primal objective function to an incoming basis function. On the other hand, our *SpSVM* methods are based on computing an estimate of the decrease in the primal objective function due to an incoming basis function; also, the dual solution is not needed.

4. Hyperparameter Tuning

In the actual design process, the values of hyperparameters need to be determined. This can be done using k-fold cross validation. Cross validation (CV) can also be used to choose d, the number of basis functions. Since the solution given by our method approaches the SVM solution as d becomes large, there is really no need to choose d at all. One can simply choose d to be as big a value as possible. But, to achieve good reduction in the classifier complexity (as well as computing time) it is a good idea to track the validation performance as a function of d and stop when this function becomes nearly flat. We proceed as follows. First an appropriate value for $d_{\max}$ is chosen. For a given choice of hyperparameters, the basis selection method (say, *SpSVM-2*) is then applied on each training set formed from the k-fold partitions till $d_{\max}$ basis functions are chosen. This gives an estimate of the k-fold CV error for each value of d from 1 to $d_{\max}$. We choose d to be the number of basis functions that gives the lowest k-fold CV error. This computation can be repeated for each set of hyperparameter values and the best choice can be decided.

Recall that, at stage d, our basis selection methods choose the $(d+1)$-th basis function from a set of κ random basis functions. To avoid the effects of this randomness on hyperparameter tuning, it is better to make this κ-set to be dependent only on d. Thus, at stage d, the basis selection methods will choose the same set of κ random basis functions for all hyperparameter values.

We applied the above ideas on 11 benchmark data sets from (Rätsch) using *SpSVM-2* as the basis selection method. The Gaussian kernel, $k(x_i,x_j) = 1+\exp(-\gamma\|x_i - x_j\|^2)$ was used. The hyperparameters, λ and γ were tuned using 3-fold cross validation. The values, 2^i, $i = -7,\cdots,7$ were used for each of these parameters. Ten different train-test partitions were tried to get an idea of the variability in generalization performance. We used $\kappa = 25$ and $d_{\max} = 25$. (The *Titanic* data set has three input variables, which are all binary; hence we set $d_{\max} = 8$ for this data set.)

Table 1 (already introduced in Section 1) gives the results. For comparison we also give the results for the SVM (solution of (1)); in the case of SVM, the number of support vectors (n_{SV}) is the

10. The CPLEX LP/QP solver is used to obtain these solutions.

number of basis functions. Clearly, our method achieves an impressive reduction in the number of basis functions, while yielding test error rates comparable to the full SVM.

5. Comparison with Kernel Matching Pursuit

Kernel matching pursuit (KMP) (Vincent and Bengio, 2002) was mainly given as a method of greedily selecting basis functions for the non-regularized kernel least squares problem. As we already explained in Section 3, our basis selection methods can be viewed as extensions of the basic ideas of KMP to the SVM case. In this section we empirically compare the performances of these two methods. For both methods we only consider *Basis Selection Method 2* and refer to the two methods simply as *KMP* and *SpSVM*. It is also interesting to study the effect of the regularizer term ($\lambda\|w\|^2/2$ in (1)) on generalization. The regularizer can be removed by setting $\lambda = 0$. The original KMP formulation of Vincent and Bengio (2002) considered such a non-regularized formulation only. In the case of SVM, when perfect separability of the training data is possible, it is improper to set $\lambda = 0$ without actually rewriting the primal formulation in a different form; so, in our implementation we brought in the effect of no-regularization by setting λ to the small value, 10^{-5}. Thus, we compare 4 methods: *KMP-R*, *KMP-NR*, *SpSVM-R* and *SpSVM-NR*. Here "*R*" and "*NR*" refer to regularization and no-regularization, respectively.

Figures 3 and 4 compare the four methods on six data sets. Except on *M3V8*, SpSVM gives a better performance than KMP. The better performance of *KMP* on *M3V8* is probably due to the fact that the examples corresponding to each of the digits, 3 and 8, are distributed as a Gaussian, which is suited to the least squares loss function. Note that in the case of *M3VOthers* where the "Others" class (corresponding to all digits other than 3) is far from a Gaussian distribution, SVM does better than KMP.

The no-regularization methods, *KMP-NR* and *SpSVM-NR* give an interesting performance. In the initial stages of basis addition we are in the underfitting zone and so they perform as well (in fact, a shade better) than their respective regularized counterparts. But, as expected, they start overfitting when many basis functions are added. See, for example the performance on *Adult* data set given in Figure 3. Thus, when using these non-regularized methods, a lot of care is needed in choosing the right number of basis functions. The number of basis functions at which overfitting sets-in is smaller for *SpSVM-NR* than that of *KMP-NR*. This is because of the fact that, while KMP has to concentrate on reducing the residual on all examples in its optimization, SVM only needs to concentrate on the examples violating the margin condition.

It is also useful to mention the method, MARK[11] of Bennett et al. (2002) which is closely related to KMP. In this method, a new basis function (say, the one corresponding to the j-th training example) is evaluated by looking at the magnitude (larger the better) of the gradient of the primal objective function with respect to β_j evaluated at the current β_J. This gradient is the dot product of the kernel column containing K_{ij} and the residual vector having the elements, $o_i - y_i$. The computational cost as well as the performance of MARK are close to those of KMP. MARK can also be easily extended to the SVM problem in (1): all that we need to do is to replace the residual vector mentioned above by the vector having the elements, $y_i \max\{0, 1 - y_i o_i\}$. This modified method (which uses our Newton optimization method as the base solver) is close to our *SpSVM-2* in terms of computational cost as well as performance. Note that, if we optimize (7) for β_j using only a

11. The basis selection idea used in MARK is also given in the earlier papers, Mallat and Zhang (1993) and Adler et al. (1996) under the name, *Basic Matching Pursuit*.

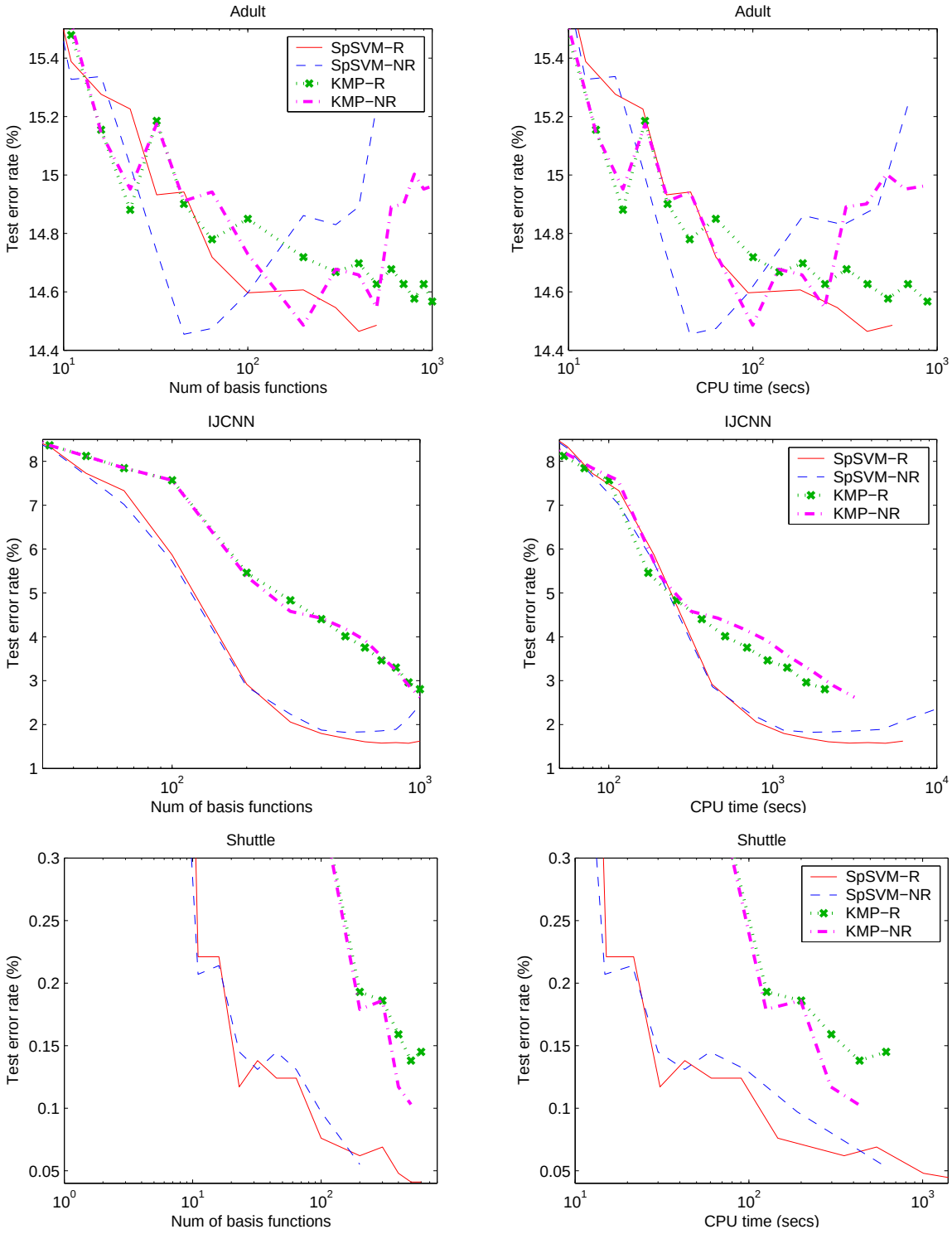


Figure 3: KMP vs SpSVM (with/without regularization) on *Adult*, *IJCNN* & *Shuttle*.

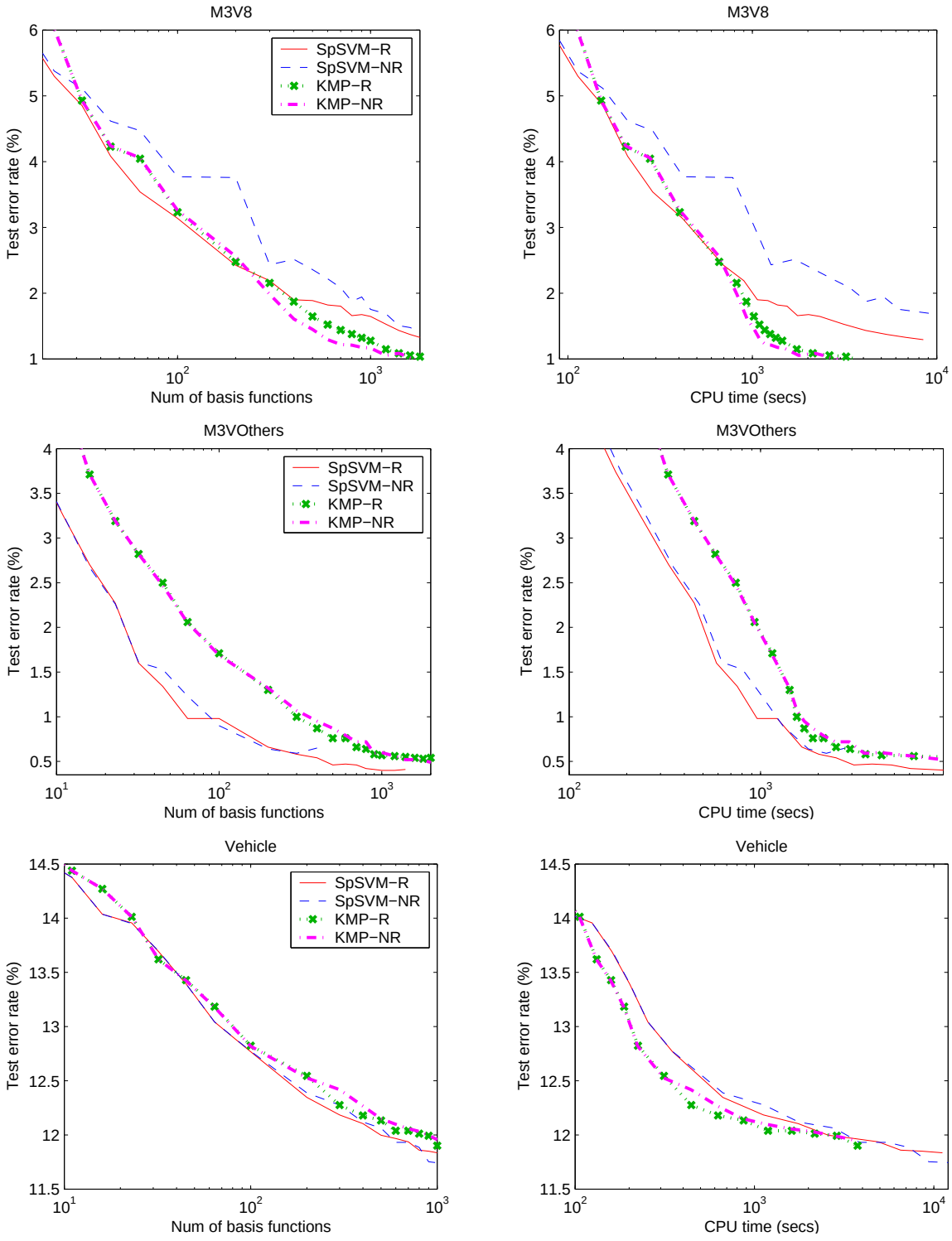


Figure 4: KMP vs SpSVM (with/without regularization) on *M3V8*, *M3VOthers* & *Vehicle*.

single Newton step, the difference between MARK (as adapted above to SVMs) and *SpSVM-2* is only in the use of the second order information.

6. Additional Tuning

We discuss in this section the choice of κ for *SpSVM* as well as the possibility of not solving (4) every time a new basis function is added.

6.1 Few Retrainings

It might be a bit costly to perform the Newton optimization described in Section 2.1 each time a new basis function is added. Indeed, it is unlikely that the set of support vectors changes a lot after each addition. Therefore, we investigate the possibility of retraining only from time to time.

We first tried to do retraining only when $|J| = 2^p$ for some $p \in \mathbb{N}$, the set of positive integers. It makes sense to use an exponential scale since we expect the solution not to change too much when J is already relatively large. Note that the overall complexity of the algorithm does not change since the cost of adding one basis function is still $O(nd)$. It is only the constant which is better, because fewer Newton optimizations are performed.

The results are presented in figure 5. For a given number of basis functions, the test error is usually not as good as if we always retrain. But on the other hand, this can be much faster. We found that a good trade-off is to retrain whenever $|J| = \lfloor 2^{p/4} \rfloor$ for $p \in \mathbb{N}$. This is the strategy we will use for the experiments in the rest of the paper.

6.2 Influence of κ

The parameter κ is the number of candidate basis functions that are being tried each time a new basis function should be added: we select a random set of κ examples and the best one (as explained in Section 3.2) among them is chosen. If $\kappa = 1$, this amounts to choosing a random example at each step (i.e. the *Random* method on figures 1 and 2) .

The influence of κ is studied in figure 6. The larger κ is, the better the test error for a given number of basis functions, but also the longer the training time. We found that $\kappa = 10$ is a good trade-off and that is the value that we will keep for the experiments presented in the next section.

Finally, an interesting question is how to choose appropriately a good value for κ and an efficient retraining strategy. Both are likely to be problem dependent, and even though $\kappa = 59$ was suggested by Smola and Schölkopf (2000), we believe that there is no universal answer. The answer would for instance depend on the cost associated with the computation of the kernel function, on the number of support vectors and on the number of training points. Indeed, the basic cost for one iteration is $O(nd)$ and the number of kernel calculations is $\kappa n_{\mathrm{SV}} + n$: the first term corresponds to trying different basis function, while the second one correspond to the inclusion of the chosen basis function. So κ should be chosen such that the kernel computations takes about the same tame as the training itself.

Ideally, an adaptive strategy should be designed to find automatically κ and to adjust the retraining schedule. The decay rate of the objective function as well as the variance of the scores produced by the basis selection scoring function would be two key quantities helpful to adjust them.

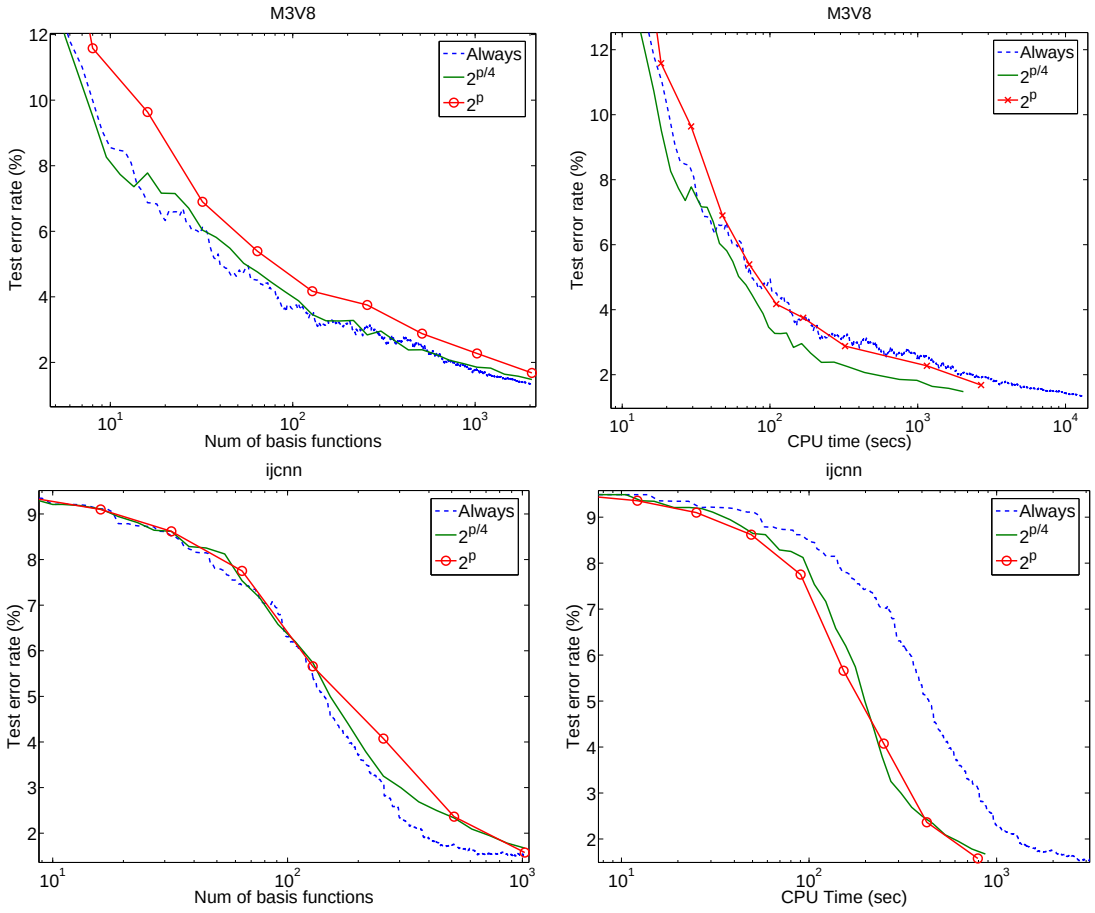


Figure 5: Three different possible retraining strategies showing a different trade-off between accuracy and time: always retraining is too time consumming; on the other hand retraining not often enough can lead to sub-optimal performances (see the top left plot). For these experiments, $\kappa = 100$ was used.

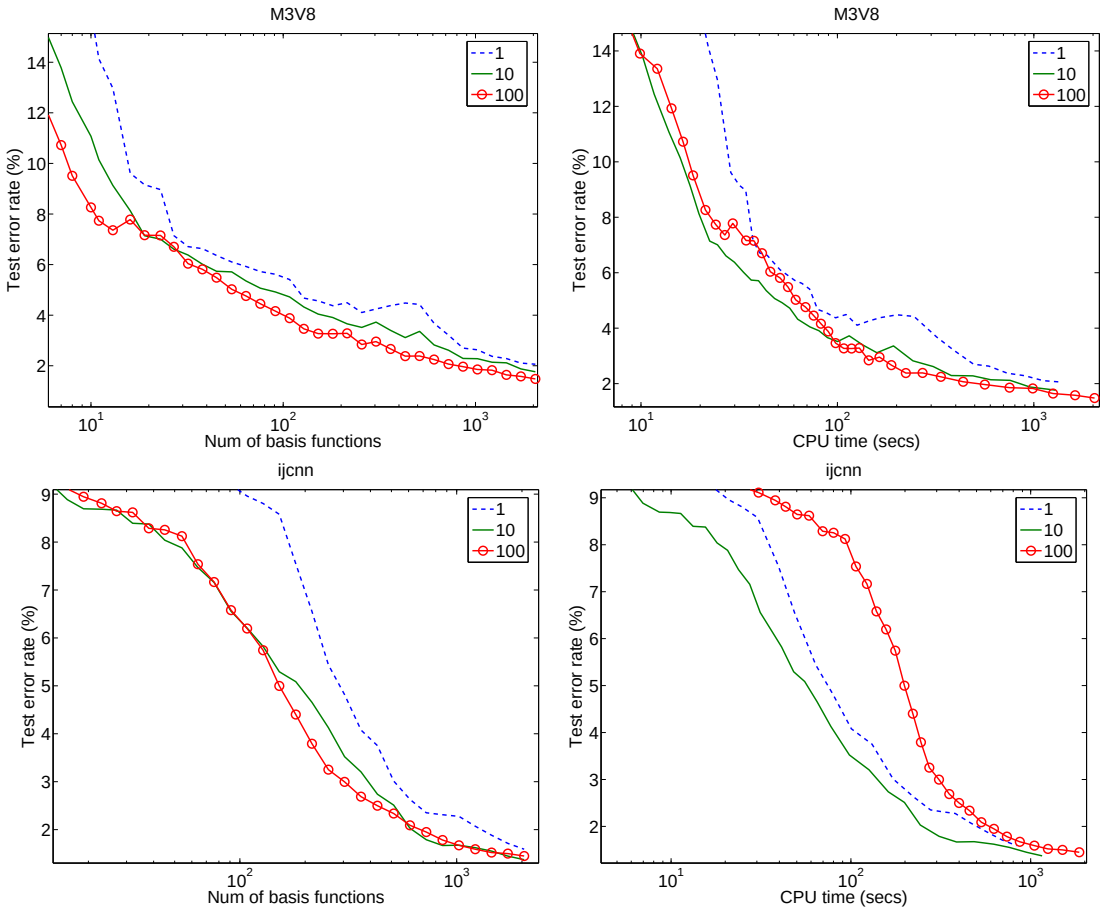


Figure 6: Influence of the paramter κ: when it is large, a good reduction is achieved (left column), but the computaional cost is larger (right column). $\kappa = 10$ seems a good trade-off.

7. Comparison with Standard SVM Training

We conclude the experimental study by comparing our method with the well known SVM solver, *SVMLight* (Joachims, 1999).[12] For this solver, we selected random training subsets of sizes from $2^{-10}n, 2^{-9}n, \ldots, n/4, n/2, n$. For each training set size, we measure the test error, the training time and the number of support vectors. The L_2 version (quadratic penalization of the slacks) is the one relevant for our study since it is the same loss function as the one we used; note that, when the number of basis functions increases towards n, the *SpSVM* solution will converge to the L_2 SVM solution. For completeness, we also included experimental results of an SVM trained with a L_1 penalization of the slacks. Finally, note that for simplicity we kept the same hyperparameters for the different sizes, but that both methods would certainly gain by additional hyerparameter tuning (for instance when the number of basis functions is smaller, the bandwith of the RBF kernel should be larger).

Results are presented in figures 7 and 8. In terms of compression (left columns), our method is usually able to reach the same accuracy as a standard SVM using less than one-tenth the number of basis functions (this confirms the results of table 1).

From a time complexity point of view also, our method is very competitive and can reach the same accuracy as an SVM in less time. The only disappointing performance is on the *M3V8* data set. A possible explanation is that for this data set, the number of support vectors is very small and a standard SVM can compute the exact solution quickly.

Finally, note that when the number of basis functions is extremely small compared to the number of training examples, *SpSVM* can be slower than a SVM trained on a small subset (left part of the right column plots). It is because solving (4) using n training examples while there are only few parameters to estimate is an overkill. It would be wiser to choose n as a function of d, the number of basis functions.

8. Conclusion

In this paper we have given a fast primal algorithm that greedily chooses a subset of the training basis functions to approximate the SVM solution. As the subset grows the solution converges to the SVM solution since choosing the subset to be the entire training set is guaranteed to yield the exact SVM solution. The real power of the method lies in its ability to form very good approximations of the SVM classifier with a clear control on the complexity of the classifier (number of basis functions) as well as the training time. In most data sets, performance very close to that of the SVM is achieved using a set of basis functions whose size is a small fraction of the number of SVM support vectors. The graded control over the training time offered by our method can be valuable in large scale data mining. Many a times, simpler algorithms such as decision trees are preferred over SVMs when there is a severe constraint on computational time. While there is no satisfactory way of doing early stopping with SVMs, our method enables the user to control the training time by choosing the number of basis functions to use.

Our method can be improved and modified in various ways. Hyperparameter tuning time can be substantially reduced by using gradient-based methods on a differentiable estimate of the generalization performance formed using k-fold cross validation and posterior probabilities. Improved methods of choosing the κ-subset of basis functions in each step can also make the method more ef-

12. The default optimization options of *SVMLight (Version 6.0)* have been used.

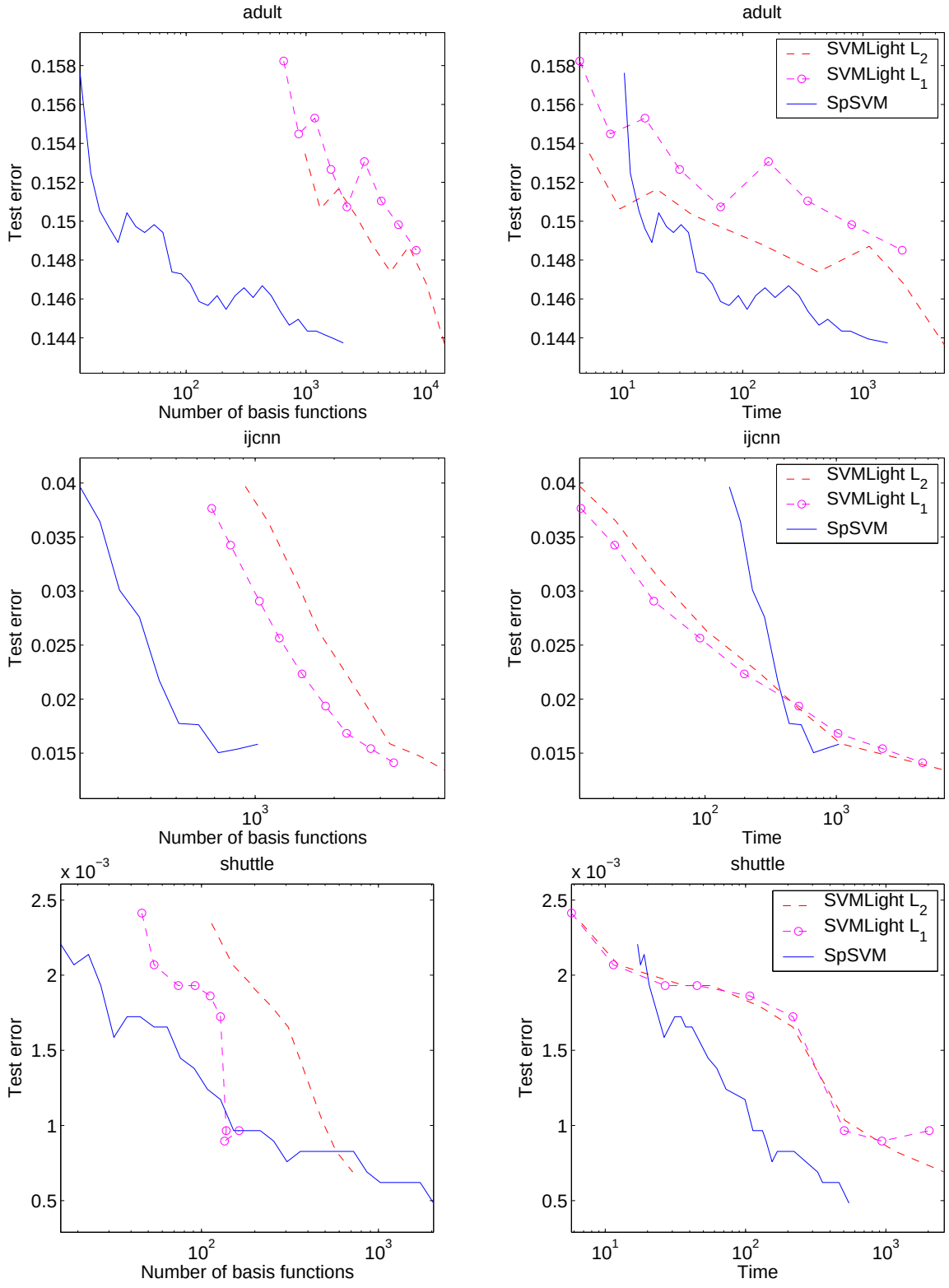


Figure 7: Comparison of *SpSVM* with SVMLight on *Adult*, *IJCNN*, *Shuttle*. For SVMLight, "Num of basis functions" should be understood as number of support vectors.

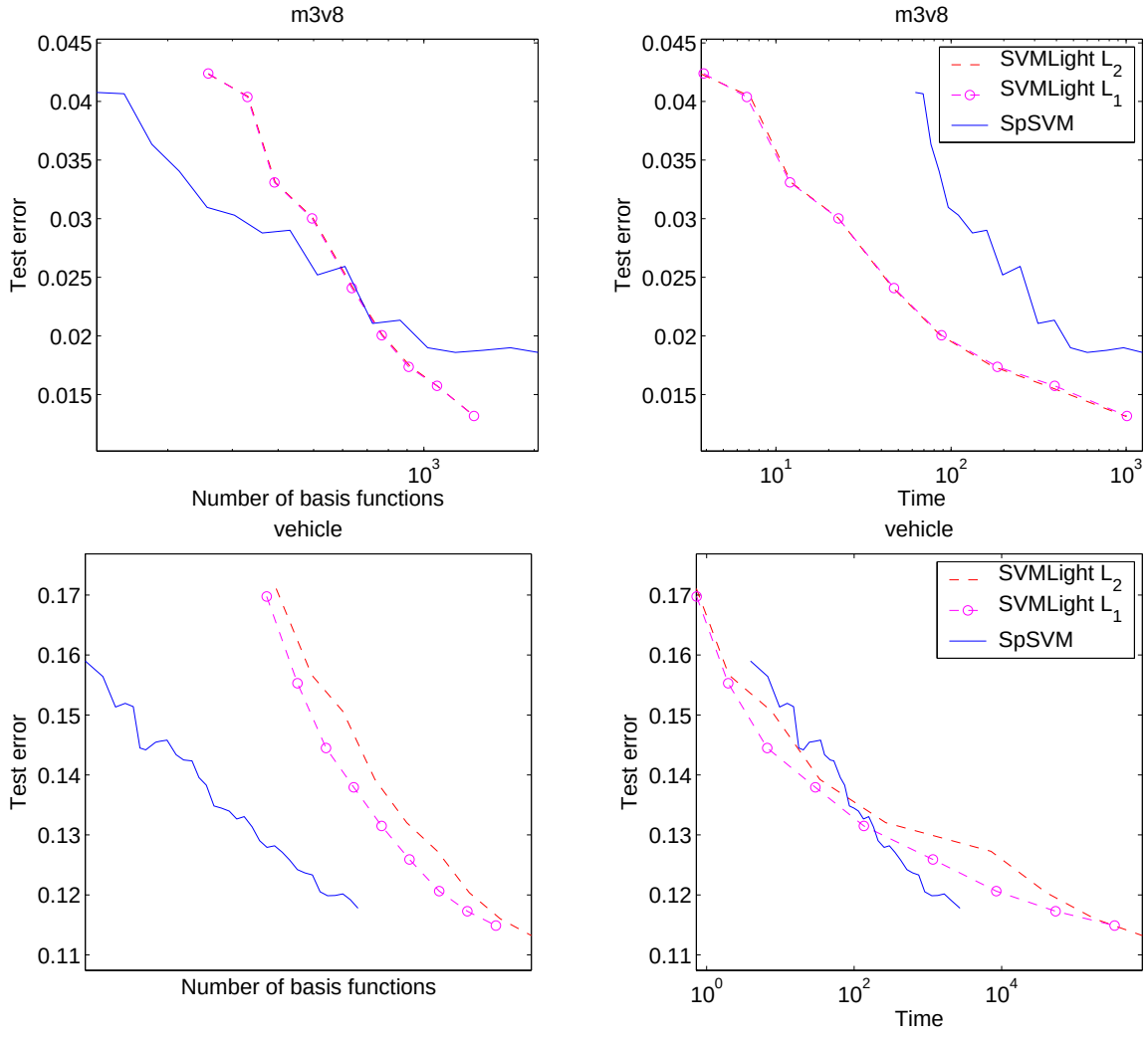


Figure 8: Comparison of *SpSVM* with SVMLight on *M3V8* and *Vehicle*. For SVMLight, "Num of basis functions" should be understood as number of support vectors.

fective. Also, all the ideas described in this paper can be easily extended to the Huber loss function using the ideas in Keerthi and DeCoste (2005).

Appendix: A Description of Data Sets Used

As in the main paper, let n denote the number of training examples. The six data sets used for the main experiments of the paper are: *Adult*, *IJCNN*, *M3V8*, *M3VOthers*, *Shuttle* and *Vehicle*. For *M3V8* and *M3VOthers* we go by the experience in (DeCoste and Schölkopf, 2002) and use the polynomial kernel, $k(x_i,x_j) = 1+(1+x_i \cdot x_j)^9$ where each x_i is normalized to have unit length. For all other data sets, we use the Gaussian kernel, $k(x_i,x_j) = 1+\exp(-\gamma\|x_i-x_j\|^2)$. The values of γ are given below.[13] In each case, the values chosen for γ and λ are ballpark values such that the methods considered in the paper give good generalization performance.

Adult data set is the version given by Platt in his *SMO* web page: `http://www.research.microsoft.com/~jplatt/smo.html`. Platt created a sequence of data sets with increasing number of examples in order to study the scaling properties of his *SMO* algorithm with respect to n. For our experiments we only used *Adult-8* which has 22,696 training examples and 9865 test examples. Each example has 123 binary features, of which typically only 14 are non-zero. We used $\gamma = 0.05$ and $\lambda = 1$.

The next five data sets are available from the following LIBSVM-Tools page: `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

IJCNN data set has 49,990 training examples and 91,701 test examples. Each example is described by 22 features. We used $\gamma = 4$ and $\lambda = 1/16$.

Shuttle data set has 43,500 training examples and 14,500 test examples. Each example is described by 9 features. This is a multiclass data set with seven classes. We looked only at the binary classification problem of differentiating class 1 from the rest. We used $\gamma = 16$ and $\lambda = 1/512$.

M3V8 data set is the binary classification problem of *MNIST* corresponding to classifying digit 3 from digit 8. The original data set has 11,982 training examples and 1984 test examples for this problem. Since the original test data set could not clearly show a distinction between several closely competing methods, we formed an extended test set by applying invariances like translation and rotation to create an extended test set comprising of 17,856 examples. (This data set can be obtained from the authors.) We used $\lambda = 0.1$.

M3VOthers data set is another binary classification problem of *MNIST* corresponding to differentiating digit 3 from all the other digits. The data set has 60,000 training examples and 10,000 test examples. We used $\lambda = 0.1$.

Vehicle data set corresponds to the "vehicle (combined, scaled to [-1,1])" version in the LIBSVM-Tools page mentioned above. It has 78,823 training examples and 19,705 test examples. Each example is described by 100 features. This is a multiclass data set with three classes. We looked only at the binary classification problem of differentiating class 3 from the rest. We used $\gamma = 1/8$ and $\lambda = 1/32$.

Apart from the above six large data sets, we also used modified versions of UCI data sets as given in (Rätsch). These data sets were used to show the sparsity that is achievable using our method; see Table 1 of Section 1 and the detailed discussion in Section 4.

13. For both, the polynomial and Gaussian kernels, the additive term "1" gives the effect of including the threshold term in the classifier and regularizing it.

References

J. Adler, B. D. Rao, and K. Kreutz-Delgado. Comparison of basis selection methods. In *Proceedings of the 30th Asilomar conference on signals, systems and computers*, pages 252–257, 1996.

F. Bach and M. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the Twenty-second International Conference on Machine Learning (ICML)*, 2005.

K. P. Bennett, M. Momma, and M. J. Embrechts. MARK: A boosting algorithm for heterogeneous kernel models. In *Proceedings of SIGKDD'02*, 2002.

J. Bi, T. Zhang, and K. P. Bennet. Column generation boosting methods for mixture of kernels. In *Proceedings of SIGKDD'04*, 2004.

C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In *Proceedings of the* 9^{th} *NIPS Conference*, pages 375–381, 1997.

O. Chapelle. Training a support vector machine in the primal. *Journal of Machine Learning Research*, 2005. submitted.

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

T. Downs, K. E. Gates, and A. Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1180, 2001.

T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, Massachussetts, 1999.

S. S. Keerthi and W. Chu. A matching pursuit approach to sparse Gaussian process regression. In *Proceedings of the* 18^{th} *NIPS Conference*, 2006.

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

N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The Informative Vector Machine. In *Proceedings of the* 15^{th} *NIPS Conference*, pages 609–616, 2003.

Y. J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, Philadelphia, 2001.

K. M. Lin and C. J. Lin. A study on reduced support vector machines. *IEEE TNN*, 14:1449–1459, 2003.

S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on ASSP*, 41:3397–3415, 1993.

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

J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, 31:148–162, 1977.

E. Osuna and F. Girosi. Reducing the run-time complexity of support vector machines. In *Proceedings of the International Conference on Pattern Recognition*, 1998.

E. Parrado-Hernández, I. Mora-Jimenéz, J. Arenas-García, A. R. Figueiras-Vidal, and A. Navia-Vázquez. Growing support vector classifiers with controlled complexity. *Pattern Recognition*, 36:1479–1488, 2003.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research, Redmond, 1998.

G. Rätsch. *Robust boosting via convex optimization*. PhD thesis, University of Potsdam, Department of Computer Science, Potsdam, Germany, 2001.

G. Rätsch. Benchmark repository. `http://ida.first.fraunhofer.de/~raetsch/`.

B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Muller, G. Raetsch, and A. J. Smola. Input space vs. feature space in kernel-based methods. *IEEE TNN*, 10:1000–1017, 1999.

M. Seeger. Low rank updates for the Cholesky decomposition. Technical report, University of California, Berkeley, 2004.

M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Proceedings of the Workshop on AI and Statistics*, 2003.

A. J. Smola and P. L. Bartlett. Sparse greedy Gaussian process regression. In *Proceedings of the* 13^{th} *NIPS Conference*, pages 619–625, 2001.

A. J. Smola and Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the* 17^{th} *International Conference on Machine Learning*, pages 911–918, 2000.

I. Steinwart. Sparseness of support vector machines - some asymptotically sharp bounds. In *Proceedings of the* 16^{th} *NIPS Conference*, pages 169–184, 2004.

T. Thies and F. Weber. Optimal reduced-set vectors for support vector machines with a quadratic kernel. *Neural Computation*, 16:1769–1777, 2004.

M. E. Tipping. Sparse Bayesian learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

P. Vincent and Y. Bengio. Kernel matching pursuit. *Machine Learning*, 48:165–187, 2002.

M. Wu, B. Schölkopf, and G. Bakir. Building sparse large margin classifiers. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

Journal of Machine Learning Research 7 (2006) 1517–1530 Submitted 8/05; Published 6/06

Exact 1-Norm Support Vector Machines via Unconstrained Convex Differentiable Minimization

Olvi L. Mangasarian* OLVI@CS.WISC.EDU
Computer Sciences Department
University of Wisconsin
Madison, WI 53706, USA

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

Support vector machines utilizing the 1-norm, typically set up as linear programs (Mangasarian, 2000; Bradley and Mangasarian, 1998), are formulated here as a completely unconstrained minimization of a convex differentiable piecewise-quadratic objective function in the dual space. The objective function, which has a Lipschitz continuous gradient and contains only one additional finite parameter, can be minimized by a generalized Newton method and leads to an exact solution of the support vector machine problem. The approach here is based on a formulation of a very general linear program as an unconstrained minimization problem and its application to support vector machine classification problems. The present approach which generalizes both (Mangasarian, 2004) and (Fung and Mangasarian, 2004) is also applied to nonlinear approximation where a minimal number of nonlinear kernel functions are utilized to approximate a function from a given number of function values.

1. Introduction

One of the principal advantages of 1-norm support vector machines (SVMs) is that, unlike 2-norm SVMs, they are very effective in reducing input space features for linear kernels and in reducing the number of kernel functions (Bradley and Mangasarian, 1998; Fung and Mangasarian, 2004) for nonlinear SVMs. With few exceptions, the simplex method (Dantzig, 1963) has been the exclusive algorithm for solving 1-norm SVMs. The interesting paper (Zhu et al., 2004) which treats the 1-norm SVM uses standard linear programming packages for solving their formulation. To the best of our knowledge there has not been an exact completely unconstrained differentiable minimization formulation of 1-norm SVMs, which is the principal concern of the present rather theoretical contribution which we outline now.

In Section 2 we show how a very general linear program can be solved as the minimization of a completely unconstrained differentiable piecewise-quadratic convex function that contains a single finite parameter. This result generalizes (Mangasarian, 2004) where linear programs with millions of constraints were solved as unconstrained minimization problems by a generalized Newton method. In Section 3 we show how to set up 1-norm SVMs, with linear and nonlinear kernels as unconstrained minimization problems and state a generalized Newton method for their solution. In Section 4 we show how to solve the problem of approximating an unknown function based on a given number of function values using a minimal number of kernel functions. We achieve this

∗. For commercial use of the algorithms described in this work, please contact the author.

by again converting a 1-norm approximation problem to an unconstrained minimization problem. Computational results given in Section 5 show that the proposed approach is faster than a conventional linear programming solver, CPLEX (ILO, 2003), and faster than another related method as well as having better input space feature suppression for a linear classifier and mostly better kernel function suppression for a nonlinear classifier. Section 6 concludes the paper.

We now describe our notation and give some background material. All vectors will be column vectors unless transposed to a row vector by a prime $'$. For a vector x in the n-dimensional real space R^n, x_+ denotes the vector in R^n with all of its negative components set to zero. This corresponds to projecting x onto the nonnegative orthant. For a vector $x \in R^n$, x_* denotes the vector in R^n with components $(x_*)_i = 1$ if $x_i > 0$ and 0 otherwise (i.e. x_* is the result of applying the step function component-wise to x). For $x \in R^n$, $\|x\|_1$, $\|x\|$ and $\|x\|_\infty$, will denote the $1-$, $2-$ and $\infty-$ norms of x. For simplicity we drop the 2 from $\|x\|_2$. The notation $A \in R^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix A' will denote the transpose of A, A_i will denote the i-th row of A and A_{ij} will denote the ij-th element of A. A vector of ones or zeroes in a real space of arbitrary dimension will be denoted by e or 0, respectively. For a piecewise-quadratic function such as, $f(x) = \frac{1}{2}||(Ax-b)_+||^2 + \frac{1}{2}x'Px$, where $A \in R^{m\times n}$, $P \in R^{n\times n}$, $P = P'$, P positive semidefinite and $b \in R^m$, the ordinary Hessian does not exist because its gradient, the $n \times 1$ vector $\nabla f(x) = A'(Ax-b)_+ + Px$, is not differentiable but is Lipschitz continuous with a Lipschitz constant of $\|A'\|\,\|A\| + \|P\|$. However, one can define its **generalized Hessian** (Hiriart-Urruty et al., 1984; Facchinei, 1995; Mangasarian, 2001) which is the $n \times n$ symmetric positive semidefinite matrix:

$$\partial^2 f(x) = A'diag(Ax-b)_* A + P,$$

where $diag(Ax-b)_*$ denotes an $m \times m$ diagonal matrix with diagonal elements $(A_i x - b_i)_*$, $i = 1,\ldots,m$. The generalized Hessian has many of the properties of the regular Hessian (Hiriart-Urruty et al., 1984; Facchinei, 1995; Mangasarian, 2001) in relation to $f(x)$. If the smallest eigenvalue of $\partial^2 f(x)$ is greater than some positive constant for all $x \in R^n$, then $f(x)$ is a strongly convex piecewise-quadratic function on R^n. A separating plane, with respect to two given point sets $\mathcal{A}$ and $\mathcal{B}$ in R^n, is a plane that attempts to separate R^n into two halfspaces such that each open halfspace contains points mostly of $\mathcal{A}$ or $\mathcal{B}$. The notation $:=$ denotes a definition.

2. Linear Programs as Exact Unconstrained Differentiable Minimization Problems

We consider in this section a very general linear program (LP) that contains nonnegative and unrestricted variables as well as inequality and equality constraints. We will show how to obtain an exact solution of this LP by a single minimization of a completely unconstrained differentiable piecewise-quadratic function that contains a single finite parameter. We begin with the primal linear program:

$$\min_{(x,y)\in R^{n+\ell}} c'x + d'y \;\; s.t. \;\; Ax + By \geq b,\; Ex + Gy = h,\; x \geq 0, \tag{1}$$

where $c \in R^n$, $d \in R^\ell$, $A \in R^{m\times n}$, $B \in R^{m\times \ell}$, $E \in R^{k\times n}$, $G \in R^{k\times \ell}$, $b \in R^m$ and $h \in R^k$, and its dual:

$$\max_{(u,v)\in R^{m+k}} b'u + h'v \;\; s.t. \;\; A'u + E'v \leq c,\; B'u + G'v = d,\; u \geq 0. \tag{2}$$

The exterior penalty problem for the dual linear program is:

$$\min_{(u,v)\in R^{m+k}} \varepsilon(-b'u - h'v) + \frac{1}{2}(\|(A'u + E'v - c)_+\|^2 + \|B'u + G'v - d\|^2 + \|(-u)_+\|^2). \tag{3}$$

Solving the exterior penalty problem for a positive sequence $\{\varepsilon_i\}$ converging to zero will yield a solution to the dual linear program (2) (Fiacco and McCormick, 1968; Bertsekas, 1999). However, we will *not* do that here because of the inherent inaccuracies associated with asymptotic exterior penalty methods and the fact that this would merely yield an approximate *dual* solution but *not* a primal solution. Instead, we will solve the exterior penalty problem for some finite value of the penalty parameter ε and from this *inexact* dual solution we shall easily extract an *exact* primal solution by using the following proposition.

Proposition 1 Exact Primal Solution Computation *Let the primal LP (1) be solvable. Then the dual exterior penalty problem (3) is solvable for all* $\varepsilon > 0$. *For any* $\varepsilon \in (0, \bar{\varepsilon}]$ *for some* $\bar{\varepsilon} > 0$, *any solution* (u, v) *of (3) generates an exact solution to primal LP (1) as follows:*

$$x = \frac{1}{\varepsilon}(A'u + E'v - c)_+, \;\; y = \frac{1}{\varepsilon}(B'u + G'v - d). \tag{4}$$

In addition, this (x, y) *minimizes:*

$$\|x\|^2 + \|y\|^2 + \|Ax + By - b\|^2, \tag{5}$$

over the solution set of the primal LP (1).

Proof The dual exterior penalty minimization problem (3) can be written in the equivalent form:

$$\begin{array}{rcl} \min\limits_{(u,v,z_1,z_2)\in R^{m+k+n+m}} \varepsilon(-b'u - h'v) + \frac{1}{2}(\|z_1\|^2 + \|B'u + G'v - d\|^2 & + & \|z_2\|^2) \\ s.t. \; -A'u - E'v + c + z_1 & \geq & 0 \\ u + z_2 & \geq & 0. \end{array} \tag{6}$$

The justification for this is that at a minimum of (6) the variables z_1 and z_2 are nonnegative, else if any component of these variables is negative the objective function can be strictly decreased by setting that component to zero while maintaining constraint feasibility. Hence, at a solution of (6), $z_1 = (A'u + E'v - c)_+$ and $z_2 = (-u)_+$. The Wolfe dual (Mangasarian, 1994, Problem 8.2.2) for the convex quadratic program (6) is:

$$\begin{array}{rcl} \max\limits_{(u,v,z_1,z_2,r,s)\in R^{m+k+n+m+n+m}} -\frac{1}{2}((\|z_1\|^2 + \|B'u\|^2 + \|G'v\|^2 + 2v'GB'u - \|d\|^2 + \|z_2\|^2) & - & c'r \\ s.t. \; -\varepsilon b + B(B'u + G'v - d) + Ar - s & = & 0 \\ -\varepsilon h + G(B'u + G'v - d) + Er & = & 0 \\ z_1 = r & \geq & 0 \\ z_2 = s & \geq & 0, \end{array} \tag{7}$$

which can be written in the equivalent form:

$$\begin{array}{rcl} -\min\limits_{(u,v,r,s)\in R^{m+k+n+m}} \frac{1}{2}(\|r\|^2 + \|B'u\|^2 + \|G'v\|^2 + 2v'GB'u - \|d\|^2 + \|s\|^2) & + & c'r \\ s.t. \; -b + B(\frac{B'u+G'v-d}{\varepsilon}) + A\frac{r}{\varepsilon} = \frac{s}{\varepsilon} & \geq & 0 \\ -h + G(\frac{B'u+G'v-d}{\varepsilon}) + E\frac{r}{\varepsilon} & = & 0 \\ r & \geq & 0. \end{array} \tag{8}$$

Note that at a solution of the exterior penalty problem (6) and the corresponding Wolfe dual (7) we have that:

$$\begin{array}{rcl} r = z_1 & = & (A'u + E'v - c)_+ \\ s = z_2 & = & (-u)_+. \end{array} \tag{9}$$

Define now:

$$\begin{array}{rcl} x & := & \frac{r}{\varepsilon} = \frac{1}{\varepsilon}(A'u + E'v - c)_+ \\ y & := & \frac{1}{\varepsilon}(B'u + G'v - d), \end{array} \tag{10}$$

where the equality in (10) follows from (9). Substituting (10) in (8) gives, after some algebra, the optimization problem (11) below. It is easiest to see that (8) follows from (11) if we substitute for x and y from (10) in (11) below and note that $0 \leq r = \varepsilon x$ and that $0 \leq s = \varepsilon(Ax + By - b)$ which follow from the constraints of (8) and the definitions (10) of x and y.

$$\begin{array}{rcccl} -\min\limits_{(x,y)\in R^{n+\ell}} & c'x + d'y & + & \frac{\varepsilon}{2}(\|x\|^2 + \|y\|^2 + \|Ax + By - b\|^2) \\ & Ax + By & \geq & b \\ & Ex + Gy & = & h \\ & x & \geq & 0. \end{array} \tag{11}$$

This convex quadratic program (11) is feasible, because the linear program (1) is feasible. It is solvable for any $\varepsilon > 0$ (Frank and Wolfe, 1956) because its objective function is bounded below since it is a strongly convex quadratic function in (x,y). Since the dual exterior penalty minimization problem objective (3) or equivalently (6) is bounded below by the negative of the objective function of (11) by the weak duality theorem (Mangasarian, 1994, Theorem 8.2.3), hence (3) is solvable for any $\varepsilon > 0$. By the perturbation theory of linear programs (Mangasarian and Meyer, 1979), it follows that for $\varepsilon \in (0, \overline{\varepsilon}]$, for some $\overline{\varepsilon} > 0$, (x,y) as defined in (10) or equivalently (4), solve the linear program (1) and additionally minimize the expression (5) over the solution set of the original linear program (1).□

A more direct, but just as laborious and rather unintuitive proof of Proposition 1 can be given by showing that the KKT necessary and sufficient optimality conditions for (11) follow from the necessary and sufficient optimality conditions of setting the gradient of the exterior penalty problem (3) equal to zero. We do not give that proof here because it does not justify how the quadratic perturbation terms of (11) arose, but it merely starts with these terms as given.

We turn now to an implementation of this result for various 1-norm SVMs.

3. 1-Norm SVMs as Unconstrained Minimization Problems

We consider first the 1-norm linear SVM binary classification problem (Mangasarian, 2000; Bradley and Mangasarian, 1998; Fung and Mangasarian, 2004):

$$\begin{array}{rcl} \min\limits_{(w,\gamma,y)} & \nu\|y\|_1 + \|w\|_1 & \\ \text{s.t.} & D(Aw - e\gamma) + y & \geq e \\ & y & \geq 0, \end{array} \tag{12}$$

where, with some abuse of notation by multiple representation, we let the $m \times n$ matrix A in this section represent m points in R^n to be separated to the best extent possible by a separating plane:

$$x'w = \gamma, \tag{13}$$

according to the class of each row of A as given by the $m \times m$ diagonal matrix D with elements $D_{ii} = \pm 1$. The objective term $\|y\|_1$ minimizes the classification error weighted with the positive parameter ν while the term $\|w\|_1$ maximizes the ∞-norm margin (Mangasarian, 1999) between the bounding planes $x'w = \gamma \pm 1$ that approximately bound each of the two classes of points represented by A. It is well known (Bradley and Mangasarian, 1998; Fung and Mangasarian, 2004) that using $\|w\|_1$ in the objective function of (12) instead of the standard 2-norm squared term $\|w\|^2$ (Vapnik, 2000; Schölkopf and Smola, 2002) results in input space feature selection by suppressing many components of w, whereas the standard 2-norm SVM does not suppress any components of w in general. We convert (12) to an explicit linear program as in (Fung and Mangasarian, 2004) by setting:

$$w = p - q, \;\; p \geq 0, \; q \geq 0, \tag{14}$$

which results in the linear program:

$$\begin{array}{rll} \min\limits_{(p,q,\gamma,y)} & \nu e'y + e'(p+q) & \\ \text{s.t.} & D(A(p-q) - e\gamma) + y & \geq \;\; e \\ & p,q,y & \geq \;\; 0. \end{array} \tag{15}$$

We note immediately that this linear program is solvable because it is feasible and its objective function is bounded below by zero. Hence, Proposition 1 can be utilized to yield the following unconstrained reformulation of the problem.

Proposition 2 Exact 1-Norm SVM Solution via Unconstrained Minimization *The unconstrained dual exterior penalty problem for the 1-norm SVM (15):*

$$\min_{u \in R^m} \; -\varepsilon e'u + \frac{1}{2}(\|(A'Du - e)_+\|^2 + \|(-A'Du - e)_+\|^2 + (-e'Du)^2 + \|(u - \nu e)_+\|^2 + \|(-u)_+\|^2), \tag{16}$$

is solvable for all $\varepsilon > 0$. For any $\varepsilon \in (0, \bar{\varepsilon}]$ for some $\bar{\varepsilon} > 0$, any solution u of (16) generates an exact solution of the 1-norm SVM classification problem (12) as follows:

$$\begin{array}{lcl} w = p - q = & = & \frac{1}{\varepsilon}((A'Du - e)_+ - (-A'Du - e)_+), \\ \gamma & = & -\frac{1}{\varepsilon} e'Du, \\ y & = & \frac{1}{\varepsilon}(u - \nu e)_+. \end{array} \tag{17}$$

In addition this (w, γ, y) minimizes:

$$\|w\|^2 + \gamma^2 + \|y\|^2 + \|D(Aw - e\gamma) + y - e\|^2, \tag{18}$$

over the solution set of the 1-norm SVM classification problem (12).

We note here the similarity between our unconstrained penalty minimization problem (16) and the corresponding problem of (Fung and Mangasarian, 2004, Equation 23). But, we also note a major difference. In the latter, a penalty parameter α multiplies the term $\|(-u)_+\|^2$ of equation (16) above and is required to approach ∞ in order to obtain an exact solution to the original problem (12). Thus, the solution obtained by (Fung and Mangasarian, 2004, Equation 23) for any finite α is only approximate, as pointed out there. Furthermore, our solution to (16) here minimizes the expression (18) rather than being merely an approximate least 2-norm solution as is the case in (Fung and

Mangasarian, 2004, Equation 11). However the generalized Newton method prescribed in (Fung and Mangasarian, 2004) for a sequence $\{\alpha \uparrow \infty\}$, is applicable here with $\alpha = 1$. For completeness we state that result here. To do that we let $f(u)$ denote the exterior penalty function (16). Then the gradient and generalized Hessian as defined in the Introduction are given as follows.

$$\begin{array}{rcl} \nabla f(u) & = & -\varepsilon e + DA(A'Du-e)_+ - DA(-A'Du-e)_+ \\ & & +Dee'Du + (u-\nu e)_+ - (-u)_+. \end{array} \tag{19}$$

$$\begin{array}{rcl} \partial^2 f(u) & = & DA(diag((A'Du-e)_* + (-A'Du-e)_*)A'D \\ & & +Dee'D + diag((u-\nu e)_* + (-u)_*) \\ & = & DA(diag(|A'Du|-e)_*)A'D \\ & & +Dee'D + diag((u-\nu e)_* + (-u)_*), \end{array} \tag{20}$$

where the last equality follows from the equality:

$$(a-1)_* + (-a-1)_* = (|a|-1)_*. \tag{21}$$

To handle a nonlinear symmetric kernel $K(A,B)$ that maps $R^{m\times n} \times R^{n\times \ell}$ into $R^{m\times \ell}$ and which generates, instead of the separating plane (13), the nonlinear separating surface:

$$K(x',A')Dv = \gamma, \tag{22}$$

all we need to do is essentially to make the replacement:

$$A \longrightarrow K(A,A')D, \tag{23}$$

which we justify now. For a linear kernel $K(A,A') = AA'$, we have that $w = A'Dv$, where v is a dual variable (Mangasarian, 2000) and the primal linear programming SVM (15) becomes upon using $w = p - q = A'Dv$ and minimizing the 1-norm of v in the objective instead that of w:

$$\begin{array}{rll} \min\limits_{(v,\gamma,y)} & \nu e'y + \|v\|_1 & \\ \text{s.t.} & D(AA'Dv - e\gamma) + y & \geq e \\ & y & \geq 0. \end{array} \tag{24}$$

Setting:

$$v = r - s, \;\; r \geq 0, \; s \geq 0, \tag{25}$$

the linear program (24) becomes:

$$\begin{array}{rll} \min\limits_{(r,s,\gamma,y)} & \nu e'y + e'(r+s) & \\ \text{s.t.} & D(AA'D(r-s) - e\gamma) + y & \geq e \\ & r,s,y & \geq 0, \end{array} \tag{26}$$

which is the linear kernel SVM in terms of the dual variable $v = r - s$. If we replace the linear kernel AA' in (26) by the nonlinear kernel $K(A,A')$ we obtain the nonlinear kernel linear program:

$$\begin{array}{rll} \min\limits_{(r,s,\gamma,y)} & \nu e'y + e'(r+s) & \\ \text{s.t.} & D(K(A,A')D(r-s) - e\gamma) + y & \geq e \\ & r,s,y & \geq 0. \end{array} \tag{27}$$

We immediately note that the linear program (15) is identical to the linear program (27) if we make the replacement (23).

Finally, a word regarding the choice of ε in Propositions 1 and 2. Computationally in (Fung and Mangasarian, 2004) this does not seem to be critical and is effectively addressed as follows. By (Lucidi, 1987, Corollary 3.2), if for two successive values of ε: $\varepsilon^1 > \varepsilon^2$, the corresponding solutions of the ε-perturbed quadratic programs (11) are equal, then under certain assumptions these equal successive solutions constitute a solution of the linear programs (1) or (12) that also minimize the quadratic perturbations (5) or (18). This result can be implemented computationally by using an ε, which when decreased by some factor yields the same solution to (1) or (12). In our computational results this turned out to either 4×10^{-4} or 10^{-6}.

We state now our generalized Newton algorithm for solving the unconstrained minimization problem (16) as follows.

Algorithm 3 Generalized Newton Algorithm for (16) *Let $f(u)$, $\nabla f(u)$ and $\partial^2 f(u)$ be defined by (16),(19) and (20). Set the parameter values* ν, ε, δ, *tolerance* tol, *and* imax *(typically: $\varepsilon \in [10^{-6}, 4 \times 10^{-4}]$ for linear SVMs and $\varepsilon \in [10^{-9}, 1]$ nonlinear SVMs, $tol = 10^{-3}$, $imax = 50$, while ν and δ are set by a tuning procedure). Start with any $u^0 \in R^m$. For $i = 0, 1, \ldots$:*

(I) $u^{i+1} = u^i - \lambda_i(\partial^2 f(u^i) + \delta I)^{-1} \nabla f(u^i) = u^i + \lambda_i d^i$,
where the Armijo stepsize $\lambda_i = \max\{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ is such that:

$$f(u^i) - f(u^i + \lambda_i d^i) \geq -\frac{\lambda_i}{4} \nabla f(u^i)' d^i, \tag{28}$$

and d^i is the modified Newton direction:

$$d^i = -(\partial^2 f(u^i) + \delta I)^{-1} \nabla f(u^i). \tag{29}$$

In other words, start with $\lambda_i = 1$ and keep multiplying λ_i by $\frac{1}{2}$ until (28) is satisfied.

(II) Stop if $\|u^i - u^{i+1}\| \leq tol$ or $i = imax$. Else, set $i = i + 1$ and go to (I).

(III) Define the solution of the 1*-norm SVM (12) with least quadratic perturbation (18) by (17) with $u = u^i$.*

We state a convergence result for this algorithm now.

Proposition 4 *Let $tol = 0$, $imax = \infty$ and let $\varepsilon > 0$ be sufficiently small. Each accumulation point $\bar{u}$ of the sequence $\{u^i\}$ generated by Algorithm 3 solves the exterior penalty problem (16). The corresponding $(\bar{w}, \bar{\gamma}, \bar{y})$ obtained by setting u to $\bar{u}$ in (17) is an exact solution to the primal* 1*-norm SVM (12) which in addition minimizes the quadratic perturbation (18) over the solution set of (12).*

Proof That each accumulation point $\bar{u}$ of the sequence $\{u^i\}$ solves the minimization problem (13) follows from exterior penalty results (Fiacco and McCormick, 1968; Bertsekas, 1999) and standard unconstrained descent methods such as (Mangasarian, 1995, Theorem 2.1, Examples 2.1(i), 2.2(iv)) and the facts that the direction choice d^i of (24) satisfies, for some $c > 0$:

$$\begin{array}{rcl} -\nabla f(u^i)' d^i & = & \nabla f(u^i)'(\delta I + \partial^2 f(u^i))^{-1} \nabla f(u^i) \\ & \geq & c \|\nabla f(u^i)\|^2, \end{array} \tag{30}$$

and that we are using an Armijo stepsize (28). The last statement of the theorem follows from Proposition 2.□

We turn now to minimal kernel function approximation.

4. Minimal Kernel Function Approximation as Unconstrained Minimization Problems

We consider here the problem of constructing a kernel function approximation from a given number of function values using the 1-norm to minimize both the error in the approximation as well as the weights of the kernel functions. Utilizing the 1-norm in minimizing the kernel weights suppresses unnecessary kernel functions similar to the approach of (Mangasarian et al., 2004) except that we shall solve the resulting linear program here through an unconstrained minimization reformulation. Also, for simplicity we shall not incorporate prior knowledge as was done in (Mangasarian et al., 2004).

We consider m given function values $b \in R^m$ associated with m n-dimensional vectors represented by the m rows of the $m \times n$ matrix A. We shall fit the data points by a linear combination of symmetric kernel functions as follows:

$$K(A,A')v + e\gamma \approx b, \tag{31}$$

where the unknown parameters $v \in R^m$ and $\gamma \in R$ are determined by minimizing the 1-norm of the approximation error weighted by $\nu > 0$ and the 1-norm of v as follows:

$$\min_{(v,\gamma)\in R^{n+1}} \nu\|K(A,A')v + e\gamma - b\|_1 + \|v\|_1. \tag{32}$$

Setting

$$\begin{array}{rcl} v & = & r - s,\ r \geq 0,\ s \geq 0, \\ K(A,A')v + e\gamma - b & = & y - z,\ y \geq 0,\ z \geq 0, \end{array} \tag{33}$$

we obtain the following linear program:

$$\begin{array}{lrcl} \min\limits_{(r,s,\gamma,y,z)} & \nu e'(y+z) + e'(r+s) & & \\ \text{s.t.} & K(A,A')(r-s) + e\gamma - y + z & = & b \\ & r,s,y,z & \geq & 0, \end{array} \tag{34}$$

which is similar to the nonlinear kernel SVM classifier linear programming formulation (27) with equality constraints replacing inequality constraints. We also note that this linear program is solvable because it is feasible and its objective function is bounded below by zero. Hence, Proposition 1 can be utilized to yield the following unconstrained reformulation of the problem.

Proposition 5 Exact 1-Norm Nonlinear SVM Approximation via Unconstrained Minimization *The unconstrained dual exterior penalty problem for the 1-norm SVM approximation (34):*

$$\begin{array}{l} \min\limits_{u\in R^m} -\varepsilon b'u + \frac{1}{2}(\|(K(A,A')u - e)_+\|^2 + \|(-K(A,A')u - e)_+\|^2 + \\ \qquad\qquad (e'u)^2 + \|(-u - \nu e)_+\|^2 + \|(u - \nu e)_+\|^2), \end{array} \tag{35}$$

is solvable for all $\varepsilon > 0$. For any $\varepsilon \in (0, \bar{\varepsilon}]$ for some $\bar{\varepsilon} > 0$, any solution u of (35) generates an exact of the 1-norm SVM approximation problem (32) as follows:

$$\begin{array}{lcl} v = r - s = & = & \frac{1}{\varepsilon}((K(A,A')u - e)_+ - (-K(A,A')u - e)_+), \\ \gamma & = & \frac{1}{\varepsilon}e'u, \\ y & = & \frac{1}{\varepsilon}(-u - \nu e)_+, \\ z & = & \frac{1}{\varepsilon}(u - \nu e)_+ \end{array} \tag{36}$$

In addition this (r,s,γ,y,z) *minimizes:*

$$\|r\|^2+\|s\|^2+\gamma^2+\|y\|^2+\|z\|^2, \tag{37}$$

over the solution set of the 1*-norm SVM classification problem (34).*

Computational results utilizing the linear programming formulation (32) with prior knowledge in (Mangasarian et al., 2004) but using the simplex method of solution is effective for solving approximation problems. The unconstrained minimization formulation (35) is another method of solution which can also handle such problems without prior knowledge as well as with prior knowledge with appropriate but straightforward modifications.

We turn now to our computational results.

5. Computational Results

Computational testing was carried on a 3 Ghz Pentium 4 machine with 2GB of memory running CentOS 4 Linux and utilizing the CPLEX 7.1 (ILO, 2003) linear programming package within MATLAB 7.1 (MATLAB, 1994-2001). We tested our algorithm on six publicly available data sets. Five from the UCI Machine Learning Repository Murphy and Aha (1992): Ionosphere, Cleveland Heart, Pima Indians, BUPA Liver and Housing. The sixth data set, Galaxy Dim, is available from Odewahn et al. (1992). The results are summarized in Tables 1 and 2.

For the linear classifier (13) we compare in Table 1, NLPSVM (Fung and Mangasarian, 2004), CPLEX (ILO, 2003) and our Generalized LPNewton Algorithm for (16), on six public data sets using ten-fold cross validation. NLPSVM is essentially identical to our algorithm, except that it requires a penalty parameter multiplying the last term of (16) to approach infinity. CPLEX uses the standard linear programming package CPLEX (ILO, 2003) to solve (26). We note that our method LPNewton is faster than both NLPSVM and CPLEX on all six data sets and gives the best feature suppression based on the average number of features used by the linear classifier (13). NLPSVM has the best test set correctness on two of the data sets, and comparable correctness on the other four. The Armijo step size was not needed in either NLPSVM or LPNewton. Tuning on 10% of the training set was used to determine the parameters ν and δ from the sets $\{2^{-12},\ldots,2^{12}\}$ and $\{10^{-3},\ldots,.10^{3}\}$ respectively. Epsilon was set to the value 4.00E-04 used in (Fung and Mangasarian, 2004) for NLPSM and to 1.00E-06 for our LPNewton algorithm.

For the nonlinear classifier (22) we compare in Table 2, NLPSVM (Fung and Mangasarian, 2004), CPLEX (ILO, 2003) and our Generalized LPNewton Algorithm 3 for (27), on three public data sets using ten-fold cross validation. We note again that our method LPNewton is faster than both NLPSVM and CPLEX on all three data sets and gives the best reduction in the number of kernel functions utilized, on two of the data sets, based on the cardinality of $v=r-s$ as defined in (25) and (27). Best test set correctness was achieved on two data sets by our method and it was a close second on the third data set. Again the Armijo step size was not needed in either NLPSVM or LPNewton. Tuning and choice of the parameters ν and ε was done as for the linear classifier above. A Gaussian kernel was used for all three methods and data sets with the Gaussian parameter tuned from the set $\{2^{-12},\ldots,2^{12}\}$.

Data Set/Size	Algorithm	Iters	Time	Train %	Test %	Feat	Eps
Ionosphere	NLPSVM	69	0.1796	92.6254	83.8016	20.6	4e-4
Ionosphere	CPLEX		0.179	**92.6255**	85.4841	25.1	
Ionosphere	LPNewton	**30.7**	**0.0767**	89.6169	**87.1825**	**9.6**	1e-6
351× 34							
BUPA Liver	NLPSVM	100	0.1062	70.1791	**67.916**	5.9	4e-4
BUPA Liver	CPLEX		0.2278	**70.4994**	67.2941	6	
BUPA Liver	LPNewton	**63.3**	**0.0623**	69.1814	67.563	**5.2**	1e-6
345× 6							
Pima Indians	NLPSVM	93.2	0.2169	73.5809	72.6692	6.8	4e-4
Pima Indians	CPLEX		1.1707	**76.8086**	**75.2683**	5.8	
Pima Indians	LPNewton	**40.6**	**0.0904**	76.0563	75.0051	**4.6**	1e-6
768× 8							
Cleveland	NLPSVM	42.2	0.0515	85.6742	84.1609	7.5	4e-4
Cleveland	CPLEX		0.1409	**85.9348**	84.1609	8.4	
Cleveland	LPNewton	**25.3**	**0.028**	85.7478	**84.5287**	**7.1**	1e-6
297× 13							
Housing	NLPSVM	66.6	0.0891	83.9049	83.8078	9.1	4e-4
Housing	CPLEX		0.363	**86.8035**	**84.3882**	10.5	
Housing	LPNewton	**57.4**	**0.0781**	85.6626	83.2078	**7.7**	1e-6
506× 13							
Galaxy Dim	NLPSVM	97.5	1.097	94.4392	94.4415	5.9	4e-4
Galaxy Dim	CPLEX		12.5357	**95.5153**	**95.5153**	11.5	
Galaxy Dim	LPNewton	**39.2**	**0.4297**	94.4948	94.5131	**4.8**	1e-6
4192× 14							

Table 1: **Comparison of the Linear Classifier (13) obtained by NLPSVM (Fung and Mangasarian, 2004), CPLEX (ILO, 2003) and our Generalized LPNewton Algorithm 3 for (16) on six public data sets. Time is for one fold in seconds, Train and Test corectness is the average over ten folds and Features (Feat) denote the average number over ten folds of input space features utilized by the linear classifier. Epsilon (Eps) is the finite parameter defined in (16). Best result is in bold. Note that LPNewton is fastest and has least features.**

Data Set/Size	Algorithm	Iters	Time	Train %	Test %	Card(v)	Eps
Ionosphere	NLPSVM	81.7	0.181	92.0242	89.4683	18.5	4e-4
Ionosphere	CPLEX		0.1555	**94.7773**	**91.4683**	15.5	
Ionosphere	LPNewton	**36.5**	**0.103**	92.5297	91.1587	**11.2**	1e-6
351× 34							
BUPA	NLPSVM	88.3	0.1706	68.8514	65.2521	**15.5**	4e-4
BUPA	CPLEX		0.2552	**74.1061**	69.2521	17.3	
BUPA	LPNewton	**88.2**	**0.1345**	73.6572	**70.6975**	25.5	1e+0
345× 6							
Cleveland	NLPSVM	84.6	0.1128	83.168	80.4368	9.1	4e-4
Cleveland	CPLEX		0.1097	**85.0383**	81.8161	11.8	
Cleveland	LPNewton	**80.2**	**0.1061**	83.0151	**82.8621**	**5.6**	1e-9
297× 13							

Table 2: **Comparison of the Nonlinear Classifier (22) obtained by NLPSVM (Fung and Mangasarian, 2004), CPLEX (ILO, 2003) and our Generalized LPNewton Algorithm 3 for (27) on three public data sets. Time for one fold is in seconds, Train and Test corectness is on ten folds. Card(v) denotes the average number of nonzero components of $v = r - s$ as defined in (25) and (27) and hence that is the number of kernel functions utilized by the nonlinear classifier (22). Epsilon (Eps) is the finite parameter defined in (16) with the replacement (23) of A by $K(A,A')D$. Features (Feat) denotes the average number of features over ten folds. Reduced SVM (RSVM) (Lee and Mangasarian, 2001) was used to speed all computations by using the reduced kernel $K(A,\bar{A}')$ where $\frac{m}{10}$ randomly chosen rows of A constitute the rows of rows of $\bar{A}$. Best result is in bold. Note that LPNewton is fastest.**

6. Conclusion and Outlook

We have derived an unconstrained differentiable convex minimization reformulation of a most general linear program and have applied it to 1-norm classification and approximation problems. Very effective computational results of our method on special cases of general linear programs (Mangasarian, 2004) and an approximate version for support vector machine classification (Fung and Mangasarian, 2004), as well as computational results presented in Section 5, lead us to believe that the proposed unconstrained reformulation of very general linear programs and support vector machines is a very promising computational method for solving such problems as well as extensions to knowledge-based formulations (Mangasarian, 2005; Fung et al., 2003; Mangasarian et al., 2004).

Acknowledgments

I am indebted to my PhD student Michael Thompson for the computational results given in this work. Research described in this Data Mining Institute Report 05-03, August 2005, was supported by National Science Foundation Grants CCR-0138308 and IIS-0511905, the Microsoft Corporation and ExxonMobil. Revised January 2006.

References

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, second edition, 1999.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.

G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, 1963.

F. Facchinei. Minimization of SC^1 functions and the Maratos effect. *Operations Research Letters*, 17:131–137, 1995.

A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York, NY, 1968.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

G. Fung and O. L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, July 2004. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/02-03.ps.

G. Fung, O. L. Mangasarian, and J. Shavlik. Knowledge-based support vector machine classifiers. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 521–528. MIT Press, Cambridge, MA, October 2003. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-09.ps.

J.-B. Hiriart-Urruty, J. J. Strodiot, and V. H. Nguyen. Generalized hessian matrix and second-order optimality conditions for problems with C^{L1} data. *Applied Mathematics and Optimization*, 11: 43–56, 1984.

ILOG CPLEX 9.0 User's Manual. ILOG, Incline Village, Nevada, 2003. http://www.ilog.com/products/cplex/.

Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the First SIAM International Conference on Data Mining, Chicago, April 5-7, 2001, CD-ROM*, 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-07.ps.

S. Lucidi. A new result in the theory and computation of the least-norm solution of a linear program. *Journal of Optimization Theory and Applications*, 55:103–117, 1987.

O. L. Mangasarian. *Nonlinear Programming*. SIAM, Philadelphia, PA, 1994.

O. L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33(6):1916–1925, 1995. ftp://ftp.cs.wisc.edu/tech-reports/reports/1993/tr1145.ps.

O. L. Mangasarian. A finite Newton method for classification problems. Technical Report 01-11, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, December 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-11.ps.*Optimization Methods and Software* 17, 2002, 913-929.

O. L. Mangasarian. A Newton method for linear programming. *Journal of Optimization Theory and Applications*, 121:1–18, 2004. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/02-02.ps.

O. L. Mangasarian. Knowledge-based linear programming. *SIAM Journal on Optimization*, 15: 375–382, 2005. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/03-04.ps.

O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24:15–23, 1999. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-07r.ps.

O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.

O. L. Mangasarian and R. R. Meyer. Nonlinear perturbation of linear programs. *SIAM Journal on Control and Optimization*, 17(6):745–752, November 1979.

O. L. Mangasarian, J. W. Shavlik, and E. W. Wild. Knowledge-based kernel approximation. *Journal of Machine Learning Research*, 5:1127–1141, 2004. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/03-05.ps.

MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2001. http://www.mathworks.com.

P. M. Murphy and D. W. Aha. UCI machine learning repository, 1992. www.ics.uci.edu/∼mlearn/MLRepository.html.

S. Odewahn, E. Stockwell, R. Pennington, R. Humphreys, and W. Zumach. Automated star/galaxy discrimination with neural networks. *Astronomical Journal*, 103(1):318–331, 1992.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, second edition, 2000.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-Norm support vector machines. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16–NIPS2003*. MIT Press, 2004.

Journal of Machine Learning Research 7 (2006) 1531–1565 Submitted 10/05; Published 7/06

Large Scale Multiple Kernel Learning

Sören Sonnenburg SOEREN.SONNENBURG@FIRST.FRAUNHOFER.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany

Gunnar Rätsch GUNNAR.RAETSCH@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory of the Max Planck Society
Spemannstrasse 39
Tübingen, Germany

Christin Schäfer CHRISTIN.SCHAEFER@FIRST.FRAUNHOFER.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany

Bernhard Schölkopf BERNHARD SCHOELKOPF@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076, Tübingen, Germany

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

While classical kernel-based learning algorithms are based on a single kernel, in practice it is often desirable to use multiple kernels. Lanckriet et al. (2004) considered conic combinations of kernel matrices for classification, leading to a convex quadratically constrained quadratic program. We show that it can be rewritten as a semi-infinite linear program that can be efficiently solved by recycling the standard SVM implementations. Moreover, we generalize the formulation and our method to a larger class of problems, including regression and one-class classification. Experimental results show that the proposed algorithm works for hundred thousands of examples or hundreds of kernels to be combined, and helps for automatic model selection, improving the interpretability of the learning result. In a second part we discuss general speed up mechanism for SVMs, especially when used with *sparse* feature maps as appear for string kernels, allowing us to train a string kernel SVM on a 10 million real-world splice data set from computational biology. We integrated multiple kernel learning in our machine learning toolbox `SHOGUN` for which the source code is publicly available at `http://www.fml.tuebingen.mpg.de/raetsch/projects/shogun`.

Keywords: multiple kernel learning, string kernels, large scale optimization, support vector machines, support vector regression, column generation, semi-infinite linear programming

1. Introduction

Kernel based methods such as support vector machines (SVMs) have proven to be powerful for a wide range of different data analysis problems. They employ a so-called kernel function $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$ which intuitively computes the similarity between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$. The result of SVM

learning is an α-weighted linear combination of kernels with a bias b

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{k}(\mathbf{x}_i, \mathbf{x}) + b\right), \tag{1}$$

where the $\mathbf{x}_i$, $i = 1,\ldots,N$ are labeled training examples ($y_i \in \{\pm 1\}$).

Recent developments in the literature on SVMs and other kernel methods have shown the need to consider multiple kernels. This provides flexibility and reflects the fact that typical learning problems often involve multiple, heterogeneous data sources. Furthermore, as we shall see below, it leads to an elegant method to interpret the results, which can lead to a deeper understanding of the application.

While this so-called "multiple kernel learning" (MKL) problem can in principle be solved via cross-validation, several recent papers have focused on more efficient methods for multiple kernel learning (Chapelle et al., 2002; Bennett et al., 2002; Grandvalet and Canu, 2003; Ong et al., 2003; Bach et al., 2004; Lanckriet et al., 2004; Bi et al., 2004).

One of the problems with kernel methods compared to other techniques is that the resulting decision function (1) is hard to interpret and, hence, is difficult to use in order to extract relevant knowledge about the problem at hand. One can approach this problem by considering convex combinations of K kernels, i.e.

$$\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{K} \beta_k \mathbf{k}_k(\mathbf{x}_i, \mathbf{x}_j) \tag{2}$$

with $\beta_k \geq 0$ and $\sum_{k=1}^{K} \beta_k = 1$, where each kernel $\mathbf{k}_k$ uses only a distinct set of features. For appropriately designed sub-kernels $\mathbf{k}_k$, the optimized combination coefficients can then be used to understand which features of the examples are of importance for discrimination: if one is able to obtain an accurate classification by a *sparse* weighting β_k, then one can quite easily interpret the resulting decision function. This is an important property missing in current kernel based algorithms. Note that this is in contrast to the kernel mixture framework of Bennett et al. (2002) and Bi et al. (2004) where each kernel *and* each example are assigned an independent weight and therefore do not offer an easy way to interpret the decision function. We will illustrate that the considered MKL formulation provides useful insights and at the same time is very efficient.

We consider the framework proposed by Lanckriet et al. (2004), which results in a convex optimization problem - a quadratically-constrained quadratic program (QCQP). This problem is more challenging than the standard SVM QP, but it can in principle be solved by general-purpose optimization toolboxes. Since the use of such algorithms will only be feasible for small problems with few data points and kernels, Bach et al. (2004) suggested an algorithm based on sequential minimization optimization (SMO Platt, 1999). While the kernel learning problem is convex, it is also non-smooth, making the direct application of simple local descent algorithms such as SMO infeasible. Bach et al. (2004) therefore considered a smoothed version of the problem to which SMO can be applied.

In the first part of the paper we follow a different direction: We reformulate the binary classification MKL problem (Lanckriet et al., 2004) as a *semi-infinite linear program*, which can be efficiently solved using an off-the-shelf LP solver and a standard SVM implementation (cf. Section 2.1 for details). In a second step, we show how easily the MKL formulation and the algorithm is generalized to a much larger class of convex loss functions (cf. Section 2.2). Our proposed *wrapper method* works for any kernel and many loss functions: In order to obtain an efficient MKL

algorithm for a new loss function, it now suffices to have an LP solver and the corresponding single kernel algorithm (which is assumed to be efficient). Using this general algorithm we were able to solve MKL problems with up to 30,000 examples and 20 kernels within reasonable time.[1]

We also consider a *chunking* algorithm that can be considerably more efficient, since it optimizes the SVM α multipliers and the kernel coefficients β at the same time. However, for large scale problems it needs to compute and cache the K kernels separately, instead of only one kernel as in the single kernel algorithm. This becomes particularly important when the sample size N is large. If, on the other hand, the number of kernels K is large, then the amount of memory available for caching is drastically reduced and, hence, kernel caching is not effective anymore. (The same statements also apply to the SMO-like MKL algorithm proposed in Bach et al. (2004).)

Since kernel caching cannot help to solve large scale MKL problems, we sought for ways to avoid kernel caching. This is of course not always possible, but it certainly is for the class of kernels where the feature map $\Phi(\mathbf{x})$ can be explicitly computed and computations with $\Phi(\mathbf{x})$ can be implemented efficiently. In Section 3.1.1 we describe several string kernels that are frequently used in biological sequence analysis and exhibit this property. Here, the feature space can be very high dimensional, but $\Phi(\mathbf{x})$ is typically very sparse. In Section 3.1.2 we discuss several methods for efficiently dealing with high dimensional sparse vectors, which not only is of interest for MKL but also for speeding up ordinary SVM classifiers. Finally, we suggest a modification of the previously proposed chunking algorithm that exploits these properties (Section 3.1.3). In the experimental part we show that the resulting algorithm is more than 70 times faster than the plain chunking algorithm (for 50,000 examples), even though large kernel caches were used. Also, we were able to solve MKL problems with up to one million examples and 20 kernels and a 10 million real-world splice site classification problem from computational biology. We conclude the paper by illustrating the usefulness of our algorithms in several examples relating to the interpretation of results and to automatic model selection. Moreover, we provide an extensive benchmark study comparing the effect of different improvements on the running time of the algorithms.

We have implemented all algorithms discussed in this work in C++ with interfaces to *Matlab*, *Octave*, *R* and *Python*. The source code is freely available at

`http://www.fml.tuebingen.mpg.de/raetsch/projects/shogun`.

The examples used to generate the figures are implemented in *Matlab* using the *Matlab* interface of the `SHOGUN` toolbox. They can be found together with the data sets used in this paper at `http://www.fml.tuebingen.mpg.de/raetsch/projects/lsmkl`.

2. A General and Efficient Multiple Kernel Learning Algorithm

In this section we first derive our MKL formulation for the binary classification case and then show how it can be extended to general cost functions. In the last subsection we will propose algorithms for solving the resulting semi-infinite linear programs (SILPs).

2.1 Multiple Kernel Learning for Classification Using SILP

In the multiple kernel learning problem for binary classification one is given N data points $(\mathbf{x}_i, y_i)$ ($y_i \in \{\pm 1\}$), where $\mathbf{x}_i$ is translated via K mappings $\Phi_k(\mathbf{x}) \mapsto \mathbb{R}^{D_k}$, $k = 1, \ldots, K$, from the input into K

1. The results are not shown.

feature spaces $(\Phi_1(\mathbf{x}_i),\ldots,\Phi_K(\mathbf{x}_i))$ where D_k denotes the dimensionality of the k-th feature space. Then one solves the following optimization problem (Bach et al., 2004), which is equivalent to the linear SVM for $K = 1$:[2]

MKL Primal for Classification

$$\begin{aligned} \min \quad & \frac{1}{2}\left(\sum_{k=1}^{K}\|\mathbf{w}_k\|_2\right)^2 + C\sum_{i=1}^{N}\xi_i \\ \text{w.r.t.} \quad & \mathbf{w}_k \in \mathbb{R}^{D_k}, \xi \in \mathbb{R}^N, b \in \mathbb{R}, \\ \text{s.t.} \quad & \xi_i \geq 0 \text{ and } y_i\left(\sum_{k=1}^{K}\langle \mathbf{w}_k, \Phi_k(\mathbf{x}_i)\rangle + b\right) \geq 1-\xi_i, \ \ \forall i = 1,\ldots,N \end{aligned} \tag{3}$$

Note that the problem's solution can be written as $\mathbf{w}_k = \beta_k \mathbf{w}_k'$ with $\beta_k \geq 0$, $\forall k = 1,\ldots,K$ and $\sum_{k=1}^{K}\beta_k = 1$ (Bach et al., 2004). Note that therefore the ℓ_1-norm of β is constrained to one, while one is penalizing the ℓ_2-norm of $\mathbf{w}_k$ in each block k separately. The idea is that ℓ_1-norm constrained or penalized variables tend to have sparse optimal solutions, while ℓ_2-norm penalized variables do not (e.g. Rätsch, 2001, Chapter 5.2). Thus the above optimization problem offers the possibility to find sparse solutions on the block level with non-sparse solutions within the blocks.

Bach et al. (2004) derived the dual for problem (3). Taking their problem (D_K), squaring the constraints on gamma, multiplying the constraints by $\frac{1}{2}$ and finally substituting $\frac{1}{2}\gamma^2 \mapsto \gamma$ leads to the to the following *equivalent* multiple kernel learning dual:

MKL Dual for Classification

$$\begin{aligned} \min \quad & \gamma - \sum_{i=1}^{N}\alpha_i \\ \text{w.r.t.} \quad & \gamma \in \mathbb{R}, \alpha \in \mathbb{R}^N \\ \text{s.t.} \quad & \mathbf{0} \leq \alpha \leq \mathbf{1}C, \ \sum_{i=1}^{N}\alpha_i y_i = 0 \\ & \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{k}_k(\mathbf{x}_i,\mathbf{x}_j) \leq \gamma, \ \ \forall k = 1,\ldots,K \end{aligned}$$

where $\mathbf{k}_k(\mathbf{x}_i,\mathbf{x}_j) = \langle \Phi_k(\mathbf{x}_i), \Phi_k(\mathbf{x}_j)\rangle$. Note that we have one quadratic constraint per kernel ($S_k(\alpha) \leq \gamma$). In the case of $K = 1$, the above problem reduces to the original SVM dual. We will now move the term $-\sum_{i=1}^{N}\alpha_i$, into the constraints on γ. This can be equivalently done by adding $-\sum_{i=1}^{N}\alpha_i$ to both sides of the constraints and substituting $\gamma - \sum_{i=1}^{N}\alpha_i \mapsto \gamma$:

2. We assume $\mathrm{tr}(K_k) = 1$, $k = 1,\ldots,K$ and set d_j in Bach et al. (2004) to one.

MKL Dual* for Classification

$$\begin{aligned} \min \quad & \gamma \\ \text{w.r.t.} \quad & \gamma \in \mathbb{R}, \alpha \in \mathbb{R}^N \\ \text{s.t.} \quad & \mathbf{0} \le \alpha \le \mathbf{1}C, \ \sum_{i=1}^N \alpha_i y_i = 0 \\ & \underbrace{\frac{1}{2}\sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{k}_k(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^N \alpha_i}_{=:S_k(\alpha)} \le \gamma, \ \ \forall k = 1, \ldots, K \end{aligned} \tag{4}$$

In order to solve (4), one may solve the following saddle point problem: minimize

$$\mathcal{L} := \gamma + \sum_{k=1}^K \beta_k (S_k(\alpha) - \gamma) \tag{5}$$

w.r.t. $\alpha \in \mathbb{R}^N, \gamma \in \mathbb{R}$ (with $\mathbf{0} \le \alpha \le C\mathbf{1}$ and $\sum_i \alpha_i y_i = 0$), and maximize it w.r.t. $\beta \in \mathbb{R}^K$, where $\mathbf{0} \le \beta$. Setting the derivative w.r.t. to γ to zero, one obtains the constraint $\sum_{k=1}^K \beta_k = 1$ and (5) simplifies to: $\mathcal{L} = S(\alpha, \beta) := \sum_{k=1}^K \beta_k S_k(\alpha)$. While one *minimizes* the objective w.r.t. α, at the same time one *maximizes* w.r.t. the kernel weighting β. This leads to a

Min-Max Problem

$$\begin{aligned} \max_\beta \min_\alpha \quad & \sum_{k=1}^K \beta_k S_k(\alpha) \\ \text{w.r.t.} \quad & \alpha \in \mathbb{R}^N, \ \beta \in \mathbb{R}^K \\ \text{s.t.} \quad & 0 \le \alpha \le C \ , \ 0 \le \beta, \sum_{i=1}^N \alpha_i y_i = 0 \text{ and } \sum_{k=1}^K \beta_k = 1. \end{aligned} \tag{6}$$

This problem is very similar to Equation (9) in Bi et al. (2004) when "composite kernels," i.e. linear combinations of kernels are considered. There the first term of $S_k(\alpha)$ has been moved into the constraint, still β including the $\sum_{k=1}^K \beta_k = 1$ is missing.[3]

Assume α^* were the optimal solution, then $\theta^* := S(\alpha^*, \beta)$ would be minimal and, hence, $S(\alpha, \beta) \ge \theta^*$ for all α (subject to the above constraints). Hence, finding a saddle-point of (5) is equivalent to solving the following semi-infinite linear program:

Semi-Infinite Linear Program (SILP)

$$\begin{aligned} \max \quad & \theta \\ \text{w.r.t.} \quad & \theta \in \mathbb{R}, \beta \in \mathbb{R}^K \\ \text{s.t.} \quad & \mathbf{0} \le \beta, \ \sum_k \beta_k = 1 \text{ and } \sum_{k=1}^K \beta_k S_k(\alpha) \ge \theta \\ & \text{for all } \alpha \in \mathbb{R}^N \text{ with } \mathbf{0} \le \alpha \le C\mathbf{1} \text{ and } \sum_i y_i \alpha_i = 0 \end{aligned} \tag{7, 8}$$

3. In Bi et al. (2004) it is argued that the approximation quality of composite kernels is inferior to mixtures of kernels where a weight is assigned per example *and* kernel as in Bennett et al. (2002). For that reason and as no efficient methods were available to solve the composite kernel problem, they only considered mixtures of kernels and in the experimental validation used a uniform weighting in the composite kernel experiment. Also they did not consider to use composite kernels as a method to interpret the resulting classifier but looked at classification accuracy instead.

Note that this is a linear program, as θ and β are only linearly constrained. However there are infinitely many constraints: one for each $\alpha \in \mathbb{R}^N$ satisfying $0 \leq \alpha \leq C$ and $\sum_{i=1}^N \alpha_i y_i = 0$. Both problems (6) and (7) have the same solution. To illustrate that, consider β is fixed and we minimize α in (6). Let α^* be the solution that minimizes (6). Then we can increase the value of θ in (7) as long as none of the infinitely many α-constraints (8) is violated, i.e. up to $\theta = \sum_{k=1}^K \beta_k S_k(\alpha^*)$. On the other hand as we increase θ for a fixed α the maximizing β is found. We will discuss in Section 2.3 how to solve such semi-infinite linear programs.

2.2 Multiple Kernel Learning with General Cost Functions

In this section we consider a more general class of MKL problems, where one is given an *arbitrary* strictly convex and differentiable loss function, for which we derive its MKL SILP formulation. We will then investigate in this general MKL SILP using different loss functions, in particular the soft-margin loss, the ε-insensitive loss and the quadratic loss.

We define the MKL primal formulation for a strictly convex and differentiable loss function $L(f(\mathbf{x}), y)$ as:

MKL Primal for Generic Loss Functions

$$\begin{aligned} \min \quad & \frac{1}{2}\left(\sum_{k=1}^K \|\mathbf{w}_k\|\right)^2 + \sum_{i=1}^N L(f(\mathbf{x}_i), y_i) \\ \text{w.r.t.} \quad & \mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_K) \in \mathbb{R}^{D_1} \times \cdots \times \mathbb{R}^{D_K} \\ \text{s.t.} \quad & f(\mathbf{x}_i) = \sum_{k=1}^K \langle \Phi_k(\mathbf{x}_i), \mathbf{w}_k \rangle + b, \ \ \forall i = 1, \ldots, N \end{aligned} \tag{9}$$

In analogy to Bach et al. (2004) we treat problem (9) as a second order cone program (SOCP) leading to the following dual (see Appendix A for the derivation):

MKL Dual* for Generic Loss Functions

$$\begin{aligned} \min \quad & \gamma \\ \text{w.r.t.} \quad & \gamma \in \mathbb{R}, \ \alpha \in R^N \\ \text{s.t.} \quad & \sum_{i=1}^N \alpha_i = 0 \quad \text{and} \\ & \frac{1}{2}\left\| \sum_{i=1}^N \alpha_i \Phi_k(\mathbf{x}_i) \right\|_2^2 - \sum_{i=1}^N L(L'^{-1}(\alpha_i, y_i), y_i) + \sum_{i=1}^N \alpha_i L'^{-1}(\alpha_i, y_i) \leq \gamma, \ \forall k = 1, \ldots, K \end{aligned} \tag{10}$$

Here L'^{-1} denotes the inverse of the derivative of $L(f(\mathbf{x}), y)$ w.r.t. the prediction $f(\mathbf{x})$. To derive the SILP formulation we follow the same recipe as in Section 2.1: deriving the Lagrangian leads to a max-min problem formulation to be eventually reformulated as a SILP:

SILP for Generic Loss Functions

$$\begin{aligned} \max \quad & \theta \\ \text{w.r.t.} \quad & \theta \in \mathbb{R}, \beta \in \mathbb{R}^K \\ \text{s.t.} \quad & \mathbf{0} \le \beta, \quad \sum_{k=1}^{K} \beta_k = 1 \quad \text{and} \quad \sum_{k=1}^{K} \beta_k S_k(\alpha) \ge \theta, \ \forall \alpha \in \mathbb{R}^N, \ \sum_{i=1}^{N} \alpha_i = 0, \end{aligned} \tag{11}$$

where

$$S_k(\alpha) = -\sum_{i=1}^{N} L(L'^{-1}(\alpha_i, y_i), y_i) + \sum_{i=1}^{N} \alpha_i L'^{-1}(\alpha_i, y_i) + \frac{1}{2} \left\| \sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i) \right\|_2^2.$$

We assumed that $L(x,y)$ is strictly convex and differentiable in x. Unfortunately, the soft margin and ε-insensitive loss do not have these properties. We therefore consider them separately in the sequel.

Soft Margin Loss We use the following loss in order to approximate the soft margin loss:

$$L_\sigma(x,y) = \frac{C}{\sigma} \log(1 + \exp(\sigma(1 - xy))).$$

It is easy to verify that

$$\lim_{\sigma \to \infty} L_\sigma(x,y) = C(1-xy)_+.$$

Moreover, L_σ is strictly convex and differentiable for $\sigma < \infty$. Using this loss and assuming $y_i \in \{\pm 1\}$, we obtain (cf. Appendix B.3):

$$S_k(\alpha) = -\sum_{i=1}^{N} \frac{C}{\sigma} \left(\log\left(\frac{C y_i}{\alpha_i + C y_i} \right) + \log\left(-\frac{\alpha_i}{\alpha_i + C y_i} \right) \right) + \sum_{i=1}^{N} \alpha_i y_i + \frac{1}{2} \left\| \sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i) \right\|_2^2.$$

If $\sigma \to \infty$, then the first two terms vanish provided that $-C \le \alpha_i \le 0$ if $y_i = 1$ and $0 \le \alpha_i \le C$ if $y_i = -1$. Substituting $\alpha_i = -\tilde{\alpha}_i y_i$, we obtain

$$S_k(\tilde{\alpha}) = -\sum_{i=1}^{N} \tilde{\alpha}_i + \frac{1}{2} \left\| \sum_{i=1}^{N} \tilde{\alpha}_i y_i \Phi_k(\mathbf{x}_i) \right\|_2^2 \text{ and } \sum_{i=1}^{N} \tilde{\alpha}_i y_i = 0,$$

with $0 \le \tilde{\alpha}_i \le C$ ($i = 1, \ldots, N$) which is the same as (7).

One-Class Soft Margin Loss The one-class SVM soft margin (e.g. Schölkopf and Smola, 2002) is very similar to the two-class case and leads to

$$S_k(\alpha) = \frac{1}{2} \left\| \sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i) \right\|_2^2$$

subject to $\mathbf{0} \le \alpha \le \frac{1}{\nu N} \mathbf{1}$ and $\sum_{i=1}^{N} \alpha_i = 1$.

ε-insensitive Loss Using the same technique for the epsilon insensitive loss $L(x,y) = C(1-|x-y|)_+$, we obtain

$$\begin{aligned} S_k(\alpha,\alpha^*) &= \frac{1}{2}\left\|\sum_{i=1}^N(\alpha_i-\alpha_i^*)\Phi_k(\mathbf{x}_i)\right\|_2^2 - \sum_{i=1}^N(\alpha_i+\alpha_i^*)\varepsilon - \sum_{i=1}^N(\alpha_i-\alpha_i^*)y_i \\ \text{and} \quad & \sum_{i=1}^N(\alpha_i-\alpha_i^*)y_i = 0, \text{ with } \mathbf{0} \le \alpha,\alpha^* \le C\mathbf{1}. \end{aligned}$$

It is easy to derive the dual problem for other loss functions such as the quadratic loss or logistic loss (see Appendix B.1 & B.2). Note that the dual SILP's only differ in the definition of S_k and the domains of the α's.

2.3 Algorithms to Solve SILPs

All *semi-infinite linear programs* considered in this work have the following structure:

$$\begin{aligned} \max \quad & \theta \\ \text{w.r.t.} \quad & \theta \in \mathbb{R}, \beta \in \mathbb{R}^K \\ \text{s.t.} \quad & \mathbf{0} \le \beta, \quad \sum_{k=1}^K \beta_k = 1 \quad \text{and} \quad \sum_{k=1}^K \beta_k S_k(\alpha) \ge \theta \text{ for all } \alpha \in \mathcal{C}. \end{aligned} \tag{12}$$

They have to be optimized with respect to β and θ. The constraints depend on definition of S_k and therefore on the choice of the cost function. Using Theorem 5 in Rätsch et al. (2002) one can show that the above SILP has a solution if the corresponding primal is feasible and bounded (see also Hettich and Kortanek, 1993). Moreover, there is no duality gap, if $\mathcal{M} = \text{co}\{[S_1(\alpha),\ldots,S_K(\alpha)]^\top \mid \alpha \in \mathcal{C}\}$ is a closed set. For all loss functions considered in this paper this condition is satisfied.

We propose to use a technique called *Column Generation* to solve (12). The basic idea is to compute the optimal (β,θ) in (12) for a restricted subset of constraints. It is called the *restricted master problem*. Then a second algorithm generates a new, yet unsatisfied constraint determined by α. In the best case the other algorithm finds the constraint that maximizes the constraint violation for the given intermediate solution (β,θ), i.e.

$$\alpha_\beta := \operatorname*{argmin}_{\alpha\in\mathcal{C}} \sum_k \beta_k S_k(\alpha). \tag{13}$$

If α_β satisfies the constraint $\sum_{k=1}^K \beta_k S_k(\alpha_\beta) \ge \theta$, then the solution (θ,β) is optimal. Otherwise, the constraint is added to the set of constraints and the iterations continue.

Algorithm 1 is a special case of a set of SILP algorithms known as *exchange methods*. These methods are known to converge (cf. Theorem 7.2 in Hettich and Kortanek, 1993). However, no convergence rates for such algorithm are known.[4]

Since it is often sufficient to obtain an approximate solution, we have to define a suitable convergence criterion. Note that the problem is solved when all constraints are satisfied. Hence, it is a

4. It has been shown that solving semi-infinite problems like (7), using a method related to boosting (e.g. Meir and Rätsch, 2003) one requires at most $T = o(\log(M)/\hat{\varepsilon}^2)$ iterations, where $\hat{\varepsilon}$ is the remaining constraint violation and the constants may depend on the kernels and the number of examples N (Rätsch, 2001; Rätsch and Warmuth, 2005; Warmuth et al., 2006). At least for not too small values of $\hat{\varepsilon}$ this technique produces reasonably fast good approximate solutions.

natural choice to use the normalized maximal constraint violation as a convergence criterion, i.e. the algorithm stops if $\varepsilon_{MKL} \geq \varepsilon_{MKL}^t := \left|1 - \frac{\sum_{k=1}^K \beta_k^t S_k(\alpha^t)}{\theta^t}\right|$, where ε_{MKL} is an accuracy parameter, (β^t, θ^t) is the optimal solution at iteration $t-1$ and α^t corresponds to the newly found maximally violating constraint of the next iteration.

In the following we will formulate algorithms that alternately optimize the parameters α and β.

2.3.1 A WRAPPER ALGORITHM

The wrapper algorithm (see Algorithm 1) divides the problem into an inner and an outer subproblem. The solution is obtained by alternatively solving the outer problem using the results of the inner problem as input and vice versa until convergence. The outer loop constitutes the *restricted master problem* which determines the optimal β for a fixed α using an of-the-shelf linear optimizer. In the inner loop one has to identify unsatisfied constraints, which, fortunately, turns out to be particularly simple. Note that (13) is for all considered cases exactly the dual optimization problem of the single kernel case for fixed β. For instance for binary classification with soft-margin loss, (13) reduces to the standard SVM dual using the kernel $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \sum_k \beta_k \mathbf{k}_k(\mathbf{x}_i, \mathbf{x}_j)$:

$$\begin{aligned} v = \quad & \min_{\alpha \in \mathbb{R}^N} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^N \alpha_i \\ & \text{s.t.} \qquad \mathbf{0} \leq \alpha \leq C\mathbf{1} \text{ and } \sum_{i=1}^N \alpha_i y_i = 0. \end{aligned}$$

Hence, we can use a standard SVM implementation with a single kernel in order to identify the most violated constraint $v \leq \theta$. Since there exists a large number of efficient algorithms to solve the single kernel problems for all sorts of cost functions, we have therefore found an easy way to extend their applicability to the problem of Multiple Kernel Learning. Also, if the kernels are computed on-the-fly within the SVM still only a single kernel cache is required. The wrapper algorithm is very easy to implement, very generic and already reasonably fast for small to medium size problems. However, determining α up to a fixed high precision even for intermediate solutions, while β is still far away from the global optimal is unnecessarily costly. Thus there is room for improvement motivating the next section.

2.3.2 A CHUNKING ALGORITHM FOR SIMULTANEOUS OPTIMIZATION OF α AND β

The goal is to simultaneously optimize α and β in SVM training. Usually it is infeasible to use standard optimization tools (e.g. MINOS, CPLEX, LOQO) for solving even the *SVM training* problems on data sets containing more than a few thousand examples. So-called decomposition techniques as chunking (e.g. used in Joachims, 1998) overcome this limitation by exploiting the special structure of the SVM problem. The key idea of decomposition is to freeze all but a small number of optimization variables (*working set*) and to solve a sequence of constant-size problems (subproblems of the SVM dual).

Here we would like to propose an extension of the chunking algorithm to optimize the kernel weights β and the example weights α at the same time. The algorithm is motivated from an insufficiency of the wrapper algorithm described in the previous section: If the β's are not optimal yet, then the optimization of the α's until optimality is not necessary and therefore inefficient. It would

Algorithm 1 The MKL-wrapper algorithm optimizes a convex combination of K kernels and employs a linear programming solver to iteratively solve the semi-infinite linear optimization problem (12). The accuracy parameter ε_{MKL} is a parameter of the algorithm. $S_k(\alpha)$ and C are determined by the cost function.

$S^0 = 1$, $\theta^1 = -\infty$, $\beta_k^1 = \frac{1}{K}$ for $k = 1,\ldots,K$
for $t = 1,2,\ldots$ **do**
  Compute $\alpha^t = \operatorname{argmin}_{\alpha \in C} \sum_{k=1}^{K} \beta_k^t S_k(\alpha)$ by single kernel algorithm with $\mathbf{k} = \sum_{k=1}^{K} \beta_k^t \mathbf{k}_k$
  $S^t = \sum_{k=1}^{K} \beta_k^t S_k^t$, where $S_k^t = S_k(\alpha^t)$
  if $\left|1 - \frac{S^t}{\theta^t}\right| \leq \varepsilon_{MKL}$ **then break**
  $(\beta^{t+1}, \theta^{t+1}) = \operatorname{argmax}\ \theta$
    w.r.t. $\beta \in \mathbb{R}^K, \theta \in \mathbb{R}$
    s.t. $\mathbf{0} \leq \beta,\ \sum_{k=1}^{K} \beta_k = 1$ and $\sum_{k=1}^{K} \beta_k S_k^r \geq \theta$ for $r = 1,\ldots,t$
end for

be considerably faster if for any newly obtained α in the chunking iterations, we could efficiently recompute the optimal β and then continue optimizing the α's using the new kernel weighting.

Intermediate Recomputation of β Recomputing β involves solving a linear program and the problem grows with each additional α-induced constraint. Hence, after many iterations solving the LP may become infeasible. Fortunately, there are two facts making it still possible: (a) only a small number of the added constraints remain active and one may as well remove inactive ones — this prevents the LP from growing arbitrarily and (b) for Simplex-based LP optimizers such as CPLEX there exists the so-called *hot-start feature* which allows one to efficiently recompute the new solution, if for instance only a few additional constraints are added.

The SVM$^{\text{light}}$ optimizer which we are going to modify, internally needs the output

$$\hat{g}_i = \sum_{j=1}^{N} \alpha_j y_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$$

for all training examples $i = 1,\ldots,N$ in order to select the next variables for optimization (Joachims, 1999). However, if one changes the kernel weights, then the stored $\hat{g}_i$ values become invalid and need to be recomputed. In order to avoid the full recomputation one has to additionally store a $K \times N$ matrix $g_{k,i} = \sum_{j=1}^{N} \alpha_j y_j \mathbf{k}_k(\mathbf{x}_i, \mathbf{x}_j)$, i.e. the outputs for each kernel separately. If the β's change, then $\hat{g}_i$ can be quite efficiently recomputed by $\hat{g}_i = \sum_k \beta_k g_{k,i}$. We implemented the final chunking algorithm for the MKL regression and classification case and display the latter in Algorithm 2.

2.3.3 DISCUSSION

The Wrapper as well as the chunking algorithm have both their merits: The Wrapper algorithm only relies on the repeated efficient computation of the single kernel solution, for which typically large scale algorithms exist. The chunking algorithm is faster, since it exploits the intermediate α's – however, it needs to compute and cache the K kernels separately (particularly important when

Algorithm 2 Outline of the MKL-chunking algorithm for the classification case (extension to SVM$^{\text{light}}$) that optimizes α and the kernel weighting β simultaneously. The accuracy parameter ε_{MKL} and the subproblem size Q are assumed to be given to the algorithm. For simplicity we omit the removal of inactive constraints. Also note that from one iteration to the next the LP only differs by one additional constraint. This can usually be exploited to save computing time for solving the LP.

$g_{k,i} = 0$, $\hat{g}_i = 0$, $\alpha_i = 0$, $\beta_k^1 = \frac{1}{K}$ for $k = 1,\ldots,K$ and $i = 1,\ldots,N$
for $t = 1,2,\ldots$ **do**
  Check optimality conditions and stop if optimal
  select Q suboptimal variables $i_1,\ldots,i_Q$ based on $\hat{\mathbf{g}}$ and α
  $\alpha^{old} = \alpha$
  solve SVM dual with respect to the selected variables and update α
  $g_{k,i} = g_{k,i} + \sum_{q=1}^{Q} (\alpha_{i_q} - \alpha_{i_q}^{old}) y_{i_q} \mathbf{k}_k(\mathbf{x}_{i_q}, \mathbf{x}_i)$ for all $k = 1,\ldots,M$ and $i = 1,\ldots,N$
  for $k = 1,\ldots,K$ **do**
    $S_k^t = \frac{1}{2} \sum_r g_{k,r} \alpha_r^t y_r - \sum_r \alpha_r^t$
  end for
  $S^t = \sum_{k=1}^{K} \beta_k^t S_k^t$
  if $\left| 1 - \frac{S^t}{\theta^t} \right| \geq \varepsilon_{MKL}$
    $(\beta^{t+1}, \theta^{t+1}) = \text{argmax } \theta$
      w.r.t. $\beta \in \mathbb{R}^K, \theta \in \mathbb{R}$
      s.t. $\mathbf{0} \leq \beta, \quad \sum_k \beta_k = 1$ and $\sum_{k=1}^{M} \beta_k S_k^r \geq \theta$ for $r = 1,\ldots,t$
  else
    $\theta^{t+1} = \theta^t$
  end if
  $\hat{g}_i = \sum_k \beta_k^{t+1} g_{k,i}$ for all $i = 1,\ldots,N$
end for

N is large). If, on the other hand, K is large, then the amount of memory available for caching is drastically reduced and, hence, kernel caching is not effective anymore. The same statements also apply to the SMO-like MKL algorithm proposed in Bach et al. (2004). In this case one is left with the Wrapper algorithm, unless one is able to exploit properties of the particular problem or the sub-kernels (see next section).

3. Sparse Feature Maps and Parallel Computations

In this section we discuss two strategies to accelerate SVM training. First we consider the case where the explicit mapping Φ into the kernel feature space is known as well as sparse. For this case we show that MKL training (and also SVM training in general) can be made drastically faster, in particular, when N and K are large. In the second part we discuss a simple, yet efficient way to parallelize MKL as well as SVM training.

3.1 Explicit Computations with Sparse Feature Maps

We assume that all K sub-kernels are given as

$$\mathbf{k}_k(\mathbf{x}, \mathbf{x}') = \langle \Phi_k(\mathbf{x}), \Phi_k(\mathbf{x}') \rangle$$

and the mappings Φ_k are given explicitly ($k = 1,\ldots,K$). Moreover, we suppose that the mapped examples $\Phi_k(\mathbf{x})$ are very sparse. We start by giving examples of such kernels and discuss two kernels that are often used in biological sequence analysis (Section 3.1.1). In Section 3.1.2 we discuss several strategies for efficiently storing and computing with high dimensional sparse vectors (in particular for these two kernels). Finally in Section 3.1.3 we discuss how we can exploit these properties to accelerate chunking algorithms, such as SVM$^{\text{light}}$, by a factor of up to Q (the chunking subproblem size).

3.1.1 STRING KERNELS

The Spectrum Kernel The spectrum kernel (Leslie et al., 2002) implements the n-gram or bag-of-words kernel (Joachims, 1998) as originally defined for text classification in the context of biological sequence analysis. The idea is to count how often a d-mer (a contiguous string of length d) is contained in the sequences $\mathbf{x}$ and $\mathbf{x}'$. Summing up the product of these counts for every possible d-mer (note that there are exponentially many) gives rise to the kernel value which formally is defined as follows: Let Σ be an alphabet and $u \in \Sigma^d$ a d-mer and $\#u(\mathbf{x})$ the number of occurrences of u in $\mathbf{x}$. Then the spectrum kernel is defined as the inner product of $\mathbf{k}(\mathbf{x},\mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$, where $\Phi(\mathbf{x}) = (\#u(\mathbf{x}))_{u\in\Sigma^d}$. Note that spectrum-like kernels cannot extract any positional information from the sequence which goes beyond the d-mer length. It is well suited for describing the content of a sequence but is less suitable for instance for analyzing signals where motifs may appear in a certain order or at specific positions. Also note that spectrum-like kernels are capable of dealing with sequences with varying length.

The spectrum kernel can be efficiently computed in $O(d(|\mathbf{x}| + |\mathbf{x}'|))$ using tries (Leslie et al., 2002), where $|\mathbf{x}|$ denotes the length of sequence $\mathbf{x}$. An easier way to compute the kernel for two sequences $\mathbf{x}$ and $\mathbf{x}'$ is to separately extract and sort the N d-mers in each sequence, which can be done in a preprocessing step. Note that for instance DNA d-mers of length $d \leq 16$ can be efficiently represented as a 32-bit integer value. Then one iterates over all d-mers of sequences $\mathbf{x}$ and $\mathbf{x}'$ simultaneously and counts which d-mers appear in both sequences and sums up the product of their counts. The computational complexity of the kernel computation is $O(\log(|\Sigma|)d(|\mathbf{x}| + |\mathbf{x}'|))$.

The Weighted Degree Kernel The so-called *weighted degree* (WD) kernel (Rätsch and Sonnenburg, 2004) efficiently computes similarities between sequences while taking positional information of k-mers into account. The main idea of the WD kernel is to count the (exact) co-occurrences of k-mers at corresponding positions in the two sequences to be compared. The *WD kernel of order d* compares two sequences $\mathbf{x}_i$ and $\mathbf{x}_j$ of length L by summing all contributions of k-mer matches of lengths $k \in \{1,\ldots,d\}$, weighted by coefficients β_k:

$$\mathbf{k}(\mathbf{x}_i,\mathbf{x}_j) = \sum_{k=1}^{d} \beta_k \sum_{l=1}^{L-k+1} \mathrm{I}(u_{k,l}(\mathbf{x}_i) = u_{k,l}(\mathbf{x}_j)). \tag{14}$$

Here, $u_{k,l}(\mathbf{x})$ is the string of length k starting at position l of the sequence $\mathbf{x}$ and $\mathrm{I}(\cdot)$ is the indicator function which evaluates to 1 when its argument is *true* and to 0 otherwise. For the weighting coefficients, Rätsch and Sonnenburg (2004) proposed to use $\beta_k = 2\frac{d-k+1}{d(d+1)}$. Matching substrings are thus rewarded with a score depending on the length of the substring.[5]

5. Note that although in our case $\beta_{k+1} < \beta_k$, longer matches nevertheless contribute more strongly than shorter ones: this is due to the fact that each long match also implies several short matches, adding to the value of (14). Exploiting this

Note that the WD kernel can be understood as a Spectrum kernel where the k-mers starting at different positions are treated independently of each other.[6] Moreover, it does not only consider substrings of length exactly d, but also all shorter matches. Hence, the feature space for each position has $\sum_{k=1}^{d}|\Sigma|^k = \frac{|\Sigma|^{d+1}-1}{|\Sigma|-1} - 1$ dimensions and is additionally duplicated L times (leading to $O(L|\Sigma|^d)$ dimensions). However, the computational complexity of the WD kernel is in the worst case $O(dL)$ as can be directly seen from (14).

3.1.2 EFFICIENT STORAGE OF SPARSE WEIGHTS

The considered string kernels correspond to a feature space that can be huge. For instance in the case of the WD kernel on DNA sequences of length 100 with $K = 20$, the corresponding feature space is 10^{14} dimensional. However, most dimensions in the feature space are not used since only a few of the many different k-mers actually appear in the sequences. In this section we briefly discuss three methods to efficiently deal with sparse vectors v. We assume that the elements of the vector v are indexed by some index set $\mathcal{U}$ (for sequences, e.g. $\mathcal{U} = \Sigma^d$) and that we only need three operations: `clear`, `add` and `lookup`. The first operation sets the vector v to zero, the `add` operation increases the weight of a dimension for an element $u \in \mathcal{U}$ by some amount α, i.e. $v_u = v_u + \alpha$ and `lookup` requests the value v_u. The latter two operations need to be performed as quickly as possible (whereas the performance of the `lookup` operation is of higher importance).

Explicit Map If the dimensionality of the feature space is small enough, then one might consider keeping the whole vector v in memory and to perform direct operations on its elements. Then each read or write operation is $O(1)$.[7] This approach has expensive memory requirements ($O(|\Sigma|^d)$), but is very fast and best suited for instance for the Spectrum kernel on DNA sequences with $d \leq 14$ and on protein sequences with $d \leq 6$.

Sorted Arrays More memory efficient but computationally more expensive are sorted arrays of index-value pairs (u, v_u). Assuming the L indexes are given and sorted in advance, one can efficiently change or look up a single v_u for a corresponding u by employing a binary search procedure ($O(\log(L))$). When given L' look up indexes at once, one may sort them in advance and then simultaneously traverse the two arrays in order to determine which elements appear in the first array (i.e. $O(L+L')$ operations – omitting the sorting of the second array – instead of $O(\log(L)L')$). This method is well suited for cases where L and L' are of comparable size, as for instance for computations of single Spectrum kernel elements (as proposed in Leslie et al., 2004). If, $L \gg L'$, then the binary search procedure should be preferred.

Tries Another way of organizing the non-zero elements are *tries* (Fredkin, 1960): The idea is to use a tree with at most $|\Sigma|$ siblings of depth d. The leaves store a single value: the element v_u, where $u \in \Sigma^d$ is a d-mer and the path to the leaf corresponds to u.

knowledge allows for a $O(L)$ reformulation of the kernel using "block-weights" as has been done in Sonnenburg et al. (2005b).

6. It therefore is very position dependent and does not tolerate any positional "shift". For that reason we proposed in Rätsch et al. (2005) a WD kernel *with shifts*, which tolerates a small number of shifts, that lies in between the WD and the Spectrum kernel.
7. More precisely, it is $\log d$, but for small enough d (which we have to assume anyway) the computational effort is exactly one memory access.

To `add` or `lookup` an element one only needs d operations to reach a leaf of the tree (and to create necessary nodes on the way in an `add` operation). Note that the worst-case computational complexity of the operations is independent of the number of d-mers/elements stored in the tree.

While tries are not faster than *sorted arrays* in `lookup` and need considerably more storage (e.g. for pointers to its parent and siblings), they are useful for the previously discussed WD kernel. Here we not only have to lookup one substring $u \in \Sigma^d$, but also all prefixes of u. For *sorted arrays* this amounts to d separate `lookup` operations, while for tries all prefixes of u are already known when the bottom of the tree is reached. In this case the trie has to store weights also on the internal nodes. This is illustrated for the WD kernel in Figure 1.

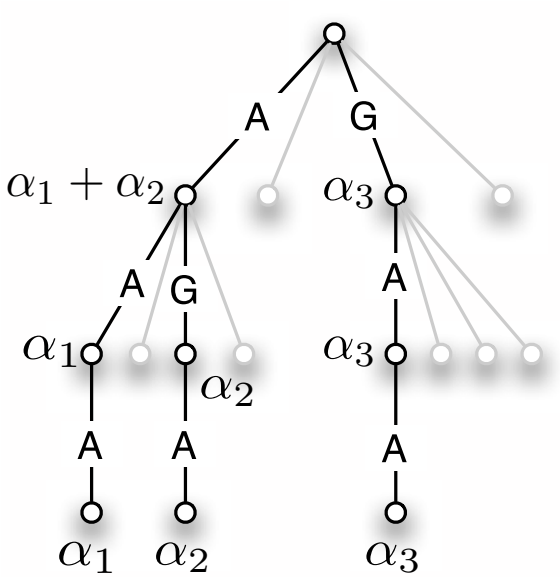


Figure 1: Three sequences AAA, AGA, GAA with weights α_1, α_2 & α_3 are added to the trie. The figure displays the resulting weights at the nodes.

3.1.3 SPEEDING UP SVM TRAINING

As it is not feasible to use standard optimization toolboxes for solving large scale SVM training problem, decomposition techniques are used in practice. Most chunking algorithms work by first selecting Q variables (working set $W \subseteq \{1,\ldots,N\}$, $Q := |W|$) based on the current solution and then solve the reduced problem with respect to the working set variables. These two steps are repeated until some optimality conditions are satisfied (see e.g. Joachims (1998)). For selecting the working set and checking the termination criteria in each iteration, the vector $\mathbf{g}$ with $g_i = \sum_{j=1}^{N} \alpha_j y_j \mathbf{k}(x_i, x_j), \quad i = 1,\ldots,N$ is usually needed. Computing $\mathbf{g}$ from scratch in every iteration which would require $O(N^2)$ kernel computations. To avoid recomputation of $\mathbf{g}$ one typically starts with $\mathbf{g} = \mathbf{0}$ and only computes updates of $\mathbf{g}$ on the working set W

$$g_i \leftarrow g_i^{old} + \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \mathbf{k}(x_i, x_j), \quad \forall i = 1,\ldots,N.$$

As a result the effort decreases to $o(QN)$ kernel computations, which can be further speed up by using kernel caching (e.g. Joachims, 1998). However kernel caching is not efficient enough for large scale problems[8] and thus most time is spend computing kernel rows for the updates of $\mathbf{g}$ on the working set W. Note however that this update as well as computing the Q kernel rows can be easily parallelized; cf. Section 4.2.1.

Exploiting $\mathbf{k}(\mathbf{x}_i,\mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ and $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$ we can rewrite the update rule as

$$g_i \leftarrow g_i^{old} + \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle = g_i^{old} + \langle \mathbf{w}^W, \Phi(\mathbf{x}_i) \rangle, \tag{15}$$

where $\mathbf{w}^W = \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \Phi(\mathbf{x}_j)$ is the normal (update) vector on the working set.

If the kernel feature map can be computed explicitly and is sparse (as discussed before), then computing the update in (15) can be accelerated. One only needs to compute and store $\mathbf{w}^W$ (using the `clear` and $\sum_{q \in W} |\{\Phi_j(\mathbf{x}_q) \neq 0\}|$ `add` operations) and performing the scalar product $\langle \mathbf{w}^W, \Phi(\mathbf{x}_i) \rangle$ (using $|\{\Phi_j(\mathbf{x}_i) \neq 0\}|$ `lookup` operations).

Depending on the kernel, the way the sparse vectors are stored Section 3.1.2 and on the sparseness of the feature vectors, the speedup can be quite drastic. For instance for the WD kernel one kernel computation requires $o(Ld)$ operations (L is the length of the sequence). Hence, computing (15) N times requires $O(NQLd)$ operations. When using tries, then one needs QL `add` operations (each $o(d)$) and NL `lookup` operations (each $o(d)$). Therefore only $o(QLd + NLd)$ basic operations are needed in total. When N is large enough it leads to a speedup by a factor of Q. Finally note that kernel caching is no longer required and as Q is small in practice (e.g. $Q = 42$) the resulting trie has rather few leaves and thus only needs little storage.

The pseudo-code of our `linadd` SVM chunking algorithm is given in Algorithm 3.

Algorithm 3 Outline of the chunking algorithm that exploits the fast computations of linear combinations of kernels (e.g. by tries).

{INITIALIZATION}
$g_i = 0$, $\alpha_i = 0$ for $i = 1,\ldots,N$
{LOOP UNTIL CONVERGENCE}
for $t = 1,2,\ldots$ **do**
  Check optimality conditions and stop if optimal
  select working set W based on $\mathbf{g}$ and α, store $\alpha^{old} = \alpha$
  solve reduced problem W and update α

  `clear` $\mathbf{w}$
  $\mathbf{w} \leftarrow \mathbf{w} + (\alpha_j - \alpha_j^{old}) y_j \Phi(\mathbf{x}_j)$ for all $j \in W$ (using `add`)
  update $g_i = g_i + \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle$ for all $i = 1,\ldots,N$ (using `lookup`)
end for

MKL Case As elaborated in Section 2.3.2 and Algorithm 2, for MKL one stores K vectors $\mathbf{g}_k, \quad k = 1,\ldots,K$: one for each kernel in order to avoid full recomputation of $\hat{\mathbf{g}}$ if a kernel weight β_k is updated. Thus to use the idea above in Algorithm 2 all one has to do is to store K normal vectors

8. For instance when using a million examples one can only fit 268 rows into 1 GB. Moreover, caching 268 rows is insufficient when for instance having many thousands of active variables.

(e.g. tries)

$$\mathbf{w}_k^W = \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \Phi_k(\mathbf{x}_j), \quad k = 1, \ldots, K$$

which are then used to update the $K \times N$ matrix $g_{k,i} = g_{k,i}^{old} + \langle \mathbf{w}_k^W, \Phi_k(\mathbf{x}_i) \rangle$ (for all $k = 1 \ldots K$ and $i = 1 \ldots N$) by which $\hat{g}_i = \sum_k \beta_k g_{k,i}$, (for all $i = 1 \ldots N$) is computed.

3.2 A Simple Parallel Chunking Algorithm

As still most time is spent in evaluating $g(\mathbf{x})$ for all training examples further speedups are gained when parallelizing the evaluation of $g(\mathbf{x})$. When using the `linadd` algorithm, one first constructs the trie (or any of the other possible more appropriate data structures) and then performs parallel `lookup` operations using several CPUs (e.g. using shared memory or several copies of the data structure on separate computing nodes). We have implemented this algorithm based on multiple *threads* (using shared memory) and gain reasonable speedups (see next section).

Note that this part of the computations is almost ideal to distribute to many CPUs, as only the updated α (or $\mathbf{w}$ depending on the communication costs and size) have to be transfered before each CPU computes a large chunk $I_k \subset \{1, \ldots, N\}$ of

$$h_i^{(k)} = \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle, \quad \forall i \in I_k, \quad \forall k = 1, \ldots, N, \text{ where } (I_1 \cup \cdots \cup I_n) = (1, \ldots, N)$$

which is transfered to a master node that finally computes $\mathbf{g} \leftarrow \mathbf{g} + \mathbf{h}$, as illustrated in Algorithm 4.

4. Results and Discussion

In the following subsections we will first apply multiple kernel learning to knowledge discovery tasks, demonstrating that it can be used for automated model selection and to interpret the learned model (Section 4.1), followed by a benchmark comparing the running times of SVMs and MKL using any of the proposed algorithmic optimizations (Section 4.2).

4.1 MKL for Knowledge Discovery

In this section we will discuss toy examples for binary classification and regression, showing that MKL can recover information about the problem at hand, followed by a brief review on problems for which MKL has been successfully used.

4.1.1 CLASSIFICATION

The first example we deal with is a binary classification problem. The task is to separate two concentric classes shaped like the outline of stars. By varying the distance between the boundary of the stars we can control the separability of the problem. Starting with a non-separable scenario with zero distance, the data quickly becomes separable as the distance between the stars increases, and the boundary needed for separation will gradually tend towards a circle. In Figure 2 three scatter plots of data sets with varied separation distances are displayed.

We generate several training and test sets for a wide range of distances (the radius of the inner star is fixed at 4.0, the outer stars radius is varied from $4.1 \ldots 9.9$). Each data set contains 2,000 observations (1,000 positive and 1,000 negative) using a moderate noise level (Gaussian noise with zero mean and standard deviation 0.3). The MKL-SVM was trained for different values of the

Algorithm 4 Outline of the parallel chunking algorithm that exploits the fast computations of linear combinations of kernels.

{ **Master node** }
{INITIALIZATION}
$g_i = 0$, $\alpha_i = 0$ for $i = 1, \ldots, N$
{LOOP UNTIL CONVERGENCE}
for $t = 1, 2, \ldots$ **do**
  Check optimality conditions and stop if optimal
  select working set W based on $\mathbf{g}$ and α, store $\alpha^{old} = \alpha$
  solve reduced problem W and update α
  transfer to Slave nodes: $\alpha_j - \alpha_j^{old}$ for all $j \in W$
  fetch from n Slave nodes: $\mathbf{h} = (\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(n)})$
  update $g_i = g_i + h_i$ for all $i = 1, \ldots, N$
end for
signal convergence to slave nodes

{ **Slave nodes** }
{LOOP UNTIL CONVERGENCE}
while not converged **do**
  fetch from Master node $\alpha_j - \alpha_j^{old}$ for all $j \in W$
  `clear` **w**
  $\mathbf{w} \leftarrow \mathbf{w} + (\alpha_j - \alpha_j^{old}) y_j \Phi(\mathbf{x}_j)$ for all $j \in W$ (using `add`)
  node k computes $h_i^{(k)} = \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle$
    for all $i = (k-1)\frac{N}{n}, \ldots, k\frac{N}{n} - 1$ (using `lookup`)
  transfer to master: $\mathbf{h}^{(k)}$
end while

regularization parameter C, where we set $\varepsilon_{MKL} = 10^{-3}$. For every value of C we averaged the test errors of all setups and choose the value of C that led to the smallest overall error ($C = 0.5$).[9]

The choice of the kernel width of the Gaussian RBF (below, denoted by RBF) kernel used for classification is expected to depend on the separation distance of the learning problem: An increased distance between the stars will correspond to a larger optimal kernel width. This effect should be visible in the results of the MKL, where we used MKL-SVMs with five RBF kernels with different widths ($2\sigma^2 \in \{0.01, 0.1, 1, 10, 100\}$). In Figure 2 we show the obtained kernel weightings for the five kernels and the test error (circled line) which quickly drops to zero as the problem becomes separable. Every column shows one MKL-SVM weighting. The courses of the kernel weightings reflect the development of the learning problem: as long as the problem is difficult the best separation can be obtained when using the kernel with smallest width. The low width kernel looses importance when the distance between the stars increases and larger kernel widths obtain a larger weight in MKL. Increasing the distance between the stars, kernels with greater widths are used. Note that the RBF kernel with largest width was not appropriate and thus never chosen. This illustrates that MKL can indeed recover information about the structure of the learning problem.

9. Note that we are aware of the fact that the test error might be slightly underestimated.

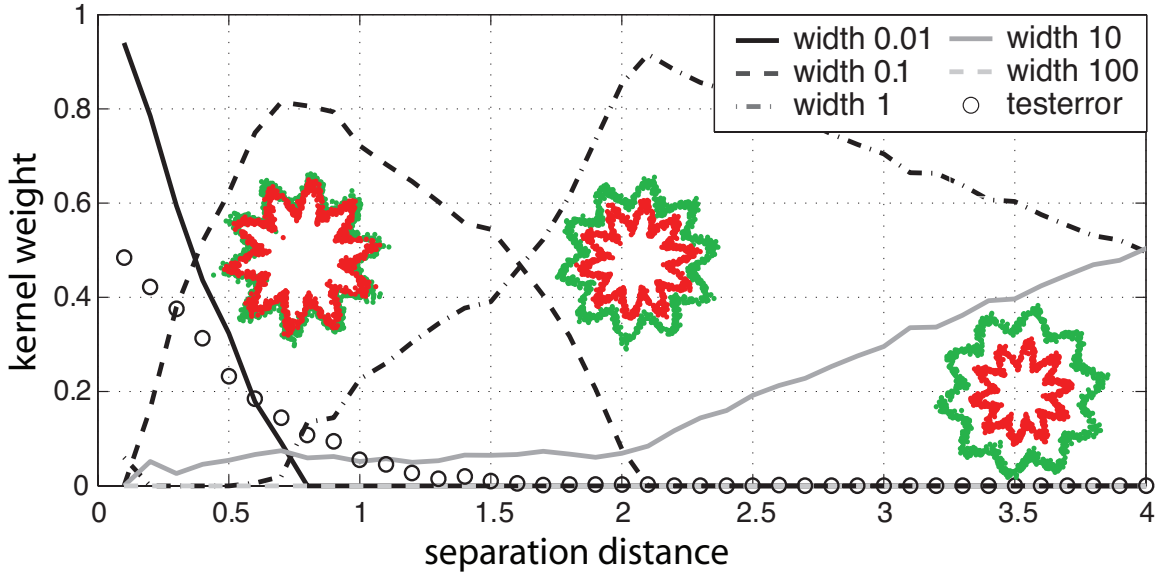


Figure 2: A 2-class toy problem where the dark gray (or green) star-like shape is to be distinguished from the light gray (or red) star inside of the dark gray star. The distance between the dark star-like shape and the light star increases from the left to the right.

4.1.2 REGRESSION

We applied the newly derived MKL support vector regression formulation to the task of learning a sine function using three RBF-kernels with different widths ($2\sigma^2 \in \{0.005, 0.05, 0.5, 1, 10\}$). To this end, we generated several data sets with increasing frequency of the sine wave. The sample size was chosen to be 1,000. Analogous to the procedure described above we choose the value of $C = 10$, minimizing the overall test error. In Figure 3 exemplarily three sine waves are depicted, where the frequency increases from left to right. For every frequency the computed weights for each kernel width are shown. One can see that MKL-SV regression switches to the width of the RBF-kernel fitting the regression problem best.

In another regression experiment, we combined a linear function with two sine waves, one of lower frequency and one of high frequency, i.e. $f(x) = \sin(ax) + \sin(bx) + cx$. Furthermore we increase the frequency of the higher frequency sine wave, i.e. we varied a leaving b and c unchanged. The MKL weighting should show a combination of different kernels. Using ten RBF-kernels of different width (see Figure 4) we trained a MKL-SVR and display the learned weights (a column in the figure). Again the sample size is 1,000 and one value for $C = 5$ is chosen via a previous experiment ($\varepsilon_{MKL} = 10^{-5}$). The largest selected width (100) models the linear component (since RBF kernels with large widths are effectively linear) and the medium width (1) corresponds to the lower frequency sine. We varied the frequency of the high frequency sine wave from low to high (left to right in the figure). One observes that MKL determines an appropriate combination of kernels of low and high widths, while decreasing the RBF kernel width with increased frequency.

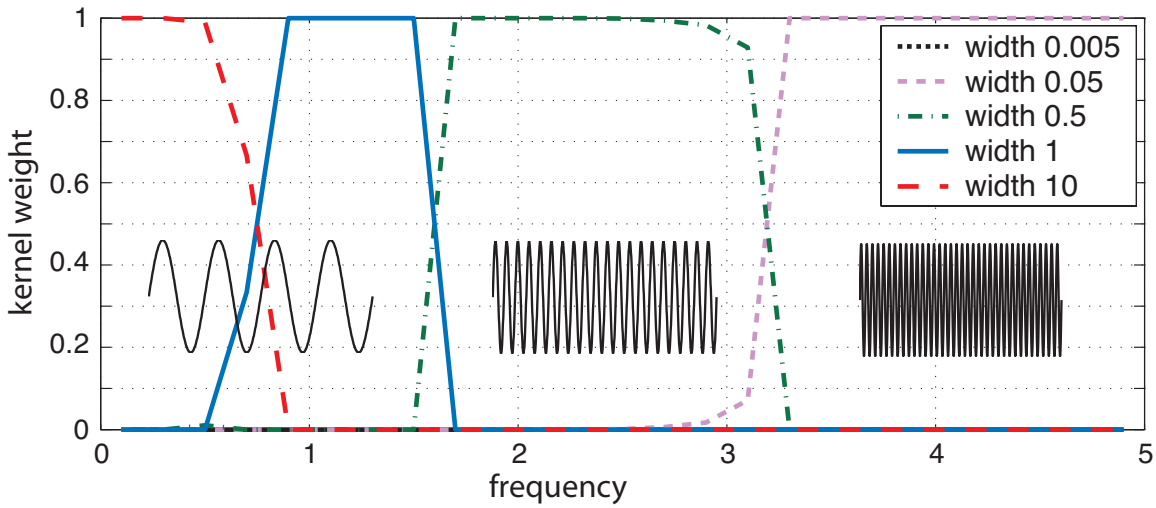


Figure 3: MKL-Support Vector Regression for the task of learning a sine wave (please see text for details).

Additionally one can observe that MKL leads to sparse solutions since most of the kernel weights in Figure 4 are depicted in blue, that is they are zero.[10]

4.1.3 REAL WORLD APPLICATIONS IN BIOINFORMATICS

MKL has been successfully used on real-world data sets in the field of computational biology (Lanckriet et al., 2004; Sonnenburg et al., 2005a). It was shown to improve classification performance on the task of ribosomal and membrane protein prediction (Lanckriet et al., 2004), where a weighting over different kernels each corresponding to a different feature set was learned. In their result, the included random channels obtained low kernel weights. However, as the data sets was rather small ($\approx 1,000$ examples) the kernel matrices could be precomputed and simultaneously kept in memory, which was not possible in Sonnenburg et al. (2005a), where a splice site recognition task for the worm *C. elegans* was considered. Here data is available in abundance (up to one million examples) and larger amounts are indeed needed to obtain state of the art results (Sonnenburg et al., 2005b).[11] On that data set we were able to solve the classification MKL SILP for $N = 1,000,000$ examples and $K = 20$ kernels, as well as for $N = 10,000$ examples and $K = 550$ kernels, using the `linadd` optimizations with the weighted degree kernel. As a result we a) were able to learn the weighting β instead of choosing a heuristic and b) were able to use MKL as a tool for interpreting the SVM classifier as in Sonnenburg et al. (2005a); Rätsch et al. (2005).

As an example we learned the weighting of a WD kernel of degree 20, which consist of a weighted sum of 20 sub-kernels each counting matching d-mers, for $d = 1,\ldots,20$. The learned

10. The training time for MKL-SVR in this setup but with 10,000 examples was about 40 minutes, when kernel caches of size 100MB are used.
11. In Section 4.2 we will use a *human* splice data set containing 15 million examples, and train WD kernel based SVM classifiers on up to 10 million examples using the parallelized `linadd` algorithm.

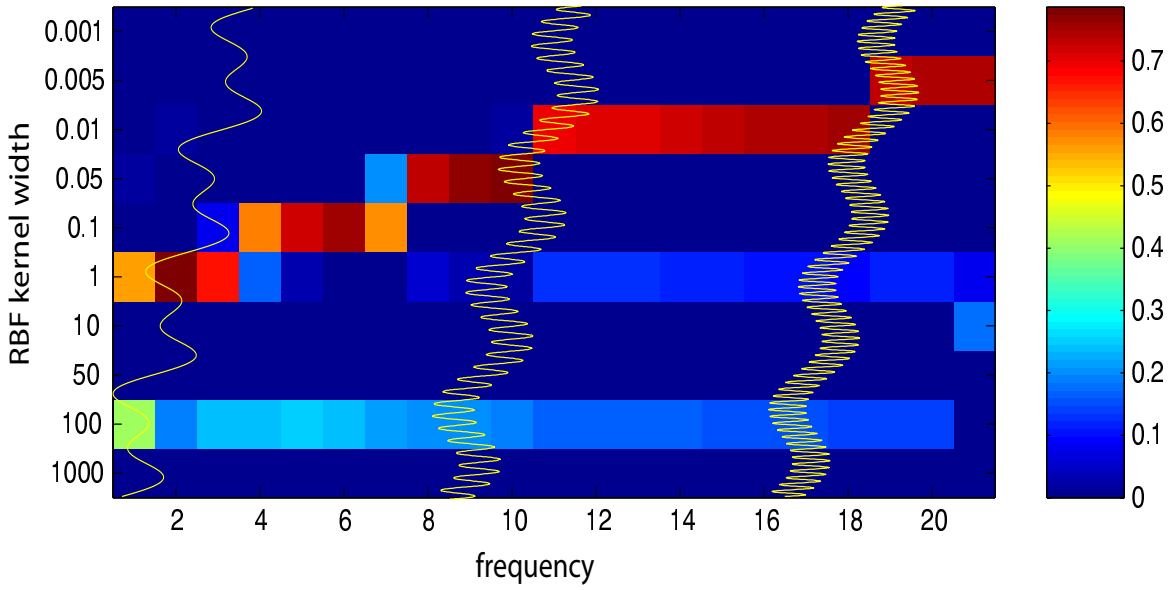


Figure 4: MKL support vector regression on a linear combination of three functions: $f(x) = \sin(ax) + \sin(bx) + cx$. MKL recovers that the original function is a combination of functions of low and high complexity. For more details see text.

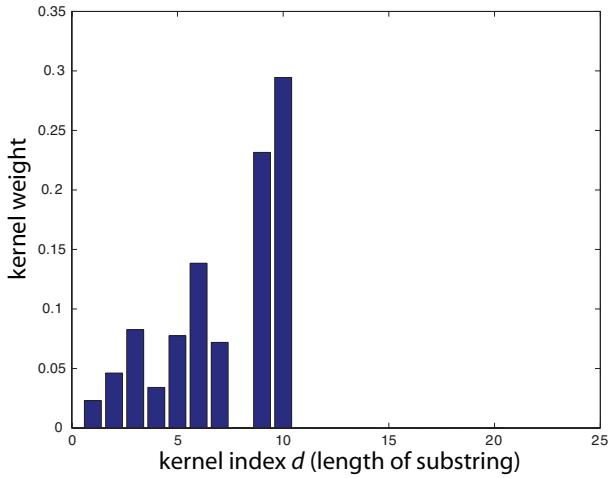


Figure 5: The learned WD kernel weighting on a million of examples.

weighting is displayed in Figure 5 and shows a peak for 6-mers and 9&10-mers. It should be noted that the obtained weighting in this experiment is only partially useful for interpretation. In the case of splice site detection, it is unlikely that k-mers of length 9 or 10 are playing the most important role. More likely to be important are substrings of length up to six. We believe that the large weights for the longest k-mers are an artifact which comes from the fact that we are combining kernels with

quite different properties, i.e. the 9th and 10th kernel leads to a combined kernel matrix that is most diagonally dominant (since the sequences are only similar to themselves but not to other sequences), which we believe is the reason for having a large weight.[12]

In the following example we consider one weight per position. In this case the combined kernels are more similar to each other and we expect more interpretable results. Figure 6 shows an

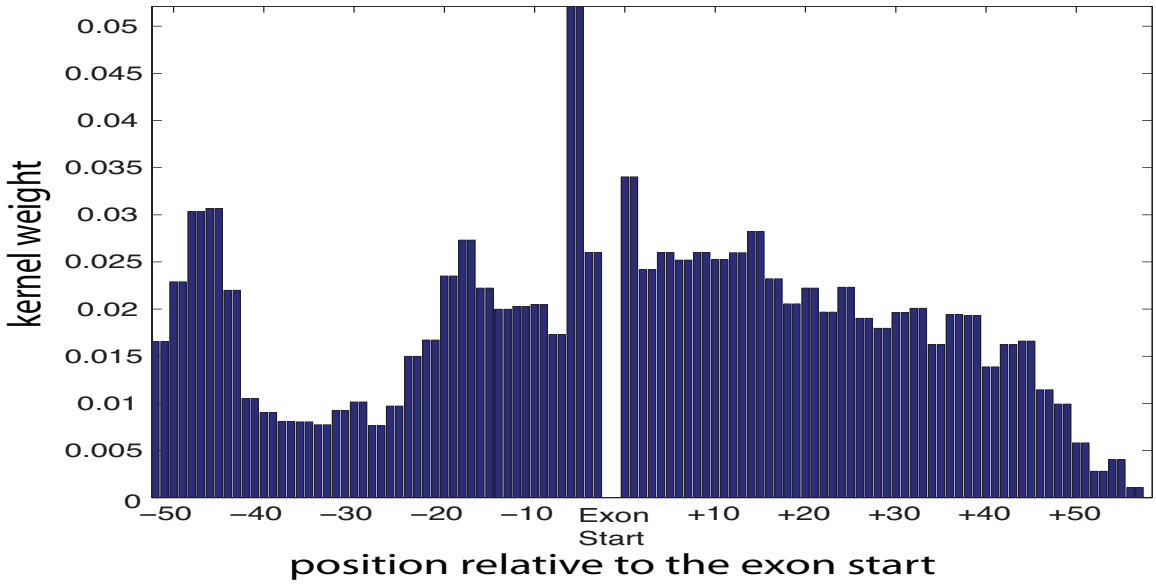


Figure 6: The figure shows an importance weighting for each position in a DNA sequence (around a so called splice site). MKL was used to determine these weights, each corresponding to a sub-kernel which uses information at that position to discriminate splice sites from non-splice sites. Different peaks correspond to different biologically known signals (see text for details). We used 65,000 examples for training with 54 sub-kernels.

importance weighting for each position in a DNA sequence (around a so called acceptor splice site, the start of an exon). We used MKL on 65,000 examples to compute these 54 weights, each corresponding to a sub-kernel which uses information at that position to discriminate true splice sites from fake ones. We repeated that experiment on ten bootstrap runs of the data set. We can identify several interesting regions that we can match to current biological knowledge about splice site recognition: a) The region -50 nucleotides (nt) to -40nt, which corresponds to the donor splice site of the previous exon (many introns in *C. elegans* are very short, often only 50nt), b) the region -25nt to -15nt that coincides with the location of the branch point, c) the intronic region closest to the splice site with greatest weight (-8nt to -1nt; the weights for the AG dimer are zero, since it appears in splice sites and decoys) and d) the exonic region (0nt to $+50$nt). Slightly surprising are the high weights in the exonic region, which we suspect only model triplet frequencies. The

12. This problem might be partially alleviated by including the identity matrix in the convex combination. However as 2-norm soft margin SVMs can be implemented by adding a constant to the diagonal of the kernel (Cortes and Vapnik, 1995), this leads to an additional 2-norm penalization.

decay of the weights seen from +15nt to +45nt might be explained by the fact that not all exons are actually long enough. Furthermore, since the sequence ends in our case at +60nt, the decay after +45nt is an edge effect as longer substrings cannot be matched.

4.2 Benchmarking the Algorithms

Experimental Setup To demonstrate the effect of the several proposed algorithmic optimizations, namely the `linadd` SVM training (Algorithm 3) and for MKL the SILP formulation with and without the `linadd` extension for single, four and eight CPUs, we applied each of the algorithms to a *human* splice site data set,[13] comparing it to the original WD formulation and the case where the weighting coefficients were learned using multiple kernel learning. The splice data set contains 159,771 true acceptor splice site sequences and 14,868,555 decoys, leading to a total of 15,028,326 sequences each 141 base pairs in length. It was generated following a procedure similar to the one in Sonnenburg et al. (2005a) for *C. elegans* which however contained "only" 1,026,036 examples. Note that the data set is very unbalanced as 98.94% of the examples are negatively labeled. We are using this data set in all benchmark experiments and trained (MKL-)SVMs using the `SHOGUN` machine learning toolbox which contains a modified version of SVM$^{\text{light}}$ (Joachims, 1999) on 500, 1,000, 5,000, 10,000, 30,000, 50,000, 100,000, 200,000, 500,000, 1,000,000, 2,000,000, 5,000,000 and 10,000,000 randomly sub-sampled examples and measured the time needed in SVM training. For classification performance evaluation we always use the same remaining 5,028,326 examples as a test data set. We set the degree parameter to $d = 20$ for the WD kernel and to $d = 8$ for the spectrum kernel fixing the SVMs regularization parameter to $C = 5$. Thus in the MKL case also $K = 20$ sub-kernels were used. SVM$^{\text{light}}$'s subproblem size (parameter `qpsize`), convergence criterion (parameter `epsilon`) and MKL convergence criterion were set to $Q = 112$, $\varepsilon_{SVM} = 10^{-5}$ and $\varepsilon_{MKL} = 10^{-5}$, respectively. A kernel cache of 1GB was used for all kernels except the precomputed kernel and algorithms using the `linadd`-SMO extension for which the kernel-cache was disabled. Later on we measure whether changing the quadratic subproblem size Q influences SVM training time. Experiments were performed on a PC powered by *eight* 2.4GHz AMD Opteron(tm) processors running Linux. We measured the training time for each of the algorithms (single, quad or eight CPU version) and data set sizes.

4.2.1 BENCHMARKING SVM

The obtained training times for the different SVM algorithms are displayed in Table 1 and in Figure 7. First, SVMs were trained using standard SVM$^{\text{light}}$ with the Weighted Degree Kernel precomputed (*WDPre*), the standard WD kernel (*WD1*) and the precomputed (*SpecPre*) and standard spectrum kernel (*Spec*). Then SVMs utilizing the `linadd` extension[14] were trained using the WD (*LinWD*) and spectrum (*LinSpec*) kernel. Finally SVMs were trained on four and eight CPUs using the parallel version of the `linadd` algorithm (*LinWD4, LinWD8*). *WD4* and *WD8* demonstrate the effect of a simple parallelization strategy where the computation of kernel rows and updates on the working set are parallelized, which works with *any* kernel.

The training times obtained when precomputing the kernel matrix (which includes the time needed to precompute the full kernel matrix) is lower when no more than 1,000 examples are used.

13. The splice data set can be downloaded from `http://www.fml.tuebingen.mpg.de/raetsch/projects/lsmkl`.
14. More precisely the `linadd` and $\mathcal{O}(L)$ block formulation of the WD kernel as proposed in Sonnenburg et al. (2005b) was used.

Note that this is a direct cause of the relatively large subproblem size $Q = 112$. The picture is different for, say, $Q = 42$ (data not shown) where the *WDPre* training time is in all cases larger than the times obtained using the original WD kernel demonstrating the effectiveness of SVM$^{\text{light}}$'s kernel cache. The overhead of constructing a trie on $Q = 112$ examples becomes even more visible: only starting from 50,000 examples `linadd` optimization becomes more efficient than the original WD kernel algorithm as the kernel cache cannot hold all kernel elements anymore.[15] Thus it would be appropriate to lower the chunking size Q as can be seen in Table 3.

The `linadd` formulation outperforms the original WD kernel by a factor of 3.9 on a million examples. The picture is similar for the spectrum kernel, here speedups of factor 64 on $500,000$ examples are reached which stems from the fact that explicit maps (and not tries as in the WD kernel case) as discussed in Section 3.1.2 could be used leading to a `lookup` cost of $O(1)$ and a dramatically reduced map construction time. For that reason the parallelization effort benefits the WD kernel more than the Spectrum kernel: on one million examples the parallelization using 4 CPUs (8 CPUs) leads to a speedup of factor 3.25 (5.42) for the WD kernel, but only 1.67 (1.97) for the Spectrum kernel. Thus parallelization will help more if the kernel computation is slow. Training with the original WD kernel with a sample size of $1,000,000$ takes about 28 hours, the `linadd` version still requires 7 hours while with the 8 CPU parallel implementation only about 6 hours and in conjunction with the `linadd` optimization a single hour and 20 minutes are needed. Finally, training on 10 million examples takes about 4 days. Note that this data set is already 2.1*GB* in size.

Classification Performance Figure 8 and Table 2 show the classification performance in terms of classification accuracy, area under the Receiver Operator Characteristic (ROC) Curve (Metz, 1978; Fawcett, 2003) and the area under the Precision Recall Curve (PRC) (see e.g. Davis and Goadrich (2006)) of SVMs on the human splice data set for different data set sizes using the WD kernel.

Recall the definition of the ROC and PRC curves: The sensitivity (or recall) is defined as the fraction of correctly classified positive examples among the total number of positive examples, i.e. it equals the true positive rate $TPR = TP/(TP+FN)$. Analogously, the fraction $FPR = FP/(TN+FP)$ of negative examples wrongly classified positive is called the false positive rate. Plotting FPR against TPR results in the Receiver Operator Characteristic Curve (ROC) Metz (1978); Fawcett (2003). Plotting the true positive rate against the positive predictive value (also precision) $PPV = TP/(FP+TP)$, i.e. the fraction of correct positive predictions among all positively predicted examples, one obtains the Precision Recall Curve (PRC) (see e.g. Davis and Goadrich (2006)). Note that as this is a very unbalanced data set the accuracy and the area under the ROC curve are almost meaningless, since both measures are independent of class ratios. The more sensible auPRC, however, steadily increases as more training examples are used for learning. Thus one should train using all available data to obtain state-of-the-art results.

Varying SVM$^{\text{light}}$'s `qpsize` parameter As discussed in Section 3.1.3 and Algorithm 3, using the `linadd` algorithm for computing the output for all training examples w.r.t. to some working set can be speed up by a factor of Q (i.e. the size of the quadratic subproblems, termed `qpsize` in SVM$^{\text{light}}$). However, there is a trade-off in choosing Q as solving larger quadratic subproblems is expensive (quadratic to cubic effort). Table 3 shows the dependence of the computing time from Q and N. For example the gain in speed between choosing $Q = 12$ and $Q = 42$ for 1 million of examples is 54%. Sticking with a mid-range Q (here $Q = 42$) seems to be a good idea for this task. However,

15. When single precision 4-byte floating point numbers are used, caching all kernel elements is possible when training with up to 16384 examples.

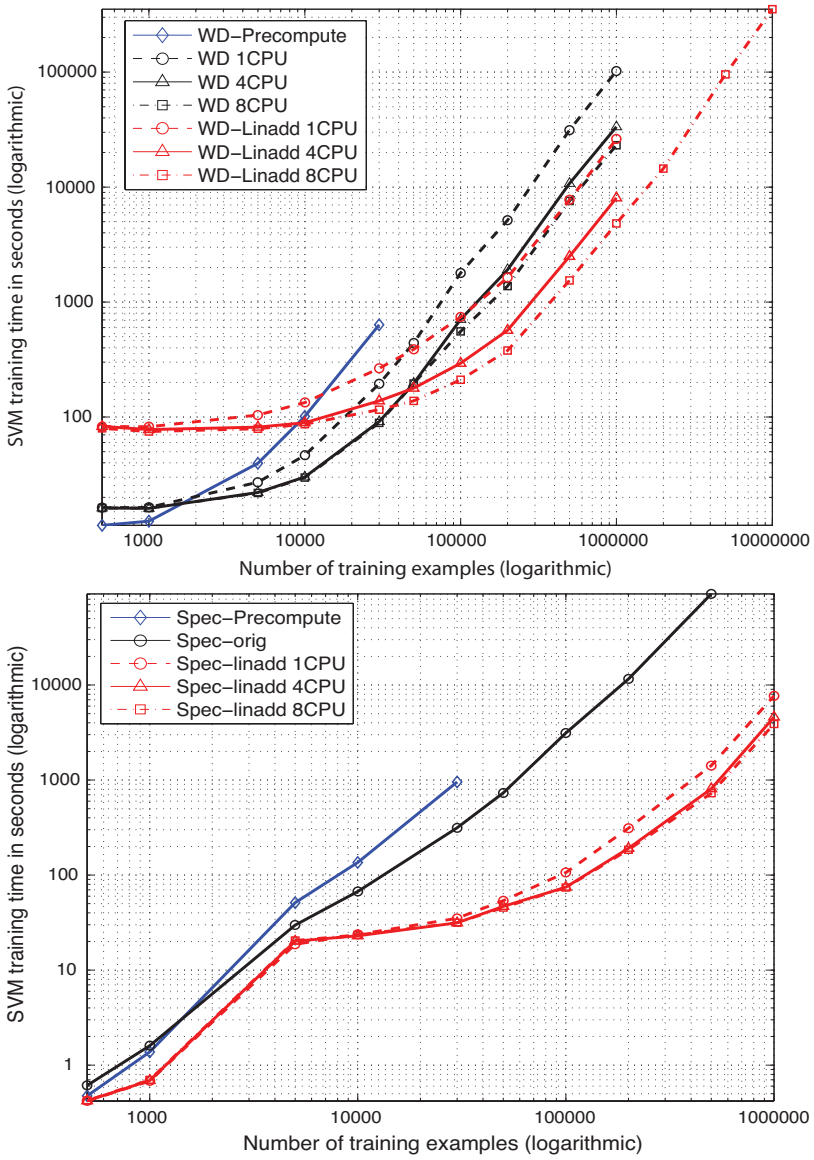


Figure 7: Comparison of the running time of the different SVM training algorithms using the weighted degree kernel. Note that as this is a log-log plot small appearing distances are large for larger N and that each slope corresponds to a different exponent. In the upper figure the Weighted Degree kernel training times are measured, the lower figure displays Spectrum kernel training times.

a large variance can be observed, as the SVM training time depends to a large extend on which Q variables are selected in each optimization step. For example on the related *C. elegans* splice data set $Q = 141$ was optimal for large sample sizes while a midrange $Q = 71$ lead to the overall best

N	*WDPre*	*WD1*	*WD4*	*WD8*	*LinWD1*	*LinWD4*	*LinWD8*
500	12	17	17	17	83	83	80
1,000	13	17	17	17	83	78	75
5,000	40	28	23	22	105	82	80
10,000	102	47	31	30	134	90	87
30,000	636	195	92	90	266	139	116
50,000	-	441	197	196	389	179	139
100,000	-	1,794	708	557	740	294	212
200,000	-	5,153	1,915	1,380	1,631	569	379
500,000	-	31,320	10,749	7,588	7,757	2,498	1,544
1,000,000	-	102,384	33,432	23,127	26,190	8,053	4,835
2,000,000	-	-	-	-	-	-	14,493
5,000,000	-	-	-	-	-	-	95,518
10,000,000	-	-	-	-	-	-	353,227

N	*SpecPre*	*Spec*	*LinSpec1*	*LinSpec4*	*LinSpec8*
500	1	1	1	1	1
1,000	2	2	1	1	1
5,000	52	30	19	21	21
10,000	136	68	24	23	24
30,000	957	315	36	32	32
50,000	-	733	54	47	46
100,000	-	3,127	107	75	74
200,000	-	11,564	312	192	185
500,000	-	91,075	1,420	809	728
1,000,000	-	-	7,676	4,607	3,894

Table 1: **(top)** Speed Comparison of the original single CPU Weighted Degree Kernel algorithm (*WD1*) in SVM^{light} training, compared to the four (*WD4*)and eight (*WD8*) CPUs parallelized version, the precomputed version (Pre) and the `linadd` extension used in conjunction with the original WD kernel for 1,4 and 8 CPUs (*LinWD1, LinWD4, LinWD8*). **(bottom)** Speed Comparison of the spectrum kernel without (*Spec*) and with `linadd` (*LinSpec1, LinSpec4, LinSpec8* using 1,4 and 8 processors). *SpecPre* denotes the precomputed version. The first column shows the sample size *N* of the data set used in SVM training while the following columns display the time (measured in seconds) needed in the training phase.

performance. Nevertheless, one observes the trend that for larger training set sizes slightly larger subproblems sizes decrease the SVM training time.

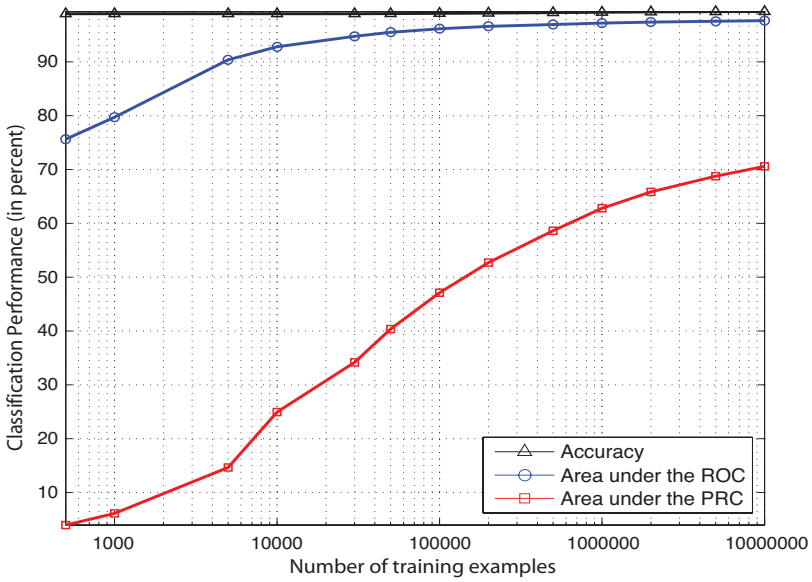


Figure 8: Comparison of the classification performance of the Weighted Degree kernel based SVM classifier for different training set sizes. The area under the Receiver Operator Characteristic (ROC) Curve, the area under the Precision Recall Curve (PRC) as well as the classification accuracy are displayed (in percent). Note that as this is a very unbalanced data set, the accuracy and the area under the ROC curve are less meaningful than the area under the PRC.

4.2.2 BENCHMARKING MKL

The WD kernel of degree 20 consist of a weighted sum of 20 sub-kernels each counting matching d-mers, for $d = 1, \ldots, 20$. Using MKL we learned the weighting on the splice site recognition task for one million examples as displayed in Figure 5 and discussed in Section 4.1.3. Focusing on a speed comparison we now show the obtained training times for the different MKL algorithms applied to learning weightings of the WD kernel on the splice site classification task. To do so, several MKL-SVMs were trained using precomputed kernel matrices (*PreMKL*), kernel matrices which are computed on the fly employing kernel caching (*MKL*[16]), MKL using the `linadd` extension (*LinMKL1*) and `linadd` with its parallel implementation[17] (*LinMKL4* and *LinMKL8* - on 4 and 8 CPUs). The results are displayed in Table 4 and in Figure 9. While precomputing kernel matrices seems beneficial, it cannot be applied to large scale cases (e.g. $> 10,000$ examples) due to the $O(KN^2)$ memory constraints of storing the kernel matrices.[18] On-the-fly-computation of the kernel matrices is computationally extremely demanding, but since kernel caching[19] is used, it is still possible on 50,000 examples in about 57 hours. Note that no WD-kernel specific optimizations are involved here, so one expects a similar result for arbitrary kernels.

16. Algorithm 2.
17. Algorithm 2 with the `linadd` extensions including parallelization of Algorithm 4.
18. Using 20 kernels on 10,000 examples requires already 7.5GB, on 30,000 examples 67GB would be required (both using single precision floats).
19. Each kernel has a cache of 1GB.

N	*Accuracy*	*auROC*	*auPRC*
500	98.93	75.61	3.97
1,000	98.93	79.70	6.12
5,000	98.93	90.38	14.66
10,000	98.93	92.79	24.95
30,000	98.93	94.73	34.17
50,000	98.94	95.48	40.35
100,000	98.98	96.13	47.11
200,000	99.05	96.58	52.70
500,000	99.14	96.93	58.62
1,000,000	99.21	97.20	62.80
2,000,000	99.26	97.36	65.83
5,000,000	99.31	97.52	68.76
10,000,000	99.35	97.64	70.57
10,000,000	-	96.03*	44.64*

Table 2: Comparison of the classification performance of the Weighted Degree kernel based SVM classifier for different training set sizes. The area under the ROC curve (*auROC*), the area under the Precision Recall Curve (*auPRC*) as well as the classification accuracy (*Accuracy*) are displayed (in percent). Larger values are better. A optimal classifier would achieve 100% Note that as this is a very unbalanced data set the accuracy and the area under the ROC curve are almost meaningless. For comparison, the classification performance achieved using a 4th order Markov chain on 10 million examples (order 4 was chosen based on model selection, where order 1 to 8 using several pseudo-counts were tried) is displayed in the last row (marked *).

The `linadd` variants outperform the other algorithms by far (speedup factor 53 on 50,000 examples) and are still applicable to data sets of size up to one million. Note that without parallelization MKL on one million examples would take more than a week, compared with 2.5 (2) days in the quad-CPU (eight-CPU) version. The parallel versions outperform the single processor version from the start achieving a speedup for 10,000 examples of 2.27 (2.75), quickly reaching a plateau at a speedup factor of 2.98 (4.49) at a level of $50,000$ examples and approaching a speedup factor of 3.28 (5.53) on 500,000 examples (efficiency: 82% (69%)). Note that the performance gain using 8 CPUs is relatively small as e.g. solving the QP and constructing the tree is not parallelized.

5. Conclusion

In the first part of the paper we have proposed a simple, yet efficient algorithm to solve the multiple kernel learning problem for a large class of loss functions. The proposed method is able to exploit the existing single kernel algorithms, thereby extending their applicability. In experiments we have illustrated that MKL for classification and regression can be useful for automatic model selection and for obtaining comprehensible information about the learning problem at hand. It would be of interest to develop and evaluate MKL algorithms for unsupervised learning such as Kernel PCA

N	112	Q 12	32	42	52	72
500	83	**4**	**1**	22	68	67
1,000	83	**7**	**7**	**11**	34	60
5,000	105	**15**	**21**	33	31	68
10,000	134	**32**	**38**	54	67	97
30,000	266	**128**	**128**	**127**	160	187
50,000	389	258	**217**	242	252	309
100,000	740	696	**494**	585	573	643
200,000	1,631	1,875	1,361	**1,320**	1,417	1,610
500,000	7,757	9,411	6,558	**6,203**	6,583	7,883
1,000,000	26,190	31,145	20,831	**20,136**	21,591	24,043

Table 3: Influence on training time when varying the size of the quadratic program Q in SVM$^{\text{light}}$, when using the `linadd` formulation of the WD kernel. While training times do not vary dramatically one still observes the tendency that with larger sample size a larger Q becomes optimal. The $Q = 112$ column displays the same result as column *LinWD1* in Table 1.

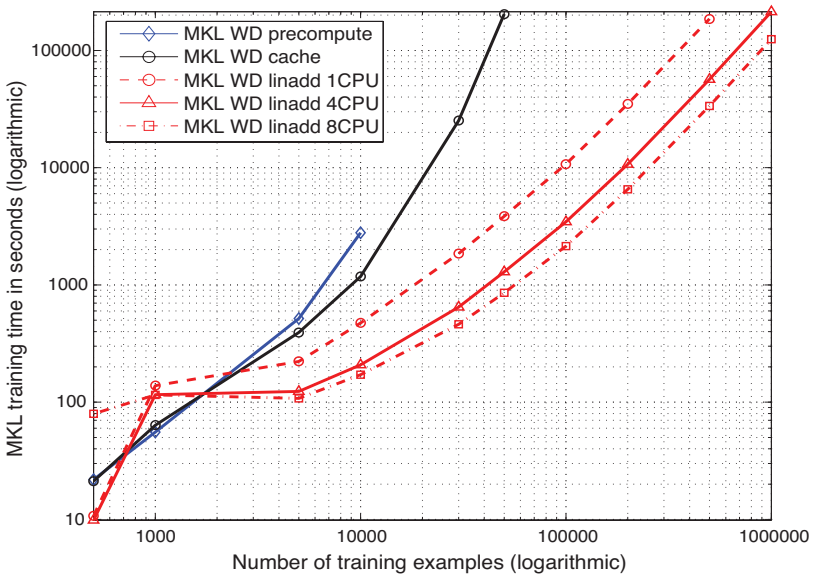


Figure 9: Comparison of the running time of the different MKL algorithms when used with the weighted degree kernel. Note that as this is a log-log plot, small appearing distances are large for larger N and that each slope corresponds to a different exponent.

and one-class classification and to try different losses on the kernel weighting β (such as L_2). In the second part we proposed performance enhancements to make large scale MKL practical: the SILP wrapper, SILP chunking and (for the special case of kernels that can be written as an inner product of sparse feature vectors, e.g., string kernels) the `linadd` algorithm, which also speeds up

N	*PreMKL*	*MKL*	*LinMKL1*	*LinMKL4*	*LinMKL8*
500	22	22	11	10	80
1,000	56	64	139	116	116
5,000	518	393	223	124	108
10,000	2,786	1,181	474	209	172
30,000	-	25,227	1,853	648	462
50,000	-	204,492	3,849	1292	857
100,000	-	-	10,745	3,456	2,145
200,000	-	-	34,933	10,677	6,540
500,000	-	-	185,886	56,614	33,625
1,000,000	-	-	-	214,021	124,691

Table 4: Speed Comparison when determining the WD kernel weight by multiple kernel learning using the chunking algorithm (MKL) and MKL in conjunction with the (parallelized) `linadd` algorithm using 1, 4, and 8 processors (*LinMKL1, LinMKL4, LinMKL8*). The first column shows the sample size N of the data set used in SVM training while the following columns display the time (measured in seconds) needed in the training phase.

standalone SVM training. For the standalone SVM using the spectrum kernel we achieved speedups of factor 64 (for the weighted degree kernel, about 4). For MKL we gained a speedup of factor 53. Finally we proposed a parallel version of the `linadd` algorithm running on a 8 CPU multiprocessor system which lead to *additional* speedups of factor up to 5.5 for MKL, and 5.4 for vanilla SVM training.

Acknowledgments

The authors gratefully acknowledge partial support from the PASCAL Network of Excellence (EU #506778), DFG grants JA 379 / 13-2 and MU 987/2-1. We thank Guido Dornhege, Olivier Chapelle, Cheng Soon Ong, Joaquin Quiñoñero Candela, Sebastian Mika, Jason Weston, Manfred Warmuth and K.-R. Müller for great discussions.

Appendix A. Derivation of the MKL Dual for Generic Loss Functions

We start from the MKL primal problem Equation (9):

$$\begin{aligned} \min \quad & \frac{1}{2}\left(\sum_{k=1}^{K}\|\mathbf{w}_k\|\right)^2 + \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) \\ \text{w.r.t.} \quad & \mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_K) \in \mathbb{R}^{D_1} \times \cdots \times \mathbb{R}^{D_K} \\ \text{s.t.} \quad & f(\mathbf{x}_i) = \sum_{k=1}^{K} \langle \Phi_k(\mathbf{x}_i), \mathbf{w}_k \rangle + b, \;\; \forall i = 1, \ldots, N \end{aligned}$$

Introducing $u \in \mathbb{R}$ allows us to move $\sum_{k=1}^{K} \|\mathbf{w}_k\|$ into the constraints and leads to the following equivalent problem

$$\begin{aligned} \min \quad & \frac{1}{2}u^2 + \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) \\ \text{w.r.t.} \quad & u \in \mathbb{R},\ (\mathbf{w}_1, \ldots, \mathbf{w}_K) \in \mathbb{R}^{D_1} \times \cdots \times \mathbb{R}^{D_K} \\ \text{s.t.} \quad & f(\mathbf{x}_i) = \sum_{k=1}^{K} \langle \Phi_k(\mathbf{x}_i), \mathbf{w}_k \rangle + b,\ \forall i = 1, \ldots, N \\ & \sum_{k=1}^{K} \|\mathbf{w}_k\| \leq u \end{aligned}$$

Using $t_k \in \mathbb{R},\ k = 1, \ldots, K$, it can be equivalently transformed into

$$\begin{aligned} \min \quad & \frac{1}{2}u^2 + \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) \\ \text{w.r.t.} \quad & u \in \mathbb{R},\ t_k \in \mathbb{R}, \mathbf{w}_k \in \mathbb{R}^{D_k},\ \forall k = 1, \ldots, K \\ \text{s.t.} \quad & f(\mathbf{x}_i) = \sum_{k=1}^{K} \langle \Phi_k(\mathbf{x}_i), \mathbf{w}_k \rangle + b,\ \forall i = 1, \ldots, N \\ & \|\mathbf{w}_k\| \leq t_k,\ \sum_{k=1}^{K} t_k \leq u. \end{aligned}$$

Recall that the second-order cone of dimensionality D is defined as

$$\mathcal{K}_D = \{(\mathbf{x}, c) \in \mathbb{R}^D \times \mathbb{R},\ \|\mathbf{x}\|_2 \leq c\}.$$

We can thus reformulate the original MKL primal problem (Equation (9)) using the following *equivalent* second-order cone program, as the norm constraint on $\mathbf{w}_k$ is implicitly taken care of:

Conic Primal

$$\begin{aligned} \min \quad & \frac{1}{2}u^2 + \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) \\ \text{w.r.t.} \quad & u \in \mathbb{R},\ t_k \in \mathbb{R}, (\mathbf{w}_k, t_k) \in \mathcal{K}_{D_k},\ \forall k = 1, \ldots, K \\ \text{s.t.} \quad & f(\mathbf{x}_i) = \sum_{k=1}^{K} \langle \Phi_k(\mathbf{x}_i), \mathbf{w}_k \rangle + b,\ \forall i = 1, \ldots, N \\ & \sum_{k=1}^{K} t_k \leq u \end{aligned}$$

We are now going to derive the conic dual following the recipe of Boyd and Vandenberghe (2004) (see p. 266). First we derive the conic Lagrangian and then using the infimum w.r.t. the primal variables in order to obtain the conic dual. We therefore introduce Lagrange multipliers $\alpha \in \mathbb{R}^K$, $\gamma \in \mathbb{R}$, $\gamma \geq 0$ and $(\lambda_k, \mu_k) \in {\mathcal{K}^*}_D$ living on the self dual cone ${\mathcal{K}^*}_D = \mathcal{K}_D$. Then the conic

Lagrangian is given as

$$\begin{aligned}\mathcal{L}(\mathbf{w},b,\mathbf{t},u,\alpha,\gamma,\lambda,\mu) &= \frac{1}{2}u^2+\sum_{i=1}^{N}L(f(\mathbf{x}_i),y_i)-\sum_{i=1}^{N}\alpha_i f(\mathbf{x}_i)+\\ &+\sum_{i=1}^{N}\alpha_i\sum_{k=1}^{K}(\langle\Phi_k(\mathbf{x}_i),\mathbf{w}_k\rangle+b)+\gamma\left(\sum_{k=1}^{K}t_k-u\right)-\sum_{k=1}^{K}(\langle\lambda_k,\mathbf{w}_k\rangle+\mu_k t_k).\end{aligned}$$

To obtain the dual, the derivatives of the Lagrangian w.r.t. the primal variables, $\mathbf{w},b,\mathbf{t},u$ have to vanish which leads to the following constraints

$$\begin{aligned}\partial_{\mathbf{w}_k}\mathcal{L} &= \sum_{i=1}^{N}\alpha_i\Phi_k(\mathbf{x}_i)-\lambda_k \Rightarrow \lambda_k=\sum_{i=1}^{N}\alpha_i\Phi_k(\mathbf{x}_i)\\ \partial_b\mathcal{L} &= \sum_{i=1}^{N}\alpha_i \Rightarrow \sum_{i=1}^{N}\alpha_i=0\\ \partial_{t_k}\mathcal{L} &= \gamma-\mu_k \Rightarrow \gamma=\mu_k\\ \partial_u\mathcal{L} &= u-\gamma\Rightarrow\gamma=u\\ \partial_{f(\mathbf{x}_i)}\mathcal{L} &= L'(f(\mathbf{x}_i),y_i)-\alpha_i\Rightarrow f(\mathbf{x}_i)=L'^{-1}(\alpha_i,y_i).\end{aligned}$$

In the equation L' is the derivative of the loss function w.r.t. $f(x)$ and L'^{-1} is the inverse of L' (w.r.t. $f(x)$) for which to exist L is required to be strictly convex and differentiable. We now plug in what we have obtained above, which makes λ_k, μ_k and all of the primal variables vanish. Thus the dual function is

$$\begin{aligned}D(\alpha,\gamma) &= -\frac{1}{2}\gamma^2+\sum_{i=1}^{N}L(L'^{-1}(\alpha_i,y_i),y_i)-\sum_{i=1}^{N}\alpha_i L'^{-1}(\alpha_i,y_i)+\\ &+\sum_{i=1}^{N}\alpha_i\sum_{k=1}^{K}\langle\Phi_k(\mathbf{x}_i),\mathbf{w}_k\rangle-\sum_{k=1}^{K}\sum_{i=1}^{N}\alpha_i\langle\Phi_k(\mathbf{x}_i),\mathbf{w}_k\rangle\\ &= -\frac{1}{2}\gamma^2+\sum_{i=1}^{N}L(L'^{-1}(\alpha_i,y_i),y_i)-\sum_{i=1}^{N}\alpha_i L'^{-1}(\alpha_i,y_i).\end{aligned}$$

As constraints remain $\gamma\geq 0$, due to the bias $\sum_{i=1}^{N}\alpha_i=0$ and the second-order cone constraints

$$\|\lambda_k\|=\left\|\sum_{i=1}^{N}\alpha_i\Phi_k(\mathbf{x}_i)\right\|_2\leq\gamma,\ \forall k=1,\ldots,K.$$

This leads to:

$$\begin{aligned}\max\quad & -\frac{1}{2}\gamma^2+\sum_{i=1}^{N}L(L'^{-1}(\alpha_i,y_i),y_i)-\sum_{i=1}^{N}\alpha_i L'^{-1}(\alpha_i,y_i)\\ \text{w.r.t.}\quad & \gamma\in\mathbb{R},\ \alpha\in R^N\\ \text{s.t.}\quad & \gamma\geq 0,\ \sum_{i=1}^{N}\alpha_i=0\\ & \left\|\sum_{i=1}^{N}\alpha_i\Phi_k(\mathbf{x}_i)\right\|_2\leq\gamma,\ \forall k=1,\ldots,K\end{aligned}$$

Squaring the latter constraint, multiplying by $\frac{1}{2}$, relabeling $\frac{1}{2}\gamma^2 \mapsto \gamma$ and dropping the $\gamma \geq 0$ constraint as it is fulfilled implicitly, we obtain the MKL dual for arbitrary strictly convex loss functions.

Conic Dual

$$\begin{aligned}
\min \quad & \underbrace{\gamma - \sum_{i=1}^{N} L(L'^{-1}(\alpha_i, y_i), y_i) + \sum_{i=1}^{N} \alpha_i L'^{-1}(\alpha_i, y_i)}_{:=T} \\
\text{w.r.t.} \quad & \gamma \in \mathbb{R},\ \alpha \in R^N \\
\text{s.t.} \quad & \sum_{i=1}^{N} \alpha_i = 0 \\
& \frac{1}{2} \left\| \sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i) \right\|_2^2 \leq \gamma,\ \forall k = 1, \ldots, K.
\end{aligned}$$

Finally adding the second term in the objective (T) to the constraint on γ and relabeling $\gamma + T \mapsto \gamma$ leads to the reformulated dual Equation (10), the starting point from which one can derive the SILP formulation in analogy to the classification case.

Appendix B. Loss Functions

B.1 Quadratic Loss

For the quadratic loss case $L(x,y) = C(x-y)^2$ we obtain as the derivative $L'(x,y) = 2C(x-y) =: z$ and $L'^{-1}(z,y) = \frac{1}{2C}z + y$ for the inverse of the derivative. Recall the definition of

$$S_k(\alpha) = -\sum_{i=1}^{N} L(L'^{-1}(\alpha_i, y_i), y_i) + \sum_{i=1}^{N} \alpha_i L'^{-1}(\alpha_i, y_i) + \frac{1}{2} \left\| \sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i) \right\|_2^2.$$

Plugging in L, L'^{-1} leads to

$$\begin{aligned}
S_k(\alpha) &= -\sum_{i=1}^{N} \left(\frac{1}{2C}\alpha_i + y_i - y_i\right)^2 + \sum_{i=1}^{N} \alpha_i \left(\frac{1}{2C}\alpha_i + y_i\right) + \frac{1}{2} \left\| \sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i) \right\|_2^2 \\
&= \frac{1}{4C} \sum_{i=1}^{N} \alpha_i^2 + \sum_{i=1}^{N} \alpha_i y_i + \frac{1}{2} \left\| \sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i) \right\|_2^2.
\end{aligned}$$

B.2 Logistic Loss

Very similar to the Hinge loss the derivation for the logistic loss $L(x,y) = \log(1 + e^{-xy})$ will be given for completeness.

$$L'(x,y) = \frac{-ye^{-xy}}{1+e^{-xy}} = -\frac{ye^{(1-xy)}}{1+e^{(1-xy)}} =: z.$$

The inverse function for $y \neq 0$ and $y + z \neq 0$ is given by

$$L'^{-1}(z,y) = -\frac{1}{y} \log\left(-\frac{z}{y+z}\right)$$

and finally we obtain

$$S_k(\alpha) = \sum_{i=1}^{N} \log\left(1 - \frac{\alpha_i}{y_i + \alpha_i}\right) - \sum_{i=1}^{N} \frac{\alpha_i}{y_i} \log\left(-\frac{\alpha_i}{y_i + \alpha_i}\right) + \frac{1}{2}\left\|\sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i)\right\|_2^2.$$

B.3 Smooth Hinge Loss

Using the Hinge Loss $L(x,y) = \frac{C}{\sigma}\log(1+e^{\sigma(1-xy)})$ with $\sigma > 0$, $y \in \mathbb{R}$ fixed, $x \in \mathbb{R}$ one obtains as derivative

$$L'(x,y) = \frac{-\sigma C y e^{\sigma(1-xy)}}{\sigma(1+e^{\sigma(1-xy)})} = -\frac{Cye^{\sigma(1-xy)}}{1+e^{\sigma(1-xy)}} =: z.$$

Note that with y fixed, z is bounded: $0 \leq \mathrm{abs}(z) \leq \mathrm{abs}(Cy)$ and $\mathrm{sign}(y) = -\mathrm{sign}(z)$ and therefore $-\frac{z}{Cy+z} > 0$ for $Cy + z \neq 0$. The inverse function is derived as

$$\begin{aligned}
z + ze^{\sigma(1-xy)} &= -Cye^{\sigma(1-xy)} \\
(Cy+z)e^{\sigma(1-xy)} &= -z \\
e^{\sigma(1-xy)} &= -\frac{z}{Cy+z} \\
\sigma(1-xy) &= \log(-\frac{z}{Cy+z}) \\
1-xy &= \frac{1}{\sigma}\log(-\frac{z}{Cy+z}) \\
x &= \frac{1}{y}(1-\frac{1}{\sigma}\log(-\frac{z}{Cy+z})),\ y \neq 0 \\
L'^{-1}(z,y) &= \frac{1}{y}(1-\frac{1}{\sigma}\log(-\frac{z}{Cy+z}))
\end{aligned}$$

Define $C_1 = \frac{1}{2}\left\|\sum_{i=1}^{N} \alpha_i \Phi_k(\mathbf{x}_i)\right\|_2^2$ and $C_2 = \sum_{i=1}^{N} \alpha_i \frac{1}{y_i}\left(1 - \frac{1}{\sigma}\log(-\frac{\alpha_i}{Cy_i+\alpha_i})\right)$

Using these ingredients it follows for $S_k(\alpha)$

$$\begin{aligned}
S_k(\alpha) &= -\sum_{i=1}^{N} L\left(\frac{1}{y_i}\left(1 - \frac{1}{\sigma}\log(-\frac{\alpha_i}{Cy_i+\alpha_i})\right), y_i\right) + C_2 + C_1 \\
&= -\sum_{i=1}^{N} \frac{1}{\sigma}\log\left(1 + e^{\sigma\left(1-\left(\frac{y_i}{y_i}\left(1-\frac{1}{\sigma}\log(-\frac{\alpha_i}{Cy_i+\alpha_i})\right)\right)\right)}\right) + C_2 + C_1 \\
&= -\sum_{i=1}^{N} \frac{1}{\sigma}\log\left(1 - \frac{\alpha_i}{Cy_i+\alpha_i}\right) + \sum_{i=1}^{N} \frac{\alpha_i}{y_i}\left(1 - \frac{1}{\sigma}\log(-\frac{\alpha_i}{Cy_i+\alpha_i})\right) + C_1.
\end{aligned}$$

References

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In C. E. Brodley, editor, *Twenty-first international conference on Machine learning*. ACM, 2004.

K. P. Bennett, M. Momma, and M. J. Embrechts. MARK: a boosting algorithm for heterogeneous kernel models. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 24–31. ACM, 2002.

J. Bi, T. Zhang, and K. P. Bennett. Column-generation boosting methods for mixture of kernels. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 521–526. ACM, 2004.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002.

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

J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. Technical report #1551, University of Wisconsin Madison, January 2006.

T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. Technical report hpl-2003-4, HP Laboratories, Palo Alto, CA, USA, January 2003.

E. Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, 1960.

Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in SVMs. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 553–560, Cambridge, MA, 2003. MIT Press.

R. Hettich and K. O. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Review*, 3:380–429, 1993.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, Lecture Notes in Computer Science, pages 137–142, Berlin / Heidelberg, 1998. Springer-Verlag.

T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, USA, 1999. MIT Press.

G.R.G. Lanckriet, T. De Bie, N. Cristianini, M.I. Jordan, and W.S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20:2626–2635, 2004.

C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In R. B. Altman, A. K. Dunker, L. Hunter, K. Lauderdale, and T. E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, Kaua'i, Hawaii, 2002.

C. Leslie, R. Kuang, and E. Eskin. Inexact matching string kernels for protein classification. In *Kernel Methods in Computational Biology*, MIT Press series on Computational Molecular Biology, pages 95–112. MIT Press, 2004.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Proc. of the first Machine Learning Summer School in Canberra*, LNCS, pages 119–184. Springer, 2003.

C.E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, VIII(4), October 1978.

C. S. Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, volume 15, pages 478–485, Cambridge, MA, 2003. MIT Press.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, USA, 1999. MIT Press.

G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Potsdam, Germany, 2001.

G. Rätsch and S. Sonnenburg. *Accurate Splice Site Prediction for Caenorhabditis Elegans*, pages 277–298. MIT Press series on Computational Molecular Biology. MIT Press, 2004.

G. Rätsch and M.K. Warmuth. Efficient margin maximization with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.

G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1–3):193–221, 2002. Special Issue on New Methods for Model Selection and Model Combination. Also NeuroCOLT2 Technical Report NC-TR-2000-085.

G. Rätsch, S. Sonnenburg, and B. Schölkopf. RASE: Recognition of alternatively spliced exons in C. elegans. *Bioinformatics*, 21:i369–i377, 2005.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

S. Sonnenburg, G. Rätsch, and C. Schäfer. Learning interpretable SVMs for biological sequence classification. In S. Miyano, J. P. Mesirov, S. Kasif, S. Istrail, P. A. Pevzner, and M. Waterman, editors, *Research in Computational Molecular Biology, 9th Annual International Conference, RECOMB 2005*, volume 3500, pages 389–407. Springer-Verlag Berlin Heidelberg, 2005a.

S. Sonnenburg, G. Rätsch, and B. Schölkopf. Large scale genomic sequence SVM classifiers. In L. D. Raedt and S. Wrobel, editors, *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 849–856, New York, NY, USA, 2005b. ACM Press.

M.K. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In *ICML '06: Proceedings of the 23nd international conference on Machine learning*. ACM Press, 2006.

Journal of Machine Learning Research 7 (2006) 1567–1599 Submitted 10/05; Revised 2/06; Published 7/06

Efficient Learning of Label Ranking by Soft Projections onto Polyhedra

Shai Shalev-Shwartz[*] SHAIS@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University
Jerusalem 91904, Israel

Yoram Singer[†] SINGER@GOOGLE.COM
Google Inc.
1600 Amphitheatre Pkwy
Mountain View CA 94043, USA

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

We discuss the problem of learning to rank labels from a real valued feedback associated with each label. We cast the feedback as a preferences graph where the nodes of the graph are the labels and edges express preferences over labels. We tackle the learning problem by defining a loss function for comparing a predicted graph with a feedback graph. This loss is materialized by decomposing the feedback graph into bipartite sub-graphs. We then adopt the maximum-margin framework which leads to a quadratic optimization problem with linear constraints. While the size of the problem grows quadratically with the number of the nodes in the feedback graph, we derive a problem of a significantly smaller size and prove that it attains the same minimum. We then describe an efficient algorithm, called SOPOPO, for solving the reduced problem by employing a soft projection onto the polyhedron defined by a reduced set of constraints. We also describe and analyze a wrapper procedure for batch learning when multiple graphs are provided for training. We conclude with a set of experiments which show significant improvements in run time over a state of the art interior-point algorithm.

1. Introduction

To motivate the problem discussed in this paper let us consider the following application. Many news feeds such as Reuters and Associated Press tag each news article they handle with labels drawn from a predefined set of possible topics. These tags are used for routing articles to different targets and clients. Each tag may also be associated with a degree of relevance, often expressed as a numerical value, which reflects to what extent a topic is relevant to the news article on hand. Tagging each individual article is clearly a laborious and time consuming task. In this paper we describe and analyze an efficient algorithmic framework for learning and inferring preferences over labels. Furthermore, in addition to the task described above, our learning apparatus includes as special cases problems ranging from binary classification to total order prediction.

∗. Part of S. Shalev-Shwartz's work was done while visiting Google.
†. Author for correspondences.

We focus on batch learning in which the learning algorithm receives a set of training examples, each example consists of an instance and a target vector. The goal of the learning process is to deduce an accurate mapping from the instance space to the target space. The target space $\mathcal{Y}$ is a *predefined* set of labels. For concreteness, we assume that $\mathcal{Y} = \{1, 2, \ldots, k\}$. The prediction task is to assert preferences over the labels. This setting in particular generalizes the notion of a single tag or label $y \in \mathcal{Y} = \{1, 2, \ldots, k\}$, typically used in multiclass categorization tasks, to a full set of preferences over the labels. Preferences are encoded by a vector $\gamma \in \mathbb{R}^k$, where $\gamma_y > \gamma_{y'}$ means that label y is more relevant to the instance than label y'. The preferences over the labels can also be described as a weighted directed graph: the nodes of the graph are the labels and weighted edges encode pairwise preferences over pairs of labels. In Fig. 1 we give the graph representation for the target vector $(-1, 0, 2, 0, -1)$ where each edge marked with its weight. For instance, the weight of the edge $(3, 1)$ is $\gamma_3 - \gamma_1 = 3$.

The class of mappings we employ in this paper is the set of linear functions. While this function class may seem restrictive, the pioneering work of Vapnik (1998) and colleagues demonstrates that by using Mercer kernels one can employ highly non-linear predictors, called support vector machines (SVM) and still entertain all the formal properties and simplicity of linear predictors. We propose a SVM-like learning paradigm for predicting the preferences over labels. We generalize the definition of the hinge-loss used in SVM to the label ranking setting. Our generalized hinge loss contrasts the predicted preferences graph and the target preferences graph by decomposing the target graph into bipartite sub-graphs. As we discuss in the next section, this decomposition into sub-graphs is rather flexible and enables us to analyze several previously defined loss functions in a single unified setting. This definition of the generalized hinge loss lets us pose the learning problem as a quadratic optimization problem while the structured decomposition leads to an efficient and effective optimization procedure.

The main building block of our optimization procedure is an algorithm which performs fast and frugal SOft Projections Onto a POlyhedron and is therefore abbreviated SOPOPO. Generalizing the iterative algorithm proposed by Hildreth (1957) (see also Censor and Zenios (1997)) from half-space constraints to polyhedra constraints, we also derive and analyze an iterative algorithm which on each iteration performs a soft projection onto a single polyhedron. The end result is a fast optimization procedure for label ranking from general real-valued feedback.

The paper is organized as follows. In Sec. 2 we start with a formal definition of our setting and cast the learning task as a quadratic programming problem. We also make references to previous work on related problems that are covered by our setting. Our efficient optimization procedure for the resulting quadratic problem is described in two steps. First, we present in Sec. 3 the SOPOPO algorithm for projecting onto a single polyhedron. Then, in Sec. 4, we derive and analyze an iterative algorithm which solves the original quadratic optimization problem by successive activations of SOPOPO. Experiments are provided in Sec. 5 and concluding remarks are given in Sec. 6.

Before moving to the specifics, we would like to stress that while the learning task discussed in this paper is well rooted in the machine learning community, the focus of the paper is the design and analysis of an optimization apparatus. The readers interested in the broad problem of learning preferences, including its learning theoretic facets such as generalization properties are referred for instance to (Cohen et al., 1999; Herbrich et al., 2000; Rudin et al., 2005; Agarwal and Niyogi, 2005; Clemenon et al., 2005) and the many references therein.

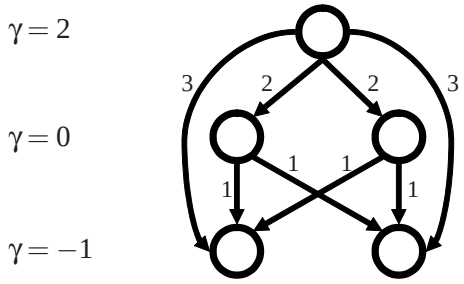


Figure 1: The graph induced by the feedback $\gamma = (-1,0,2,0,-1)$.

2. Problem Setting

In this section we introduce the notation used throughout the paper and formally describe our problem setting. We denote scalars with lower case letters (e.g. x and α), and vectors with bold face letters (e.g. $\mathbf{x}$ and $\boldsymbol{\alpha}$). Sets are designated by upper case Latin letters (e.g. E) and set of sets by bold face (e.g. $\mathbf{E}$). The set of non-negative real numbers is denoted by $\mathbb{R}_+$. For any $k \geq 1$, the set of integers $\{1,\ldots,k\}$ is denoted by $[k]$. We use the notation $(a)_+$ to denote the hinge function, namely, $(a)_+ = \max\{0,a\}$.

Let $\mathcal{X}$ be an instance domain and let $\mathcal{Y} = [k]$ be a predefined set of labels. A target for an instance $\mathbf{x} \in \mathcal{X}$ is a vector $\gamma \in \mathbb{R}^k$ where $\gamma_y > \gamma_{y'}$ means that y is more relevant to $\mathbf{x}$ than y'. We also refer to γ as a label ranking. We would like to emphasize that two different labels may attain the same rank, that is, $\gamma_y = \gamma_{y'}$ while $y \neq y'$. In this case, we say that y and y' are of equal relevance to $\mathbf{x}$. We can also describe γ as a weighted directed graph. The nodes of the graph are labeled by the elements of $[k]$ and there is a directed edge of weight $\gamma_r - \gamma_s$ from node r to node s iff $\gamma_r > \gamma_s$. In Fig. 1 we give the graph representation for the label-ranking vector $\gamma = (-1,0,2,0,-1)$.

The learning goal is to learn a ranking function of the form $\mathbf{f} : \mathcal{X} \to \mathbb{R}^k$ which takes $\mathbf{x}$ as an input instance and returns a ranking vector $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^k$. We denote by $f_r(\mathbf{x})$ the rth element of $\mathbf{f}(\mathbf{x})$. Analogous to the target vector, γ, we say that label y is more relevant than label y' with respect to the predicted ranking if $f_y(\mathbf{x}) > f_{y'}(\mathbf{x})$. We assume that the label-ranking functions are linear, namely,

$$f_r(\mathbf{x}) = \mathbf{w}_r \cdot \mathbf{x} \ ,$$

where each $\mathbf{w}_r$ is a vector in $\mathbb{R}^n$ and $\mathcal{X} \subseteq \mathbb{R}^n$. As we discuss briefly at the end of Sec. 4, our algorithm can be generalized straightforwardly to non-linear ranking functions by employing Mercer kernels (Vapnik, 1998).

We focus on a batch learning setting in which a training set $S = \{(\mathbf{x}^i, \gamma^i)\}_{i=1}^m$ is provided. Thus, each example consists of an instance $\mathbf{x}^i \in \mathcal{X}$ and a label-ranking $\gamma^i \in \mathbb{R}^k$. The performance of a label-ranking function $\mathbf{f}$ on an example $(\mathbf{x}, \gamma)$ is evaluated via a loss function $\ell : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$. Clearly, we want the loss of a predicted ranking to be small if it expresses similar preferences over pairs as the given label-ranking. Moreover, we view the difference $\gamma_r - \gamma_s$ for a pair of labels r and s as an encoding of the importance of the ordering of r ahead of s. That is, the larger this difference is the more we prefer r over s. We view this requirement as a lower bound on the difference between $f_r(\mathbf{x})$ and $f_s(\mathbf{x})$. Formally, for each pair of labels $(r,s) \in \mathcal{Y} \times \mathcal{Y}$ such that $\gamma_r > \gamma_s$, we define the loss of $\mathbf{f}$

with respect to the pair as,

$$\ell_{r,s}(\mathbf{f}(\mathbf{x}),\gamma) \;=\; ((\gamma_r-\gamma_s)-(f_r(\mathbf{x})-f_s(\mathbf{x})))_+ \quad . \tag{1}$$

The above definition of loss extends the hinge-loss used in binary classification problems (Vapnik, 1998) to the problem of label-ranking. The loss $\ell_{r,s}$ reflects the amount by which the constraint $f_r(\mathbf{x})-f_s(\mathbf{x})\geq\gamma_r-\gamma_s$ is not satisfied. While the construction above is defined for pairs, our goal though is to associate a loss with the *entire* predicted ranking and not a single pair. Thus, we need to combine the individual losses over pairs into one meaningful loss. In this paper we take a rather flexible approach by specifying an apparatus for combining the individual losses over pairs into a single loss. We combine the different pair-based losses into a single loss by grouping the pairs of labels into independent sets each of which is isomorphic to a *complete bipartite* graph. Formally, given a target label-ranking vector $\gamma\in\mathbb{R}^k$, we define $\mathbf{E}(\gamma)=\{E_1,\ldots,E_d\}$ to be a collection of subsets of $\mathcal{Y}\times\mathcal{Y}$. For each $j\in[d]$, define V_j to be the set of labels which support the edges in E_j, that is,

$$V_j=\{y\in\mathcal{Y}:\exists r \text{ s.t. } (r,y)\in E_j\vee(y,r)\in E_j\} \quad . \tag{2}$$

We further require that $\mathbf{E}(\gamma)$ satisfies the following conditions,

1. For each $j\in[d]$ and for each $(r,s)\in E_j$ we have $\gamma_r>\gamma_s$.
2. For each $i\neq j\in[d]$ we have $E_i\cap E_j=\emptyset$.
3. For each $j\in[d]$, the sub-graph defined by (V_j,E_j) is a complete bipartite graph. That is, there exists two sets A and B, such that $A\cap B=\emptyset$, $V_j=A\cup B$, and $E_j=A\times B$.

In Fig. 2 we illustrate a few possible decompositions into bipartite graphs for a given label-ranking.

The loss of each sub-graph (V_j,E_j) is defined as the maximum over the losses of the pairs belonging to the sub-graph. In order to add some flexibility we also allow different sub-graphs to have different contribution to the loss. We do so by associating a weight σ_j with each sub-graph. The general form of our loss is therefore,

$$\ell(\mathbf{f}(\mathbf{x}),\gamma) \;=\; \sum_{j=1}^{d}\sigma_j\max_{(r,s)\in E_j}\ell_{r,s}(\mathbf{f}(\mathbf{x}),\gamma) \quad , \tag{3}$$

where each $\sigma_j\in\mathbb{R}_+$ is a non-negative weight. The weights σ_j can be used to associate importance values with each sub-graph (V_j,E_j) and to facilitate different notions of losses. For example, in multilabel classification problems, each instance is associated with a set of relevant labels which come from a predefined set $\mathcal{Y}$. The multilabel classification problem is a special case of the label ranking problem discussed in this paper and can be realized by setting $\gamma_r=1$ if the r'th label is relevant and otherwise defining $\gamma_r=0$. Thus, the feedback graph itself is of a bipartite form. Its edges are from $A\times B$ where A consists of all the relevant labels and B of the irrelevant ones. If we decide to set $\mathbf{E}(\gamma)$ to contain the single set $A\times B$ and define $\sigma_1=1$ then $\ell(\mathbf{f}(\mathbf{x}),\gamma)$ amounts to the *maximum* value of $\ell_{r,s}$ over pairs of edges in $A\times B$. Thus, the loss of this decomposition distills to the worst loss suffered over all pairs of comparable labels. Alternatively, we can set $\mathbf{E}(\gamma)$ to consist of all the sets $\{(r,s)\}$ for each $(r,s)\in A\times B$ and define $\sigma_j=1/|\mathbf{E}(\gamma)|$. In this case the total loss $\ell(\mathbf{f}(\mathbf{x}),\gamma)$ is the *average* value of $\ell_{r,s}$ over the edges in $A\times B$. Clearly, one can devise decompositions of $\mathbf{E}(\gamma)$ which are neither all pairs of edges nor a singleton including all edges. We

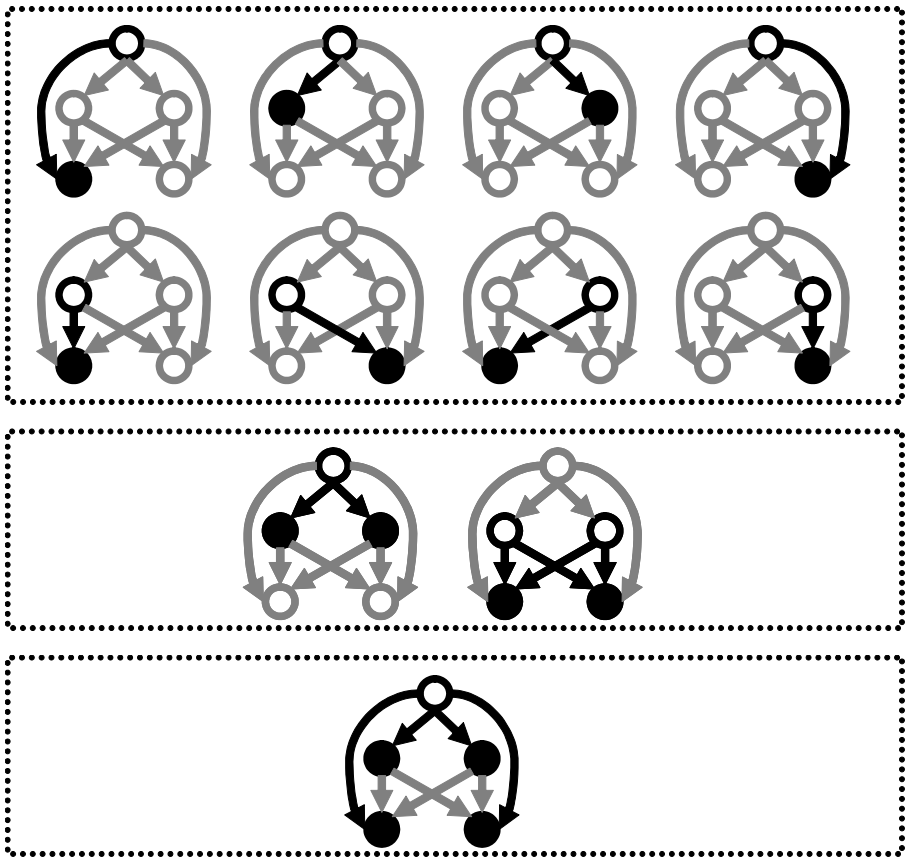


Figure 2: Three possible decompositions into complete bipartite sub-graphs of the graph from Fig. 1. Top: all-pairs decomposition; Middle: all adjacent layers; Bottom: top layer versus the rest of the layers. The edges and vertices participating in each sub-graph are depicted in black while the rest are presented in gray. In each graph the nodes constituting the set A are designated by black circles while for the nodes in B by filled black circles.

can thus capture different notions of losses for label ranking functions with multitude schemes for casting the relative importance of each subset (V_j, E_j).

Equipped with the loss function given in Eq. (3) we now formally define our learning problem. As in most learning settings, we assume that there exists an unknown distribution D over $\mathcal{X} \times \mathbb{R}^k$ and that each example in our training set is identically and independently drawn from D. The ultimate goal is to learn a label ranking function $\mathbf{f}$ which entertains a small generalization loss, $\mathbb{E}_{(\mathbf{x},\gamma)\sim D}[\ell(\mathbf{f}(\mathbf{x}),\gamma)]$. Since the distribution is not known we use instead an empirical sample from D and encompass a penalty for excessively complex label-ranking functions. Generalizing the Support Vector Machine (SVM) paradigm, we define a constrained optimization problem, whose optimal solution would constitute our label-ranking function. The objective function we need to minimize is composed of two terms. The first is the empirical loss of the label-ranking function on the training set and the second is a penalty for complexity, often referred to as a regularization term. This term amounts to the sum of the squared norms of $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$. The trade-off between the regularization

term and the empirical loss term is controlled by a parameter C. The resulting optimization problem is,

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k} \ \frac{1}{2}\sum_{j=1}^{k}\|\mathbf{w}_j\|^2 + C\sum_{i=1}^{m}\ell(\mathbf{f}(\mathbf{x}^i),\boldsymbol{\gamma}^i) \ , \tag{4}$$

where $f_y(\mathbf{x}^i) = \mathbf{w}_y \cdot \mathbf{x}^i$. Note that the loss function in Eq. (3) can also be represented as the solution of the following optimization problem,

$$\begin{aligned} \ell(\mathbf{f}(\mathbf{x}),\boldsymbol{\gamma}) \ = \ & \min_{\boldsymbol{\xi}\in\mathbb{R}^d_+} \ \sum_{j=1}^{d}\sigma_j\,\xi_j \\ & \text{s.t. } \ \forall j\in[d],\ \forall (r,s)\in E_j,\ \ \mathbf{f}_r(\mathbf{x}) - \mathbf{f}_s(\mathbf{x}) \geq \gamma_r - \gamma_s - \xi_j \ , \end{aligned} \tag{5}$$

where $d = |\mathbf{E}(\boldsymbol{\gamma})|$. Thus, we can rewrite the optimization problem given in Eq. (4) as a quadratic optimization problem,

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\boldsymbol{\xi}} \quad & \frac{1}{2}\sum_{j=1}^{k}\|\mathbf{w}_j\|^2 + C\sum_{i=1}^{m}\sum_{j=1}^{|\mathbf{E}(\boldsymbol{\gamma}^i)|}\sigma_j\,\xi^i_j \\ \text{s.t. } & \forall i\in[m],\forall E_j\in\mathbf{E}(\boldsymbol{\gamma}^i),\ \ \forall (r,s)\in E_j,\quad \mathbf{w}_r\cdot\mathbf{x}^i - \mathbf{w}_s\cdot\mathbf{x}^i \geq \gamma^i_r - \gamma^i_s - \xi^i_j \\ & \forall i,j,\ \ \xi^i_j \geq 0 \ . \end{aligned} \tag{6}$$

To conclude this section, we would like to review the rationale for choosing an one-sided loss for each pair by casting a single inequality for each (r,s). It is fairly easy to define a two-sided loss for a pair by mimicking regression problems. Concretely, we could replace the definition of $\ell_{r,s}$ as given in Eq. (1) with the loss $|f_r(\mathbf{x}) - f_s(\mathbf{x}) - (\gamma_r - \gamma_s)|$. This loss penalizes for *any* deviation from the desired difference of $\gamma_r - \gamma_s$. Instead, our loss is one sided as it penalizes only for not achieving a lower-bound. This choice is more natural in ranking applications. For instance, suppose we need to induce a ranking over 4 labels where the target label ranking is $(-1,2,0,0)$. Assume that the predicted ranking is instead $(-5,3,0,0)$. In most ranking and search applications such a predicted ranking would be perceived as being right on target since the preferences it expresses over pairs are on par with the target ranking. Furthermore, in most ranking applications, overly demotion of the most irrelevant items and excessive promotion of the most relevant ones is perceived as beneficial rather than a deficiency. Put another way, the set of target values encode minimal margin requirements and over-achieving these margin requirements should not be penalized.

Related Work Various known supervised learning problems can be viewed as special cases of the label ranking setting described in this paper. First, note that when there are only two labels we obtain the original constrained optimization of support vector machines for binary classification (Cortes and Vapnik, 1995) with the bias term set to zero. In the binary case, our algorithm reduces to the SOR algorithm described in (Mangasarian and Musicant, 1999). The multiclass problem, in which the target is a single label $y \in \mathcal{Y}$, can also be derived from our setting by defining $\gamma_y = 1$ and $\gamma_r = 0$ for all $r \neq y$. A few special-purpose algorithms have been suggested to solve the multiclass SVM problems. The multiclass version of Weston and Watkins (1999) is obtained by defining $\mathbf{E}(\boldsymbol{\gamma}) = \{\{(y,r)\}\}_{r\neq y}$, that is, each subset consists of a single pair (y,r). The multiclass version of Crammer and Singer (2001) can be obtained by simply setting $\mathbf{E}(\boldsymbol{\gamma})$ to be a single set containing all the pairs (y,r) for $r \neq y$, namely $\mathbf{E}(\boldsymbol{\gamma}) = \{\{(y,1),\ldots,(y,y-1),(y,y+1),\ldots,(y,k)\}\}$.

While the learning algorithms from (Weston and Watkins, 1999) and (Crammer and Singer, 2001) are seemingly different, they can be solved using the same algorithmic infrastructure presented in this paper. Proceeding to more complex decision problems, the task of multilabel classification or ranking is concerned with predicting a set or relevant labels or ranking the labels in accordance to their relevance to the input instance. This problem was studied by several authors (Elisseeff and Weston, 2001; Crammer and Singer, 2002; Dekel et al., 2003). Among these studies, the work of Elisseeff and Weston (2001) is probably the closest to ours yet it is still a derived special case of our setting . Elisseeff and Weston focus on a feedback vector γ which constitutes a bipartite graph by itself and define a constrained optimization problem with a *separate* slack variable for each edge in the graph. Formally, each instance $\mathbf{x}$ is associated with a set of relevant labels denoted Y. As discussed in the example above, the multilabel categorization setting can thus be realized by defining $\gamma_r = 1$ for all $r \in Y$ and $\gamma_s = 0$ for all $s \notin Y$. The construction of Elisseeff and Weston can be recovered by defining $\mathbf{E}(\gamma) = \{\{(r,s)\}|\gamma_r > \gamma_s\}$. Our approach is substantially more general as it allows much richer and flexible ways to decompose the multilabel problem as well as more general label ranking problems.

3. Fast "Soft" Projections

In the previous section we introduced the learning apparatus. Our goal now is to derive and analyze an efficient algorithm for solving the label ranking problem. In addition to efficiency, we also require that the algorithm would be general and flexible so it can be used with *any* decomposition of the feedback according to $\mathbf{E}(\gamma)$. While the algorithm presented in this and the coming sections is indeed efficient and general, its derivation is rather complex. We therefore would like to present it in a bottom-up manner starting with a sub-problem which constitutes the main building block of the algorithm. In this sub-problem we assume that we have obtained a label-ranking function realized by the set $\mathbf{u}_1, \ldots, \mathbf{u}_k$ and the goal is to modify the ranking function so as to fit better a newly obtained example. To further simplify the derivation, we focus on the case where $\mathbf{E}(\gamma)$ contains a single complete bipartite graph whose set of edges are simply denoted by E. The end result is the following simplified constrained optimization problem,

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi} \quad & \frac{1}{2}\sum_{y=1}^{k} \|\mathbf{w}_y - \mathbf{u}_y\|^2 + C\xi \\ \text{s.t.} \quad & \forall (r,s) \in E, \;\; \mathbf{w}_r \cdot \mathbf{x} - \mathbf{w}_s \cdot \mathbf{x} \geq \gamma_r - \gamma_s - \xi \\ & \xi \geq 0 \;\; . \end{aligned} \tag{7}$$

Here $\mathbf{x} \in \mathcal{X}$ is a single instance and E is a set of edges which induces a complete bipartite graph.

The focus of this section is an efficient algorithm for solving Eq. (7). This optimization problem finds the set closest to $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ which approximately satisfies a system of linear constraints with a single slack (relaxation) variable ξ. Put another way, we can view the problem as the task of finding a relaxed projection of the set $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ onto the polyhedron defined by the set of linear constraints induced from E. We thus refer to this task as the soft projection. Our algorithmic solution, while being efficient, is rather detailed and its derivation consists of multiple complex steps. We therefore start with a high level overview of its derivation. We first derive a dual version of the problem defined by Eq. (7). Each variable in the dual problem corresponds to an edge in E. Thus, the total number of dual variables can be as large as $k^2/4$. We then introduce a new

and more compact optimization problem which has only k variables. We prove that the reduced problem nonetheless attains the same optimum as the original dual problem. This reduction is one of the two major steps in the derivation of an efficient soft projection procedure. We next show that the reduced problem can be decoupled into two simpler constrained optimization problems each of which corresponds to one layer in the bipartite graph induced by E. The two problems are tied by a single variable. We finally reach an efficient solution by showing that the optimal value of the coupling variable can be efficiently computed in $O(k\log(k))$ time. We recap our entire derivation by providing the pseudo-code of the resulting algorithm at the end of the section.

3.1 The Dual Problem

To start, we would like to note that the primal objective function is convex and all the primal constraints are linear. A necessary and sufficient condition for strong duality to hold in this case is that there exists a feasible solution to the primal problem (see for instance (Boyd and Vandenberghe, 2004)). A feasible solution can indeed obtained by simply setting $\mathbf{w}_y = \mathbf{0}$ for all y and defining $\xi = \max_{(r,s)\in E}(\gamma_r - \gamma_s)$. Therefore, strong duality holds and we can obtain a solution to the primal problem by finding the solution of its dual problem. To do so we first write the Lagrangian of the primal problem given in Eq. (7), which amounts to,

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\sum_{y=1}^{k}\|\mathbf{w}_y - \mathbf{u}_y\|^2 + C\xi + \sum_{(r,s)\in E}\tau_{r,s}\left(\gamma_r - \gamma_s - \xi + \mathbf{w}_s\cdot\mathbf{x} - \mathbf{w}_r\cdot\mathbf{x}\right) - \zeta\xi \\
&= \frac{1}{2}\sum_{y=1}^{k}\|\mathbf{w}_y - \mathbf{u}_y\|^2 + \xi\left(C - \sum_{(r,s)\in E}\tau_{r,s} - \zeta\right) + \sum_{(r,s)\in E}\tau_{r,s}\left(\gamma_r - \mathbf{w}_r\cdot\mathbf{x} - \gamma_s + \mathbf{w}_s\cdot\mathbf{x}\right) \ ,
\end{aligned}$$

where $\tau_{r,s} \geq 0$ for all $(r,s) \in E$ and $\zeta \geq 0$. To derive the dual problem we now can use the strong duality. We eliminate the primal variables by minimizing the Lagrangian with respect to its primal variables. First, note that the minimum of the term $\xi(C - \sum_{(r,s)\in E}\tau_{r,s} - \zeta)$ with respect to ξ is zero whenever $C - \sum_{(r,s)\in E}\tau_{r,s} - \zeta = 0$. If however $C - \sum_{(r,s)\in E}\tau_{r,s} - \zeta \neq 0$ then this term can be made to approach $-\infty$. Since we need to maximize the dual we can rule out the latter case and pose the following constraint on the dual variables,

$$C - \sum_{(r,s)\in E}\tau_{r,s} - \zeta \ = \ 0 \ . \tag{8}$$

Next, recall our assumption that E induces a complete bipartite graph (V,E) (see also Eq. (2)). Therefore, there exists two sets A and B such that $A \cap B = \emptyset$, $V = A \cup B$, and $E = A \times B$. Using the definition of the sets A and B we can rewrite the last sum of the Lagrangian as,

$$\begin{aligned}
&\sum_{r\in A, s\in B}\tau_{r,s}\left(\gamma_r - \mathbf{w}_r\cdot\mathbf{x} - \gamma_s + \mathbf{w}_s\cdot\mathbf{x}\right) = \\
&\qquad \sum_{r\in A}\left(\gamma_r - \mathbf{w}_r\cdot\mathbf{x}\right)\sum_{s\in B}\tau_{r,s} \ - \ \sum_{s\in B}\left(\gamma_s - \mathbf{w}_s\cdot\mathbf{x}\right)\sum_{r\in A}\tau_{r,s} \ .
\end{aligned}$$

Eliminating the remaining primal variables $\mathbf{w}_1,\ldots,\mathbf{w}_k$ is done by differentiating the Lagrangian with respect to $\mathbf{w}_r$ for all $r \in [k]$ and setting the result to zero. For all $y \in A$, the above gives the set of constraints,

$$\nabla_{\mathbf{w}_y}\mathcal{L} \ = \ \mathbf{w}_y - \mathbf{u}_y - \left(\sum_{s\in B}\tau_{y,s}\right)\mathbf{x} \ = \ 0 \ . \tag{9}$$

Similarly, for $y \in B$ we get that,

$$\nabla_{\mathbf{w}_y} \mathcal{L} \;=\; \mathbf{w}_y - \mathbf{u}_y + \left(\sum_{r \in A} \tau_{r,y} \right) \mathbf{x} \;=\; 0 \;\; . \tag{10}$$

Finally, we would like to note that for any label $y \notin A \cup B$ we get that $\mathbf{w}_y - \mathbf{u}_y = 0$. Thus, we can omit all such labels from our derivation. Summing up, we get that,

$$\mathbf{w}_y \;=\; \begin{cases} \mathbf{u}_y + \left(\sum_{s \in B} \tau_{y,s}\right) \mathbf{x} & y \in A \\ \mathbf{u}_y - \left(\sum_{r \in A} \tau_{r,y}\right) \mathbf{x} & y \in B \\ \mathbf{u}_y & \text{otherwise} \end{cases} \;\; . \tag{11}$$

Plugging Eq. (11) and Eq. (8) into the Lagrangian and rearranging terms give the following dual objective function,

$$\begin{aligned} D(\tau) \quad = \quad & -\frac{1}{2} \|\mathbf{x}\|^2 \sum_{y \in A} \left(\sum_{s \in B} \tau_{y,s} \right)^2 - \frac{1}{2} \|\mathbf{x}\|^2 \sum_{y \in B} \left(\sum_{r \in A} \tau_{r,y} \right)^2 \\ & + \sum_{y \in A} (\gamma_y - \mathbf{u}_y \cdot \mathbf{x}) \sum_{s \in B} \tau_{y,s} - \sum_{y \in B} (\gamma_y - \mathbf{u}_y \cdot \mathbf{x}) \sum_{r \in A} \tau_{r,y} \;\; . \end{aligned} \tag{12}$$

In summary, the resulting dual problem is,

$$\max_{\tau \in \mathbb{R}_+^{|E|}} D(\tau) \qquad \text{s.t.} \quad \sum_{(r,s) \in E} \tau_{r,s} \leq C \;\; . \tag{13}$$

3.2 Reparametrization of the Dual Problem

Each dual variable $\tau_{r,s}$ corresponds to an edge in E. Thus, the number of dual variables may be as large as $k^2/4$. However, the dual objective function depends only on sums of variables $\tau_{r,s}$. Furthermore, each primal vector $\mathbf{w}_y$ also depends on sums of dual variables (see Eq. (11)). We exploit these useful properties to introduce an equivalent optimization of a smaller size with only k variables. We do so by defining the following variables,

$$\forall y \in A, \;\; \alpha_y = \sum_{s \in B} \tau_{y,s} \qquad \text{and} \qquad \forall y \in B, \;\; \beta_y = \sum_{r \in A} \tau_{r,y} \;\; . \tag{14}$$

The primal variables $\mathbf{w}_y$ from Eq. (11) can be rewritten using α_y and β_y as follows,

$$\mathbf{w}_y \;=\; \begin{cases} \mathbf{u}_y + \alpha_y \mathbf{x} & y \in A \\ \mathbf{u}_y - \beta_y \mathbf{x} & y \in B \\ \mathbf{u}_y & \text{otherwise} \end{cases} \;\; . \tag{15}$$

Overloading our notation and using $D(\alpha, \beta)$ to denote dual objective function in terms of α and β, we can rewrite the dual objective of Eq. (12) as follows,

$$D(\alpha, \beta) \;= -\frac{1}{2} \|\mathbf{x}\|^2 \left(\sum_{y \in A} \alpha_y^2 + \sum_{y \in B} \beta_y^2 \right) + \sum_{y \in A} (\gamma_y - \mathbf{u}_y \cdot \mathbf{x}) \, \alpha_y - \sum_{y \in B} (\gamma_y - \mathbf{u}_y \cdot \mathbf{x}) \, \beta_y \;\; . \tag{16}$$

Note that the definition of α_y and β_y from Eq. (14) implies that α_y and β_y are non-negative. Furthermore, by construction we also get that,

$$\sum_{y\in A}\alpha_y \;=\; \sum_{y\in B}\beta_y \;=\; \sum_{(r,s)\in E}\tau_{r,s} \;\leq\; C \;. \tag{17}$$

In summary, we have obtained the following constrained optimization problem,

$$\max_{\alpha\in\mathbb{R}_+^{|A|},\;\beta\in\mathbb{R}_+^{|B|}} D(\alpha,\beta) \quad \text{s.t.} \quad \sum_{y\in A}\alpha_y = \sum_{y\in B}\beta_y \leq C \;. \tag{18}$$

We refer to the above optimization problem as the *reduced* problem since it encompasses at most $k = |V|$ variables. In appendix A we show that the reduced problem and the original dual problem from Eq. (13) are equivalent. The end result is the following corollary.

Corollary 1 *Let $(\alpha^\star,\beta^\star)$ be the optimal solution of the reduced problem in Eq. (18). Define $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$ as in Eq. (15). Then, $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$ is the optimal solution of the soft projection problem defined by Eq. (7).*

We now move our focus to the derivation of an efficient algorithm for solving the reduced problem. To make our notation easy to follow, we define $p = |A|$ and $q = |B|$ and construct two vectors $\mu \in \mathbb{R}^p$ and $\nu \in \mathbb{R}^q$ such that for each $a \in A$ there is an element $(\gamma_a - \mathbf{u}_a\cdot\mathbf{x})/\|\mathbf{x}\|^2$ in μ and for each $b \in B$ there is an element $-(\gamma_b - \mathbf{u}_b\cdot\mathbf{x})/\|\mathbf{x}\|^2$ in ν. The reduced problem can now be rewritten as,

$$\begin{aligned}\min_{\alpha\in\mathbb{R}_+^p,\beta\in\mathbb{R}_+^q} \quad & \frac{1}{2}\|\alpha-\mu\|^2 + \frac{1}{2}\|\beta-\nu\|^2 \\ \text{s.t.} \quad & \sum_{i=1}^{p}\alpha_i \;=\; \sum_{j=1}^{q}\beta_j \;\leq\; C \;.\end{aligned} \tag{19}$$

3.3 Decoupling the Reduced Optimization Problem

In the previous section we showed that the soft projection problem given by Eq. (7) is equivalent to the reduced optimization problem of Eq. (19). Note that the variables α and β are tied together through a single equality constraint $\|\alpha\|_1 = \|\beta\|_1$. We represent this coupling of α and β by rewriting the optimization problem in Eq. (19) as,

$$\min_{z\in[0,C]} \quad g(z;\mu) + g(z;\nu) \;,$$

where

$$g(z;\mu) \;=\; \min_{\alpha}\frac{1}{2}\|\alpha-\mu\|^2 \quad \text{s.t.} \quad \sum_{i=1}^{p}\alpha_i \;=\; z \;,\;\; \alpha_i \geq 0 \;, \tag{20}$$

and similarly

$$g(z;\nu) \;=\; \min_{\beta}\frac{1}{2}\|\beta-\nu\|^2 \quad \text{s.t.} \quad \sum_{j=1}^{q}\beta_j \;=\; z \;,\;\; \beta_j \geq 0 \;. \tag{21}$$

The function $g(z;\cdot)$ takes the same functional form whether we use μ or ν as the second argument. We therefore describe our derivation in terms of $g(z;\mu)$. Clearly, the same derivation is also applicable to $g(z;\nu)$. The Lagrangian of $g(z;\mu)$ is,

$$\mathcal{L} \;=\; \frac{1}{2}\|\alpha-\mu\|^2+\theta\left(\sum_{i=1}^{p}\alpha_i-z\right)-\zeta\cdot\alpha \;\;,$$

where $\theta\in\mathbb{R}$ is a Lagrange multiplier and $\zeta\in\mathbb{R}^p_+$ is a vector of non-negative Lagrange multipliers. Differentiating with respect to α_i and comparing to zero gives the following KKT condition,

$$\frac{d\mathcal{L}}{d\alpha_i} \;=\; \alpha_i-\mu_i+\theta-\zeta_i \;=\; 0 \;\;.$$

The complementary slackness KKT condition implies that whenever $\alpha_i>0$ we must have that $\zeta_i=0$. Thus, if $\alpha_i>0$ we get that,

$$\alpha_i \;=\; \mu_i-\theta+\zeta_i \;=\; \mu_i-\theta \;\;. \tag{22}$$

Since all the non-negative elements of the vector α are tied via a single variable we would have ended with a much simpler problem had we known the indices of these elements. On a first sight, this task seems difficult as the number of potential subsets of α is clearly exponential in the dimension of α. Fortunately, the particular form of the problem renders an efficient algorithm for identifying the non-zero elements of α. The following lemma is a key tool in deriving our procedure for identifying the non-zero elements.

Lemma 2 *Let α be the optimal solution to the minimization problem in Eq. (20). Let s and j be two indices such that $\mu_s>\mu_j$. If $\alpha_s=0$ then α_j must be zero as well.*

Proof Assume by contradiction that $\alpha_s=0$ yet $\alpha_j>0$. Let $\tilde{\alpha}\in\mathbb{R}^k$ be a vector whose elements are equal to the elements of α except for $\tilde{\alpha}_s$ and $\tilde{\alpha}_j$ which are interchanged, that is, $\tilde{\alpha}_s=\alpha_j$, $\tilde{\alpha}_j=\alpha_s$, and for every other $r\notin\{s,j\}$ we have $\tilde{\alpha}_r=\alpha_r$. It is immediate to verify that the constraints of Eq. (20) still hold. In addition we have that,

$$\|\alpha-\mu\|^2-\|\tilde{\alpha}-\mu\|^2 \;=\; \mu_s^2+(\alpha_j-\mu_j)^2-(\alpha_j-\mu_s)^2-\mu_j^2 \;=\; 2\alpha_j(\mu_s-\mu_j) \;>\; 0 \;\;.$$

Therefore, we obtain that $\|\alpha-\mu\|^2>\|\tilde{\alpha}-\mu\|^2$, which contradicts the fact that α is the optimal solution. ■

Let I denote the set $\{i\in[p]:\alpha_i>0\}$. The above lemma gives a simple characterization of the set I. Let us reorder the μ such that $\mu_1\geq\mu_2\geq\ldots\geq\mu_p$. Simply put, Lemma 2 implies that after the reordering, the set I is of the form $\{1,\ldots,\rho\}$ for some $1\leq\rho\leq p$. Had we known ρ we could have simply use Eq. (22) and get that

$$\sum_{i=1}^{p}\alpha_i=\sum_{i=1}^{\rho}\alpha_i=\sum_{i=1}^{\rho}(\mu_i-\theta)=z \;\;\Rightarrow\;\; \theta=\frac{1}{\rho}\left(\sum_{i=1}^{\rho}\mu_i-z\right) \;\;.$$

In summary, given ρ we can summarize the optimal solution for α as follows,

$$\alpha_i \;=\; \begin{cases} \mu_i-\dfrac{1}{\rho}\left(\displaystyle\sum_{i=1}^{\rho}\mu_i-z\right) & i\leq\rho \\ 0 & i>\rho \end{cases} \;\;. \tag{23}$$

We are left with the problem of finding the optimal value of ρ. We could simply enumerate all possible values of ρ in $[p]$, for each possible value compute α as given by Eq. (23), and then choose the value for which the objective function ($\|\alpha-\mu\|^2$) is the smallest. While this procedure can be implemented quite efficiently, the following lemma provides an even simpler solution once we reorder the elements of μ to be in a non-increasing order.

Lemma 3 *Let α be the optimal solution to the minimization problem given in Eq. (20) and assume that $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_p$. Then, the number of strictly positive elements in α is,*

$$\rho(z,\mu) \;=\; \max\left\{ j \in [p] \;:\; \mu_j - \frac{1}{j}\left(\sum_{r=1}^{j}\mu_r - z\right) > 0 \right\} \;.$$

The proof of this technical lemma is deferred to the appendix.

Had we known the optimal value of z, i.e. the argument attaining the minimum of $g(z;\mu)+g(z;\nu)$ we could have calculated the optimal dual variables $\alpha^\star$ and $\beta^\star$ by first finding $\rho(z,\mu)$ and $\rho(z,\nu)$ and then finding α and β using Eq. (23). This is a classical chicken-and-egg problem: we can easily calculate the optimal solution given some side information, however, obtaining the side information seems as difficult as finding the optimal solution. One option is to perform a search over an ε-net of values for z in $[0,C]$. For each candidate value for z from the ε-net we can find α and β and then choose the value which attains the lowest objective value ($g(z;\mu)+g(z;\nu)$). While this approach may be viable in many cases, it is still quite time consuming. To our rescue comes the fact that $g(z;\mu)$ and $g(z;\nu)$ entertain a very special structure. Rather than enumerating over all possible values of z we need to check at most $k+1$ possible values for z. To establish the last part of our efficient algorithm which performs this search for the optimal value of z we need the following theorem. The theorem is stated with μ but, clearly, it also holds for ν .

Theorem 4 *Let $g(z;\mu)$ be as defined in Eq. (20). For each $i \in [p]$, define*

$$z_i = \sum_{r=1}^{i}\mu_r - i\mu_i \;.$$

Then, for each $z \in [z_i, z_{i+1}]$ the function $g(z;\mu)$ is equivalent to the following quadratic function,

$$g(z;\mu) \;=\; \frac{1}{i}\left(\sum_{r=1}^{i}\mu_r - z\right)^2 + \sum_{r=i+1}^{p}\mu_r^2 \;.$$

Moreover, g is continuous, continuously differentiable, and convex in $[0,C]$.

The proof of this theorem is also deferred to the appendix. The good news that the theorem carries is that $g(z;\mu)$ and $g(z;\nu)$ are convex and therefore their sum is also convex. Furthermore, the function $g(z;\cdot)$ is piecewise quadratic and the points where it changes from one quadratic function to another are simple to compute. We refer to these points as knots. In the next sub-section we exploit the properties of the function g to devise an efficient procedure for finding the optimal value of z and from there the road to the optimal dual variables is clear and simple.

INPUT: instance $\mathbf{x} \in \mathcal{X}$; target ranking γ ; sets A, B

current prototypes $\mathbf{u}^1, \ldots, \mathbf{u}^k$; regularization parameter C

MARGINS:

$$\mu = \text{sort}\left\{ (\gamma_a - \mathbf{u}^a \cdot \mathbf{x}) / \|\mathbf{x}\|^2 \mid a \in A \right\}$$
$$\nu = \text{sort}\left\{ (\mathbf{u}^b \cdot \mathbf{x} - \gamma_b) / \|\mathbf{x}\|^2 \mid b \in B \right\}$$

KNOTS:

$$\forall i \in [p] : \; z_i = \textstyle\sum_{r=1}^{i} \mu_r - i\mu_i \qquad \forall j \in [q] : \; \tilde{z}_j = \sum_{s=1}^{j} \nu_s - j\nu_j$$
$$\mathcal{Q} = \{z_i : z_i < C\} \cup \left\{\tilde{z}_j : \tilde{z}_j < C\right\} \cup \{C\}$$

INTERVALS:

$$\forall z \in \mathcal{Q} : \quad R(z) = |\{z_i : z_i \leq z\}| \quad ; \quad S(z) = \left|\{\tilde{z}_j : \tilde{z}_j \leq z\}\right|$$
$$\forall z \in \mathcal{Q} : \quad N(z) = \min\{z' \in \mathcal{Q} : z' > z\} \cup \{C\}$$

LOCAL MIN:

$$O(z) \;=\; \left(S(z) \sum_{r=1}^{R(z)} \mu_r + R(z) \sum_{r=1}^{S(z)} \nu_r \right) / \left(R(z) + S(z)\right)$$

GLOBAL MIN:

If $(\exists z \in \mathcal{Q} \;\; \text{s.t.} \;\; O(z) \in [z, N(z)])$ **Then**

$z^\star = O(z) \;\; ; \;\; i^\star = R(z) \;\; ; \;\; j^\star = S(z)$

Else If $(\mu_1 + \nu_1 \leq 0)$

$z^\star = 0 \;\; ; \;\; i^\star = 1 \;\; ; \;\; j^\star = 1$

Else

$z^\star = C \;\; ; \;\; i^\star = R(C) \;\; ; \;\; j^\star = S(C)$

DUAL'S AUXILIARIES:

$$\theta_\alpha = \frac{1}{i^\star} \left(\sum_{r=1}^{i^\star} \mu_r - z^\star \right) \;\; ; \;\; \theta_\beta = \frac{1}{j^\star} \left(\sum_{r=1}^{j^\star} \nu_r - z^\star \right)$$

OUTPUT:

$$\forall a \in A : \; \alpha_a \;=\; \left(\tfrac{\gamma_a - u_a \cdot \mathbf{x}}{\|\mathbf{x}\|^2} - \theta_\alpha \right)_+ \;\; \text{and} \;\; \mathbf{w}_a \;=\; \mathbf{u}_a + \alpha_a \mathbf{x}$$
$$\forall b \in B : \; \beta_b \;=\; \left(\tfrac{u_b \cdot \mathbf{x} - \gamma_b}{\|\mathbf{x}\|^2} - \theta_\beta \right)_+ \;\; \text{and} \;\; \mathbf{w}_b \;=\; \mathbf{u}_b - \beta_b \mathbf{x}$$

Figure 3: Pseudo-code of the soft-projection onto polyhedra (SOPOPO) algorithm.

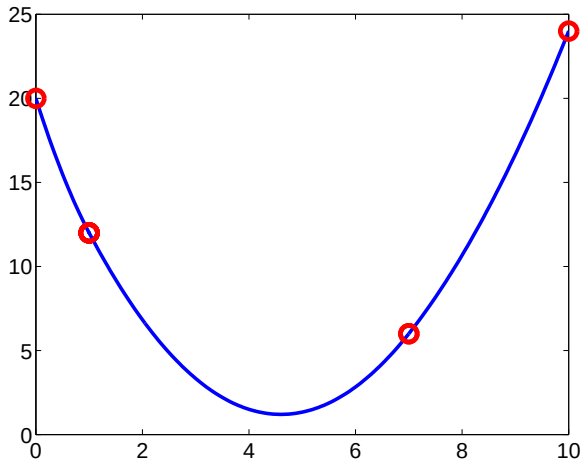


Figure 4: An illustration of the function $g(z;\mu)+g(z;\nu)$. The vectors μ and ν are constructed from, $\gamma = (1,2,3,4,5,6)$, $\mathbf{u}\cdot\mathbf{x} = (2,3,5,1,6,4)$, $A = \{4,5,6\}$, and $B = \{1,2,3\}$.

3.4 Putting it All Together

Due to the strict convexity of $g(z;\mu)+g(z;\nu)$ its minimum is unique and well defined. Therefore, it suffices to search for a seemingly *local* minimum over all the sub-intervals in which the objective function is equivalent to a quadratic function. If such a local minimum point is found it is guaranteed to be the global minimum. Once we have the value of z which constitutes the global minimum we can decouple the optimization problems for α and β and quickly find the optimal solution. There is though one last small obstacle: the objective function is the sum of two piecewise quadratic functions. We therefore need to efficiently go over the *union* of the knots derived from μ and ν. We now summarize the full algorithm for finding the optimum of the dual variables and wrap up with its pseudo-code.

Given μ and ν we find the sets of knots for each vector, take the union of the two sets, and sort the set in an ascending order. Based on the theorems above, it follows immediately that each interval between two consecutive knots in the union is also quadratic. Since $g(z;\mu)+g(z;\nu)$ is convex, the objective function in each interval can be characterized as falling into one of two cases. Namely, the objective function is either monotone (increasing or decreasing) or it attains its unique global minimum inside the interval. In the latter case the objective function clearly decreases, reaches the optimum where its derivative is zero, and then increases. See also Fig. 4 for an illustration. If the objective function is monotone in all of the intervals then the minimum is obtained at one of the boundary points $z = 0$ or $z = C$. Otherwise, we simply need to identify the interval bracketing the global minimum and then find the optimal value of z by finding the minimizer of the quadratic function associated with the interval. For instance, in Fig. 4 the minimum is attained just below 5 at the interval defined by the second and third knots. If C is, say, 10 then the optimal value for z coincides with the minimum below 5. If however C lies to the left of the minimum, say at 3, then the optimum of z is at 3. We now formally recap the entire procedure.

We utilize the following notation. For each $i \in [p]$, define the knots derived from μ

$$z_i = \sum_{r=1}^{i} \mu_r - i\mu_i \ ,$$

and similarly, for each $j \in [q]$ define

$$\tilde{z}_j = \sum_{r=1}^{j} \nu_r - j\nu_j \ .$$

From Lemma 4 we know that $g(z;\mu)$ is quadratic in each segment $[z_i, z_{i+1})$ and $g(z;\nu)$ is quadratic in each segment $[\tilde{z}_j, \tilde{z}_{j+1})$. Therefore, as already argued above, the function $g(z;\mu)+g(z;\nu)$ is also piecewise quadratic in $[0,C]$ and its knots are the points in the set,

$$\mathcal{Q} = \{z_i : z_i < C\} \cup \{\tilde{z}_j : \tilde{z}_j < C\} \cup \{C\} \ .$$

For each knot $z \in \mathcal{Q}$, we denote by $N(z)$ its consecutive knot in $\mathcal{Q}$, that is,

$$N(z) = \min \ \left(\{z' \in \mathcal{Q} : z' > z\} \cup \{C\}\right) \ .$$

We also need to know for each knot how many knots precede it. Given a knot z we define

$$R(z) = |\{z_i : z_i \le z\}| \ \text{ and } \ S(z) = |\{\tilde{z}_i : \tilde{z}_i \le z\}| \ .$$

Using the newly introduced notation we can find for a given value z its bracketing interval, $z \in [z', N(z')]$. From Thm. 4 we get that the value of the dual objective function at z is,

$$g(z;\mu)+g(z;\nu) \ = \\ \frac{1}{R(z')}\left(\sum_{r=1}^{R(z')} \mu_r - z\right)^2 + \sum_{r=R(z')+1}^{p} \mu_r^2 + \frac{1}{S(z')}\left(\sum_{r=1}^{S(z')} \nu_r - z\right)^2 + \sum_{r=S(z')+1}^{p} \nu_r^2 \ .$$

The unique minimum of the quadratic function above is attained at the point

$$O(z') = \left(S(z') \sum_{i=1}^{R(z')} \mu_i + R(z') \sum_{i=1}^{S(z')} \nu_i\right) / \left(R(z')+S(z')\right) \ .$$

Therefore, if $O(z') \in [z', N(z')]$, then the global minimum of the dual objective function is attained at $O(z')$. Otherwise, if no such interval exists, the optimum is either at $z = 0$ or at $z = C$. The minimum is achieved at $z = 0$ iff the derivative of the objective function at $z = 0$ is non-negative, namely, $-\mu_1 - \nu_1 \ge 0$. In this case, the optimal solution is $\alpha = \mathbf{0}$ and $\beta = \mathbf{0}$ which implies that $\mathbf{w}_r = \mathbf{u}_r$ for all r. If on the other hand $-\mu_1 - \nu_1 < 0$ then the optimum is attained at $z = C$. The skeleton of the pseudo-code for the fast projection algorithm is given in Fig. 3. The most expensive operation performed by the algorithm is the sorting of μ and ν. Since the sum of the dimensions of these vectors is k the time complexity of the algorithm is $\Theta(k\log(k))$.

4. From a Single Projection to Multiple Projections

We now describe the algorithm for solving the original batch problem defined by Eq. (6) using the SOPOPO algorithm as its core. We would first like to note that the general batch problem can also be viewed as a soft projection problem. We can cast the batch problem as finding the set of vectors $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$ which is closest to k zero vectors $\{\mathbf{0},\ldots,\mathbf{0}\}$ while approximately satisfying a set of systems of linear constraints where each system is associated with an independent relaxation variable. Put another way, we can view the full batch optimization problem as the task of finding a relaxed projection of the set $\{\mathbf{0},\ldots,\mathbf{0}\}$ onto multiple polyhedra each of which is defined via a set of linear constraints induced by a single sub-graph $E_j \in \mathbf{E}(\gamma^i)$. We thus refer to this task as the soft-projection onto multiple polyhedra. We devise an iterative algorithm which solves the batch problem by successively calling to the SOPOPO algorithm from Fig. 3. We describe and analyze the algorithm for a slightly more general constrained optimization which results in a simplified notation. We start with the presentation of our original formulation as an instance of the generalized problem.

To convert the problem in Eq. (6) to a more general form, we assume without loss of generality that $|\mathbf{E}(\gamma^i)| = 1$ for all $i \in [m]$. We refer to the single set in $\mathbf{E}(\gamma^i)$ as E^i. This assumption does not pose a limitation since in the case of multiple decompositions, $\mathbf{E}(\gamma^i) = \{E_1,\ldots,E_d\}$, we can replace the ith example with d pseudo-examples: $\{(\mathbf{x}^i,E_1),\ldots,(\mathbf{x}^i,E_d)\}$. Using this assumption, we can rewrite the optimization problem of Eq. (6) as follows,

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi} \quad & \frac{1}{2}\sum_{r=1}^{k}\|\mathbf{w}_r\|^2 \;+\; \sum_{i=1}^{m} C_i\xi^i \\ \text{s.t.} \quad & \forall i \in [m], \forall (r,s) \in E^i, \quad \mathbf{w}_r\cdot\mathbf{x}^i - \mathbf{w}_s\cdot\mathbf{x}^i \geq \gamma^i_r - \gamma^i_s - \xi^i \\ & \forall i, \;\; \xi^i \geq 0 \;\; , \end{aligned} \tag{24}$$

where $C_i = C\sigma^i$ is the weight of the ith slack variable. To further simplify Eq. (24), we use $\bar{\mathbf{w}} \in \mathbb{R}^{nk}$ to denote the concatenation of the vectors $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$. In addition, we associate an index, denoted j, with each $(r,s) \in E^i$ and define $\mathbf{a}^{i,j} \in \mathbb{R}^{nk}$ to be the vector,

$$\mathbf{a}^{i,j} = (\; \underbrace{\mathbf{0}}_{\text{1st block}} \;,\; \ldots \;,\; \mathbf{0} \;,\; \underbrace{\mathbf{x}^i}_{r\text{th block}} \;,\; \mathbf{0} \;,\; \ldots \;,\; \mathbf{0} \;,\; \underbrace{-\mathbf{x}^i}_{s\text{th block}} \;,\; \mathbf{0} \;,\; \ldots \;,\; \underbrace{\mathbf{0}}_{k\text{th block}} \;) \;. \tag{25}$$

We also define $b^{i,j} = \gamma^i_r - \gamma^i_s$. Finally, we define $k_i = |E^i|$. Using the newly introduced notation we can rewrite Eq. (24) as follows,

$$\begin{aligned} \min_{\bar{\mathbf{w}},\xi} \quad & \frac{1}{2}\|\bar{\mathbf{w}}\|^2 + \sum_{i=1}^{m} C_i\xi^i \\ \text{s.t.} \quad & \forall i \in [m], \forall j \in [k_i], \;\; \bar{\mathbf{w}}\cdot\mathbf{a}^{i,j} \geq b^{i,j} - \xi^i \\ & \xi^i \geq 0 \;\; . \end{aligned} \tag{26}$$

Our goal is to derive an iterative algorithm for solving Eq. (26) based on a procedure for solving a single soft-projection which takes the form,

$$\begin{aligned} \min_{\bar{\mathbf{w}},\xi^i} \quad & \frac{1}{2}\|\bar{\mathbf{w}} - \mathbf{u}\|^2 + C_i\xi^i \\ \text{s.t.} \quad & \forall j \in [k_i], \;\; \bar{\mathbf{w}}\cdot\mathbf{a}^{i,j} \geq b^{i,j} - \xi^i \\ & \xi^i \geq 0 \;\; . \end{aligned} \tag{27}$$

By construction, an algorithm for solving the more general problem defined in Eq. (26) would also solve the more specific problem defined by Eq. (6).

The rest of the section is organized as follows. We first derive the dual of the problem given in Eq. (26). We then describe an iterative algorithm which on each iteration performs a single soft-projection and present a pseudo-code of the iterative algorithm tailored for the specific label-ranking problem of Eq. (6). Finally, we analyze the convergence of the suggested iterative algorithm.

4.1 The Dual Problem

First, note that the primal objective function of the general problem is convex and all the primal constraints are linear. Therefore, using the same arguments as in Sec. 3.1 it is simple to show that strong duality holds and a solution to the primal problem can be obtained from the solution of its dual problem. To derive the dual problem, we first write the Lagrangian,

$$\mathcal{L} = \frac{1}{2}\|\bar{\mathbf{w}}\|^2 + \sum_{i=1}^{m} C_i \xi^i + \sum_{i=1}^{m}\sum_{j=1}^{k_i} \lambda_{i,j}\left(b^{i,j} - \xi^i - \bar{\mathbf{w}}\cdot\mathbf{a}^{i,j}\right) - \sum_{i=1}^{m}\zeta_i\xi^i \ ,$$

where $\lambda_{i,j}$ and ζ_i are non-negative Lagrange multipliers. Taking the derivative of $\mathcal{L}$ with respect to $\bar{\mathbf{w}}$ and comparing it to zero gives,

$$\bar{\mathbf{w}} \ = \ \sum_{i=1}^{m}\sum_{j=1}^{k_i} \lambda_{i,j}\,\mathbf{a}^{i,j} \ . \tag{28}$$

As in the derivation of the dual objective function for a single soft projection, we get that the following must hold at the optimum,

$$\forall i \in [m], \quad \sum_{j=1}^{k_i} \lambda_{i,j} - C_i - \zeta_i = 0 \ . \tag{29}$$

Since $\lambda_{i,j}$ and ζ_i are non-negative Lagrange multipliers we get that the set of feasible solutions of the dual problem is,

$$S \ = \ \left\{ \lambda \ \middle| \ \forall i, \ \sum_{j=1}^{k_i} \lambda_{i,j} \ \le \ C_i \ \text{ and } \ \forall i,j, \ \lambda_{i,j} \ge 0 \right\} \ .$$

Using Eq. (28) and Eq. (29) to further rewrite the Lagrangian gives the dual objective function,

$$D(\lambda) \ = \ -\frac{1}{2}\left\| \sum_{i=1}^{m}\sum_{j=1}^{k_i} \lambda_{i,j}\,\mathbf{a}^{i,j} \right\|^2 + \sum_{i=1}^{m}\sum_{j=1}^{k_i} \lambda_{i,j}\, b^{i,j} \ .$$

The dual of the problem defined in Eq. (26) is therefore,

$$\max_{\lambda \in S} D(\lambda) \ . \tag{30}$$

INPUT: training set $\{(\mathbf{x}^i, \gamma^i)\}_{i=1}^m$; decomposition function $\mathbf{E}(\gamma)$; regularization parameter C

INITIALIZE:

$\forall i \in [m], A_j \times B_j \in \mathbf{E}(\gamma^i),\ (a,b) \in A_j \times B_j,$ set $\alpha_a^{i,j} = 0,\ \beta_b^{i,j} = 0$

$\forall r \in [k]$, set $\mathbf{w}_r = \mathbf{0}$

LOOP:

Choose a sub-graph $i \in [m], A_j \times B_j \in \mathbf{E}(\gamma^i)$

UPDATE:

$$\forall a \in A_j : \mathbf{u}_a = \mathbf{w}_a - \alpha_a^{i,j}\mathbf{x}_i \quad \forall b \in B_j : \mathbf{u}_b = \mathbf{w}_b + \beta_b^{i,j}\mathbf{x}_i$$

SOLVE:

$$(\alpha^{i,j}, \beta^{i,j}, \{\mathbf{w}_r\}) = \text{SOPOPO}(\{\mathbf{u}_r\}, \mathbf{x}^i, \gamma^i, A_j, B_j, C\sigma_j^i)$$

OUTPUT: The final vectors $\{\mathbf{w}_r\}_{r=1}^k$

Figure 5: The procedure for solving the preference graphs problem via soft-projections.

4.2 An Iterative Procedure

We are now ready to describe our iterative algorithm. We would like to stress again that the methodology and analysis presented here have been suggested by several authors. Our procedure is a slight generalization of row action methods (Censor and Zenios, 1997) which is often referred to as decomposition methods (see also Lin (2002); Mangasarian and Musicant (1999); Platt (1998)). The iterative procedure works in rounds and operates on the dual form of the objective function. We show though that each round can be realized as a soft-projection operation. Let λ^t denote the vector of dual variables before the tth iteration of the iterative algorithm. Initially, we set $\lambda^1 = \mathbf{0}$, which constitutes a trivial feasible solution to Eq. (30). On the tth iteration of the algorithm, we choose a single example whose index is denoted r and update its dual variables. We freeze the rest of the dual variables at their current value. We cast the tth iteration as the following constrained optimization problem,

$$\lambda^{t+1} = \operatorname*{argmax}_{\lambda \in S} D(\lambda) \quad \text{s.t. } \forall i \neq r,\ \forall j \in [k_i],\ \lambda_{i,j} = \lambda_{i,j}^t \ . \tag{31}$$

Note that λ^{t+1} is essentially the same as λ^t except for the variables corresponding to the rth example, namely, $\{\lambda_{r,j} \mid j \in [k_r]\}$. In order to explicitly write the objective function conveyed by Eq. (31) let us introduce the following notation,

$$\mathbf{u} = \sum_{i \neq r} \sum_{j=1}^{k_i} \lambda_{i,j}^t \mathbf{a}^{i,j} \ . \tag{32}$$

The vector $\mathbf{u}$ is equal to the current estimate of $\bar{\mathbf{w}}$ excluding the contribution of the rth set of dual variables. With $\mathbf{u}$ on hand, we can rewrite the objective function of Eq. (31) as follows,

$$\begin{aligned} &-\frac{1}{2}\left\|\sum_{j=1}^{k_r}\lambda_{r,j}\mathbf{a}^{r,j}\right\|^2 - \left(\sum_{j=1}^{k_r}\lambda_{r,j}\mathbf{a}^{r,j}\right)\cdot\mathbf{u} - \frac{1}{2}\|\mathbf{u}\|^2 + \sum_{j=1}^{k_r}\lambda_{r,j}b^{r,j} + \sum_{i\neq r}\sum_{j=1}^{k_i}\lambda_{i,j}^t b^{i,j} \\ &\quad = -\frac{1}{2}\left\|\sum_{j=1}^{k_r}\lambda_{r,j}\mathbf{a}^{r,j}\right\|^2 + \sum_{j=1}^{k_r}\lambda_{r,j}\left(b^{r,j} - \mathbf{u}\cdot\mathbf{a}^{r,j}\right) + \Gamma \ , \end{aligned} \tag{33}$$

where Γ is a constant that does not depend on the variables in $\{\lambda_{r,j} \mid j \in [k_r]\}$. In addition the set of variables which are not fixed must reside in S, therefore,

$$\sum_{j=1}^{k_i}\lambda_{r,j} \ \leq \ C_r \ \text{ and } \ \forall j,\ \lambda_{r,j} \geq 0 \ . \tag{34}$$

The fortunate circumstances are that the optimization problem defined by Eq. (33) subject to the constraints given in Eq. (34) can be rephrased as a soft-projection problem. Concretely, let us define the following soft-projection problem,

$$\begin{aligned} \min_{\bar{\mathbf{w}},\xi^r} \quad & \frac{1}{2}\|\bar{\mathbf{w}} - \mathbf{u}\|^2 + C_r\xi^r \\ \text{s.t.} \ \ & \forall j \in [k_r],\ \ \bar{\mathbf{w}}\cdot\mathbf{a}^{r,j} \geq b^{r,j} - \xi^r \\ & \xi^r \geq 0 \ . \end{aligned} \tag{35}$$

The value of $\lambda_{r,j}^{t+1}$ is obtained from the optimal value of the dual problem of Eq. (35) as we now show. The Lagrangian of Eq. (35) is

$$\mathcal{L} \ = \ \frac{1}{2}\|\bar{\mathbf{w}} - \mathbf{u}\|^2 + C_r\xi^r + \sum_{j=1}^{k_r}\lambda_{r,j}\left(b^{r,j} - \xi^r - \bar{\mathbf{w}}\cdot\mathbf{a}^{r,j}\right) - \zeta_r\xi^r \ .$$

Differentiating with respect to $\bar{\mathbf{w}}$ and comparing to zero give,

$$\bar{\mathbf{w}} \ = \ \mathbf{u} + \sum_{j=1}^{k_r}\lambda_{r,j}\,\mathbf{a}^{r,j} \ .$$

As in the previous derivations of the dual objective functions we also get that,

$$C_r - \zeta_r - \sum_{j=1}^{k_r}\lambda_{r,j} \ = \ 0 \ ,$$

and thus the Lagrange multipliers must satisfy,

$$\sum_{j=1}^{k_r}\lambda_{r,j} \ \leq \ C_r \ .$$

Therefore, the dual problem of Eq. (35) becomes,

$$\max_{\lambda_{r,\cdot}} \quad -\frac{1}{2}\left\|\sum_{j=1}^{k_r}\lambda_{r,j}\,\mathbf{a}^{r,j}\right\|^2 + \sum_{j=1}^{k_r}\lambda_{r,j}\,(b^{r,j}-\mathbf{u}\cdot\mathbf{a}^{r,j}) \quad \text{s.t.} \\ \sum_{j=1}^{k_r}\lambda_{r,j}\le C_r \ \text{ and } \ \forall j,\ \lambda_{r,j}\ge 0\ , \tag{36}$$

which is *identical* to the problem defined by Eq. (33) subject to the constraints given by Eq. (34).

In summary, our algorithm works by updating one set of dual variables on each round while fixing the rest of the variables to their current values. Finding the optimal value of the unrestricted variables is achieved by defining an instantaneous soft-projection problem. The instantaneous soft-projection problem is readily solved using the machinery developed in the previous section. The pseudo-code of this iterative procedure is given in Fig. 5. It is therefore left to reason about the formal properties of the iterative procedure. From the definition of the update from Eq. (31) we clearly get that on each round we are guaranteed to increase the dual objective function unless we are already at the optimum. In the next subsection we show that this iterative paradigm converges to the global optimum of the dual objective function.

To conclude this section, we would like to note that a prediction of our label-ranking function is solely based on inner products between vectors from $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$ and an instance $\mathbf{x}$. In addition, as we have shown in the previous section, the solution of each soft projection takes the form $\mathbf{w}_a = \mathbf{u}_a + \alpha_a\mathbf{x}^i$ and $\mathbf{w}_b = \mathbf{u}_b - \beta_b\mathbf{x}^i$. Since we initially set all vectors to be the zero vector, we get that at each step of the algorithm all the vectors can be expressed as linear combinations of the instances. Thus, as in the case of support vector machines for classification problems, we can replace the inner product operation with any Mercer kernel (Vapnik, 1998).

4.3 Analysis of Convergence

To analyze the convergence of the iterative procedure we need to introduce a few more definitions. We denote by D^t the value of the dual objective function *before* the tth iteration and by $\Delta_t = D^{t+1} - D^t$ the increase in the dual on the tth iteration. We also denote by $\Delta^i(\lambda)$ the potential increase we have gained had we chosen the ith example for updating λ. We assume that on each iteration of the algorithm, we choose an example, whose index is r, which attains the maximal increase in the dual, therefore $\Delta^r(\lambda) = \max_i \Delta^i(\lambda^t)$. Last, let $D^\star$ and $\lambda^\star$ denote the optimal value and argument of the dual objective function. Our algorithm maximizes the dual objective on each iteration subject to the constraint that for all $i \neq r$ and $j \in [k_i]$, the variables $\lambda_{i,j}$ are kept intact. Therefore, the sequence $D^1, D^2, \ldots$ is monotonically non-decreasing.

To prove convergence we need the following lemma which says that if the algorithm is at sub-optimal solution then it will keep increasing the dual objective on the subsequent iteration.

Lemma 5 *Let λ be a suboptimal solution, $D(\lambda) < D^\star$. Then there exists an example r for which $\Delta^r(\lambda) > 0$.*

Proof Assume by contradiction that for all i, $\Delta^i(\lambda) = 0$ and yet $D(\lambda) < D^\star$. In this case we clearly have that $\lambda \neq \lambda^\star$. Let $\mathbf{v} = \lambda^\star - \lambda$ denote the difference between the optimal solution and the current solution and denote $h(\theta) = D(\lambda + \theta\mathbf{v})$ the value of the dual obtained by moving along the direction

$\mathbf{v}$ from λ. Since $D(\lambda)$ is concave then so is h. Therefore, the line tangent to h at 0 resides above h at all points but $\theta = 0$. We thus get that, $h(0) + h'(0)\theta \geq h(\theta)$ and in particular for $\theta = 1$ we obtain,

$$h'(0) \geq h(1) - h(0) = D(\lambda^\star) - D(\lambda) > 0 \ .$$

Let ∇D denote the gradient of the dual objective at λ. Since $h'(0) = \nabla D \cdot \mathbf{v}$ we get that,

$$\nabla D \cdot \mathbf{v} \ > \ 0 \ . \tag{37}$$

We now rewrite $\mathbf{v}$ as the sum of vectors,

$$\mathbf{v} = \sum_{i=1}^{m} \mathbf{z}^i \quad \text{where} \ \ z^i_{r,j} = \begin{cases} v_{r,j} & r = i \\ 0 & r \neq i \end{cases} \ .$$

In words, we rewrite $\mathbf{v}$ as the sum of vectors each of which corresponds to the dual variables appearing in a single soft-projection problem induced by the ith example. From the definition of $\mathbf{z}^i$ together with the form of the dual constraints we get that the vector $\lambda + \mathbf{z}^i$ is also a feasible solution for the dual problem. Using the assumption that for all i, $\Delta^i(\lambda) = 0$, we get that for each $\theta \in [0,1]$, $D(\lambda) \geq D(\lambda + \theta \mathbf{z}^i)$. Analogously to h we define the scalar function $h_i(\theta) = D(\lambda + \theta \mathbf{z}^i)$. Since h_i is derived from the dual problem by constraining the dual variables to reside on the line $\lambda + \theta \mathbf{z}^i$, then as the function D, h_i is also continuously differentiable. The fact that $h_i(0) \geq h_i(\theta)$ for all $\theta \in [0,1]$ now implies that $h_i'(0) \leq 0$. Furthermore, $\nabla D \cdot \mathbf{z}^i = h_i'(0) \leq 0$ for all i which gives,

$$\nabla D \cdot \mathbf{v} \ = \ \nabla D \cdot \sum_{i=1}^{m} \mathbf{z}^i \ = \ \sum_{i=1}^{m} \nabla D \cdot \mathbf{z}^i \ \leq \ 0 \ ,$$

which contradicts Eq. (37). ■

Equipped with the above lemma we are now ready to prove that the iterative algorithm converges to an optimal solution.

Theorem 6 *Let D^t denote the value of the dual objective after the t'th iteration of the algorithm defined in Eq. (31). Denote by $D^\star$ the optimum of the problem given in Eq. (30). Then, the sequence $D^1, D^2, \ldots, D^t, \ldots$ converges to $D^\star$.*

Proof Recall that the primal problem has a trivial feasible solution which is attained by setting $\bar{\mathbf{w}} = \mathbf{0}$ and $\xi^i = \max_j b^{i,j}$. For this solution the value of the primal problem is finite. Since the value of the dual problem cannot exceed the value of the primal problem we get that $D^\star < \infty$. Therefore, the sequence of dual objective values is a monotonic, non-decreasing, and upper bounded sequence, $D^1 \leq D^2 \leq \ldots \leq D^t \leq \ldots \leq D^\star < \infty$. Thus, this sequence converges to a limit which we denote by D'. It is left to show that $D' = D^\star$. Assume by contradiction that $D^\star - D' = \varepsilon > 0$. The set of feasible dual solutions, S, is a compact set. Let $\Delta' : S \to \mathbb{R}$ be the average increase of the dual over all possible choices for an example to use for updating λ,

$$\Delta'(\lambda) = \frac{1}{m} \sum_i \Delta^i(\lambda) \ .$$

On each iteration we have by construction that $\Delta_t \geq \Delta'(\lambda^t)$. Define $A = \{\lambda : D(\lambda) > D^\star - \varepsilon/2\}$. From the concavity of D we get that the set $S \setminus A$ is a compact set. Since Δ' is a continuous function

it attains a minimum value over $S \setminus A$. Denote this minimum value by κ and let $\tilde{\lambda}$ be the point which attains this minimum. From Lemma 5 we know that $\kappa > 0$ since otherwise $D(\tilde{\lambda})$ would have equal to $D^\star$ which in turn contradicts the fact that $\tilde{\lambda} \notin A$. Since for all t we know that $D^t \leq D' = D^\star - \varepsilon$ we conclude that $\lambda^t \in S \setminus A$. This fact implies that for all t ,

$$\Delta_t \geq \Delta'(\lambda^t) \geq \Delta'(\tilde{\lambda}) = \kappa \ .$$

The above lower bound on the increase in the dual implies that the sequence $D^1, D^2, D^3, \ldots$ diverges to infinity and thus $D' = \infty$ which is in contradiction to the fact that $D' = D^\star - \varepsilon < \infty$. ■

5. Experiments

In this section we compare the SOPOPO algorithm from Fig. 3 and our iterative procedure for soft-projection onto multiple polyhedra from Fig. 5 to a commercial interior point method called LOQO (Vanderbei, 1999).

Our first set of experiments focuses on assessing the efficiency of SOPOPO for soft-projection onto a *single* polyhedron. In this set of experiments, the data was generated as follows. First, we chose the number of classes $k = |\mathcal{Y}|$ and defined E to be the set $A \times B$ with $A = [k/2]$ and $B = [k] \setminus [k/2]$. We set the value of γ_r to be one for $r \in A$ and otherwise it was set to zero. We then sampled an instance $\mathbf{x}$ and a set of vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ from a 100-dimensional Normal distribution of a zero mean and an identity matrix as a covariance matrix. After generating the instance and the targets, we presented the optimization problem of Eq. (7) to SOPOPO and to the LOQO optimization package. We repeated the above experiment for different values of k ranging from 10 through 100. For each value of k we repeated the entire experiment ten times, where in each trial we generated a new problem. We then averaged the results over the ten trials. The average CPU time consumed by the two algorithms as a function of k is depicted on the left hand side of Fig. 6. We would like to note that we have implemented SOPOPO both in Matlab and C++. We used the Matlab interface to LOQO, while LOQO itself was run in its native mode. We report results using our Matlab implementation of SOPOPO in order to eliminate possible implementation advantages. Our Matlab implementation follows the pseudo-code of Fig. 3. Nevertheless, as clearly indicated by the results, the time consumed by SOPOPO is negligible and exhibits only a very minor increase with k. In contrast, the run time of LOQO increases significantly with k. The apparent advantage of our algorithm over LOQO can be attributed to a few factors. First, LOQO is a general purpose *numerical* optimization toolkit. Its generality is clearly a two edged sword as it employs a numerical interior point method regardless of the problem on hand. Furthermore, LOQO was set to solve numerically the soft-projection problem of Eq. (7) while SOPOPO solves optimally the equivalent reduced problem of Eq. (19). To eliminate the latter mitigating factor which is in favor of SOPOPO, we repeated the same experiment as before while presenting to LOQO the reduced optimization problem rather than the original soft-projection problem. The results are depicted on the right hand side of Fig. 6. Yet again, the run time of SOPOPO is still significantly lower than LOQO for $k > 300$ and as before there is no significant increase in the run time of SOPOPO as k increases.

The second experiment compares the performance of the iterative algorithm from Fig. 5 and LOQO in the batch setting described by Eq. (6). In this experiment we generated synthetic data as follows. First, we chose the number of classes $k = |\mathcal{Y}|$ and sampled m instances from a 100-dimensional Normal distribution of a zero mean and an identity covariance matrix. We next sam-

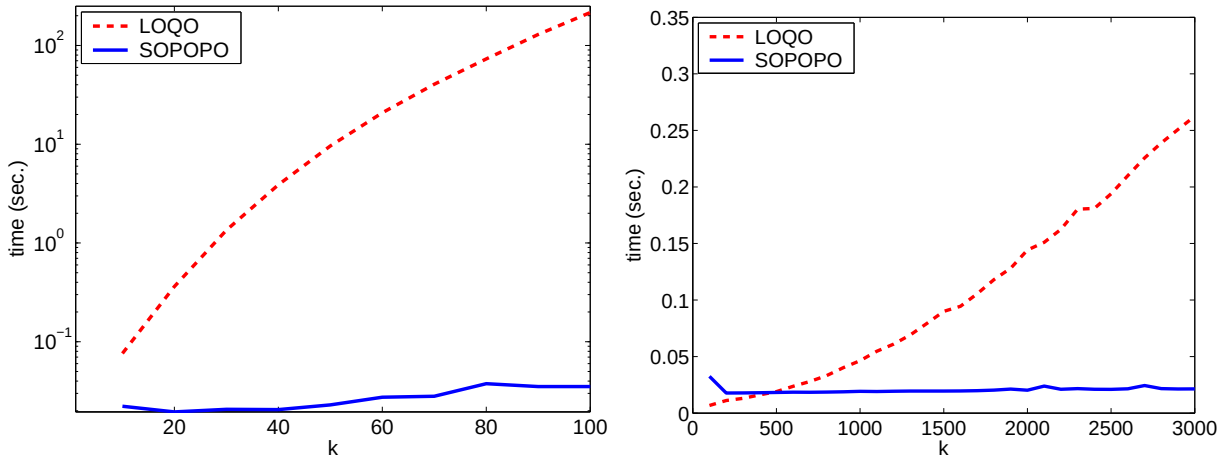


Figure 6: A comparison of the run-time of SOPOPO and LOQO on the original soft-projection problem defined in Eq. (7) (left) and on the reduced problem from Eq. (19) (right).

pled a set of vectors $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$ from the same Gaussian distribution. For each instance $\mathbf{x}^i$, we calculated the vector $\mathbf{v}^i \in \mathbb{R}^k$, whose r'th element is $\mathbf{w}_r \cdot \mathbf{x}^i$. We then set A^i to be the indices of the top $k/2$ elements of $\mathbf{v}^i$ while B^i consisted of all the rest of the elements, $[k] \setminus A^i$. For example, assume that $\mathbf{v}^i = (0.4, 4.1, 3.5, -2)$ then $A^i = \{2,3\}$ and $B^i = \{1,4\}$. As feedback we set $\gamma^i_a = 1$ for all $a \in A^i$ and for $b \in B^i$ we set $\gamma^i_b = 0$. In our running example, the resulting vector γ^i amounts to $(0,1,1,0)$. Finally, we set $\mathbf{E}(\gamma^i) = \{E^i\}$, where $E^i = A \times B$, and the value of σ was always 1. We repeated the above process for different values of k ranging from 20 through 100. The number of examples was fixed to be $10k$ and thus ranged from 200 through 1000. The value of C was set to be $1/m$. In each experiment we terminated the wrapper procedure described in Fig. 5 when the gap between the primal and dual objective functions went below 0.01. We first tried to execute LOQO with the original optimization problem described in Eq. (6). However, the resulting optimization problem was too large for LOQO to manage in a reasonable time, even for the smallest problem ($k = 20$). Our iterative algorithm solves such small problems in less than a second. To facilitate a more meaningful comparison, we used the techniques described in Sec. 3 and replaced the original optimization problem from Eq. (6) with the following reduced problem,

$$\begin{aligned} \max_{\alpha,\beta} \quad & -\frac{1}{2}\sum_{r=1}^{k}\left\|\sum_{i:r\in A^i}\alpha^i_r\mathbf{x}^i - \sum_{i:r\in B^i}\beta^i_r\mathbf{x}^i\right\|^2 + \sum_{r=1}^{k}\left(\sum_{i:r\in A^i}\alpha^i_r\gamma^i_r - \sum_{i:r\in B^i}\beta^i_r\gamma^i_r\right) \\ \text{s.t.} \quad & \forall i\in[m] : \forall a\in A^i,\ \alpha^i_a \geq 0 \ \text{ and } \ \forall b\in B^i,\ \beta^i_b \geq 0 \\ & \forall i\in[m] : \sum_{a\in A^i}\alpha^i_a \ = \ \sum_{b\in B^i}\beta^i_b \ \leq \ C \ . \end{aligned} \tag{38}$$

By presenting the reduced problem given in Eq. (38) to LOQO, we injected quite a bit of prior knowledge that made the task manageable for LOQO. The derivation of the above reduced problem is given in appendix C. The results are summarized in Fig. 7. As clearly can be seen from the graph, our iterative algorithm outperforms LOQO, in particular as the size of the problem increases. Due to the nature of the decomposition procedure, our running time is no longer independent of the

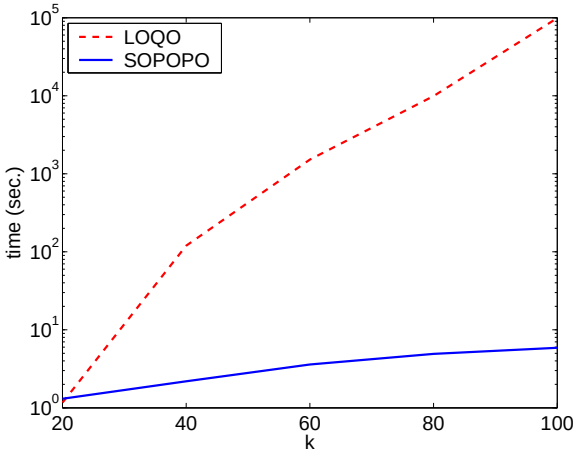


Figure 7: A comparison of the run-time in batch settings of SOPOPO and LOQO (using the reduced problem in Eq. (38)). The number of examples was set to be 10 times the number of labels (denoted k) in each problem.

value of k as the number of graphs grows with k. Nonetheless, even for $k = 100$ the run time of SOPOPO's wrapper does not exceed 4 seconds. These promising results emphasize the viability of our approach for large scale optimization problems.

The last experiment underscores an interesting property of our iterative algorithm. In this experiment we have used the same data as in the previous experiment with $k = 100$ and $m = 1000$. After each iteration of the algorithm, we examined both the increase in the dual objective after the update and the difference between the primal and dual values. The results are shown in Fig. 8. The graphs exhibit a phenomena reminiscent of a phase transition. After about 1000 iterations, which is also the number of examples, the increase in the dual objective becomes miniscule. This phase transition is also exhibited for other choices of m, k and C. Note in addition that as the number of epochs increases, the increase of the dual objective becomes very small relatively to the duality gap. It is common to use the increase of the dual objective as a stopping criterion and the last experiment indicates that this criterion does not necessarily imply convergence. We leave further investigation of these phenomena to future research.

We would like to conclude this section with a short discussion which contrasts our approach with previous algorithms. Previous large margin approaches for label ranking associate a unique slack variable with each constraint which is induced by a pair of labels. See for example (Elisseeff and Weston, 2001) and the SVM-light implementation of label ranking (Joachims, 2002). Thus, using the terminology of this paper, these methods employ the overly simple all-pair decomposition (see Fig. 2). Using the all-pair decomposition, the label ranking problem is reduced to a binary classification problem. Indeed, the soft projection problem can be solved analytically and our wrapper algorithm from Fig. 5 is equivalent to the SOR algorithm for binary classification described in (Mangasarian and Musicant, 1999). The practical performance of the SOR algorithm for binary classification was extensively studied by Mangasarian and others. One of the main contributions of this paper is a general and flexible algorithmic framework for label ranking which can be carried with more complex decompositions. Moreover, trying to import one of the previously studied

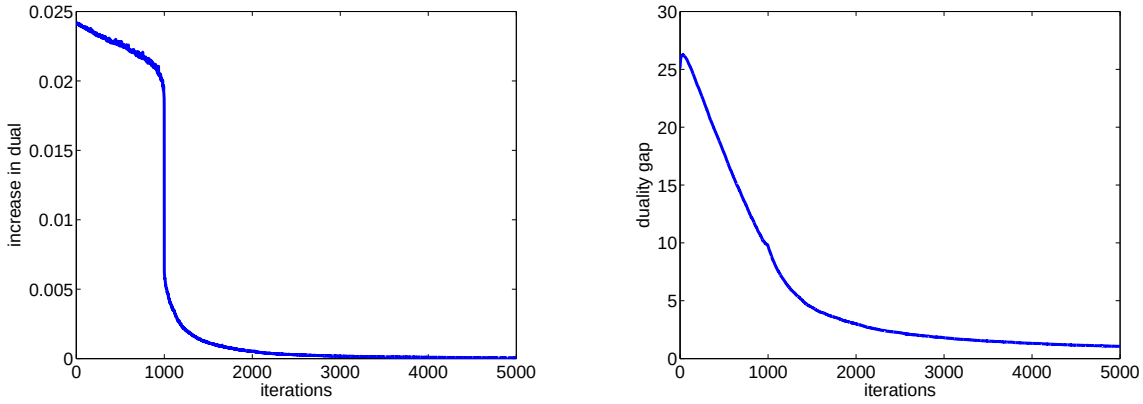


Figure 8: The increase in the dual objective (left) and the primal-dual gap (right) as a function of the number of iterations of the iterative algorithm in Fig. 5.

approach to our setting is difficult. A main obstacle is attributed to the fact that the set of feasible solutions for the dual problem must satisfy the constraint $\sum_a \alpha_a = \sum_b \beta_b \leq C$. Thus, a sequential minimization algorithm must update at least 4 dual variables on each iteration in order to preserve the feasibility of the dual solution. Therefore the SMO algorithm of Platt (1998) is not easily applicable to our setting. The SOPOPO algorithm suggests an efficient alternative by updating atomically all the dual variables of each sub-graph.

6. Discussion

We described an algorithmic framework for label ranking. Each iteration of our algorithm is based on SOPOPO, a fast procedure for soft projection onto a single polyhedron. There are several possible extensions of the work presented in this paper. One of them is further generalization of SOPOPO to more complex polyhedral constraints. Recall that SOPOPO is designed for projecting onto a polyhedron which is defined according to a complete bipartite graph. The generalization of SOPOPO to decompositions consisting of k-partite graphs is one particular interesting task. Another type of polyhedra that naturally emerges is regression problems with multiple outputs. In this setting, we would like the predicted differences $f_r(\mathbf{x}) - f_s(x)$ to be as close as possible to the target differences $\gamma_r - \gamma_s$, possibly up to an insensitivity term ε. This problem can be formalized by replacing the constraint $f_r(\mathbf{x}) - f_s(\mathbf{x}) \geq \gamma_r - \gamma_s - \xi$ with the constraint $|(f_r(\mathbf{x}) - f_s(\mathbf{x})) - (\gamma_r - \gamma_s)| \leq \varepsilon + \xi$. Yet another interesting direction is the applicability of SOPOPO to online learning ranking (Crammer and Singer, 2005) where each online update is performed efficiently using SOPOPO. The phase transition phenomenon underscored in our experiments surfaces the important issue of generalization properties of our algorithm. In particular, the fact that increases in the value of dual become miniscule suggests the usage of early stopping so long as the prediction accuracy does not degrade. Finally, we plan to work on real world applications of SOPOPO to tasks such as category ranking for text documents.

Acknowledgments

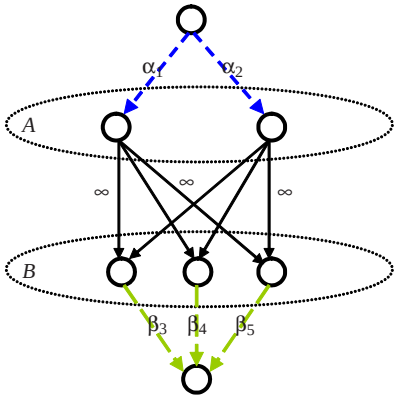


Figure 9: An illustration of the construction of a flow graph for $A = \{1,2\}$ and $B = \{3,4,5\}$.

We would like to thank Koby Crammer for numerous fruitful discussions. Aaron D'Souza and Vineet Gupta have played a key role in motivating this line of research. Thanks also to Yair Censor, Mayur Datar, Ofer Dekel, Phil Long, Michael fink, and Peter Norvig for their comments and feedback. The work of Shai Shalev-Shwartz was supported by the Israeli Science Foundation under grant no. 522/04 and by NSF ITR award 0205594.

Appendix A. The Equivalence Between the Dual Problems in Eq. (18) and Eq. (13)

In this appendix we prove that the solutions of the problem in Eq. (18) and our original dual problem from Eq. (13) are equivalent. (For an alternative derivation see also (Fung et al., 2006)). To do so, it suffices to show that for each feasible solution of the reduced problem there exists an equivalent feasible solution of the original problem and vice versa. Clearly, given τ which satisfies the constraints imposed by Eq. (13), defining α and β as given by Eq. (14) would satisfy the constraints of Eq. (18) and furthermore $D(\alpha,\beta) = D(\tau)$. Denoting the optimal solution of Eq. (13) by $\tau^\star$ and that of Eq. (18) by $(\alpha^\star,\beta^\star)$, we immediately get that $D(\alpha^\star,\beta^\star) \geq D(\tau^\star)$. We are thus left to show that for each feasible solution α,β there exists a feasible solution τ such that $D(\tau) = \mathcal{D}(\alpha,\beta)$. This reverse mapping is non-trivial and there does not exist a closed form description of the mapping from α,β to τ. The existence of such a mapping is provided in Lemma 7 below which uses the duality of max-flow and min-cut. Lemma 7 immediately implies that $D(\tau^\star) \geq D(\alpha^\star,\beta^\star)$. In summary, we have shown that both $D(\tau^\star) \leq D(\alpha^\star,\beta^\star)$ and $D(\tau^\star) \geq D(\alpha^\star,\beta^\star)$ holds and therefore $D(\tau^\star) = D(\alpha^\star,\beta^\star)$.

Lemma 7 *Let (α,β) be a feasible solution of the reduced problem given in Eq. (18). Then, there exists a feasible solution τ of the original problem (Eq. (13)) such that $D(\tau) = D(\alpha,\beta)$.*

Proof The proof is based on the duality of max-flow and min-cut (see for example Cormen et al. (1990)). Given a feasible solution (α,β) defined over the sets A and B we construct a directed graph (V',E'). The set of nodes of the graph consists of the original nodes defined by the sets A and B and two additional nodes s which serves as a source and t which is a sink, $V' = A \cup B \cup \{s,t\}$. In addition to the original edges of the bipartite graph supported by A and B we add edges from s to all the nodes in A and from all the nodes in B to t and thus $E' = (A \times B) \cup (\{s\} \times A) \cup (B \times \{t\})$.

Each edge $e \in E'$ is associated with a capacity value $c(e)$. For each $e \in A \times B$ we define $c(e) = \infty$. For each edge of the form (s,a) where $a \in A$ we define $c(e) = \alpha_a$ and analogously for (b,t) where $b \in B$ we set $c(e) = \beta_b$. An illustration of the construction is given in Fig. 9 where $A = \{1,2\}$ and $B = \{3,4,5\}$. We are now going to define a flow problem for (V', E'). We show in the sequel that *maximal* flow in the graph above defines a feasible solution for the original optimization problem. Furthermore, by using the max-flow min-cut duality, we also show that the value attained by the induced solution coincides with the value of the reduced optimization problem for (α, β).

A flow for the graph above is an assignment of non-negative values to edges, $\mathcal{F} : E' \rightarrow \mathbb{R}_+$, which satisfies

$$\begin{array}{ll} (i) & \forall (r,v) \in E',\ \mathcal{F}\left((r,v)\right) \leq \mathrm{c}(r,v) \\ (ii) & \forall v \in V',\ \displaystyle\sum_{r:(r,v)\in E'} \mathcal{F}\left((r,v)\right) = \sum_{r:(v,r)\in E'} \mathcal{F}\left((v,r)\right)\ . \end{array} \tag{39}$$

The value of a flow function is defined as the total flow outgoing the source,

$$\mathrm{val}(\mathcal{F}) \ = \ \sum_{r:(\mathrm{s},r)\in E'} \mathcal{F}\left((\mathrm{s},r)\right)\ .$$

Let $\mathcal{F}^\star$ denote the flow attaining the maximal value among all possible flows, that is $\mathrm{val}(\mathcal{F}^\star) \geq \mathrm{val}(\mathcal{F})$. We next prove that $\mathrm{val}(\mathcal{F}^\star) = \sum_{a\in A} \alpha_a$. To do so we use the max-flow min-cut duality theorem. This theorem states that the value of the maximal flow equals the value of the minimal cut of a graph. Formally, a cut of the graph is a subset $S \subset V'$ such that $\mathrm{s} \in S$ and $\mathrm{t} \notin S$. The value of a cut is defined as the total *capacity* of edges outgoing from S to $V' \setminus S$,

$$\mathrm{val}(S) \ = \ \sum_{(r,v)\in S\times(V'\setminus S)\cap E'} \mathrm{c}(r,v)\ .$$

A cut is said to be minimal if its value does not exceed the value of any other cut of the graph. The value of the cut $S = \{\mathrm{s}\}$ is equal to $\sum_{y\in A} \alpha_y$. We now show that S is a minimal cut. We note in passing that while there might exist other cuts attaining the minimum value, for our purpose it suffices to show that $S = \{s\}$ is a minimal cut. Let S' be a cut different from S. Clearly, if $\mathrm{val}(S') = \infty$ then S' cannot be minimal. We thus can safely assume that $\mathrm{val}(S') < \infty$. If there exists a node $a \in A \cap S'$ then all the nodes in B must also reside in S'. Otherwise, there exists an edge (a,b) of an infinite capacity which crosses the cut and $\mathrm{val}(S') = \infty > \mathrm{val}(S)$. Since t cannot be in S' we get that for each $b \in B$, the edge (b,t) crosses the cut and therefore the value of the cut is at least $\sum_{b\in B} \beta_b = \sum_{a\in A} \alpha_a$. If on the other hand $A \cap S' = \emptyset$ then all the edges from s to the nodes in A cross the cut. Therefore, $\mathrm{val}(S)$ is again at least $\sum_{y\in A} \alpha_y$. We have thus shown that $S = \{\mathrm{s}\}$ is a minimal cut of the flow graph.

From the duality theorem of max-flow and min-cut we get that there exists a minimal flow $\mathcal{F}^\star$ such that $\mathrm{val}(\mathcal{F}^\star) = \sum_{a\in A} \alpha_a$. Since each outgoing edge from s hits a different node in A, we must have that $\mathcal{F}^\star(\,(\mathrm{s},a)\,) = \alpha_a$ in order to reach the optimal flow value. Similarly, for each $b \in B$ we get that $\mathcal{F}^\star(\,(b,\mathrm{t})\,) = \beta_b$. We now set $\tau_{a,b} = \mathcal{F}^\star(\,(a,b)\,)$ for each $(a,b) \in A \times B$. Since a proper flow associates a non-negative value with each edge we obtain that $\tau_{a,b} \geq 0$. From the conservation of flow we get that,

$$\alpha_a = \mathcal{F}^\star(\,(\mathrm{s},a)\,) = \sum_{b\in B} \mathcal{F}^\star(\,(a,b)\,) = \sum_{b\in B} \tau_{a,b}\ ,$$

and

$$\beta_b = \mathcal{F}^\star(\,(b,\mathrm{t})\,) = \sum_{a\in A} \mathcal{F}^\star(\,(a,b)\,) = \sum_{a\in A} \tau_{a,b}\ .$$

Thus, this construction of τ from the optimal flow satisfies the equalities given in Eq. (14). By construction, each node $a \in A$ has one incoming edge (s,a) and outgoing edges to all nodes in B. Thus, the flow conservation requirement of Eq. (39) again implies that

$$C \geq \sum_{a \in A} \mathcal{F}^{\star}((s,a)) = \sum_{a \in A, b \in B} \mathcal{F}^{\star}((a,b)) = \sum_{a \in A, b \in B} \tau_{a,b} \ .$$

Therefore, τ adheres with the constraints of Eq. (13). In summary, we have constructed a feasible solution for the original constrained optimization problem which is consistent with the definitions of α and β. Therefore, $D(\tau) = D(\alpha, \beta)$ as required. ■

Appendix B. Technical Proofs

Proof of Lemma 3

Throughout the proof we assume that the elements of the vector μ are sorted in a non-ascending order, namely, $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_p$. Recall that the definition of $\rho(z,\mu)$ is,

$$\rho(z,\mu) \ = \ \max \left\{ j \in [p] \ : \ \mu_j - \frac{1}{j} \left(\sum_{r=1}^{j} \mu_r - z \right) > 0 \right\} \ .$$

For brevity, we refer to $\rho(z,\mu)$ simply as ρ. Denote by $\alpha^{\star}$ the optimal solution of the constrained optimization problem of Eq. (20) and let

$$\rho^{\star} = \max\{ j \ : \ \alpha^{\star}_j > 0 \} \ .$$

From Eq. (23) we know that $\alpha^{\star}_r = \mu_r - \theta^{\star} > 0$ for $r \leq \rho^{\star}$ where

$$\theta^{\star} = \frac{1}{\rho^{\star}} \left(\sum_{j=1}^{\rho^{\star}} \mu_j - z \right) \ ,$$

and therefore $\rho \geq \rho^{\star}$. We thus need to prove that $\rho = \rho^{\star}$. Assume by contradiction that $\rho > \rho^{\star}$. Let us denote by α the vector induced by the choice of ρ, that is, $\alpha_r = 0$ for $r > \rho$ and $\alpha_r = \mu_r - \theta$ for $r \leq \rho$, where,

$$\theta \ = \ \frac{1}{\rho} \left(\sum_{j=1}^{\rho} \mu_j - z \right) \ .$$

From the definition of ρ, we must have that $\alpha_\rho = \mu_\rho - \theta > 0$. Therefore, since the elements of μ are sorted in a non-ascending order, we get that $\alpha_r = \mu_r - \theta > 0$ for all $r \leq \rho$. In addition, the choice of θ implies that $\|\alpha\|_1 = z$. We thus get that α is a feasible solution as it satisfies the constraints of Eq. (20). Examining the objective function attained at α we get that,

$$\begin{aligned}
\|\alpha - \mu\|^2 &= \sum_{r=1}^{\rho^{\star}} \theta^2 + \sum_{r=\rho^{\star}+1}^{\rho} \theta^2 + \sum_{r=\rho+1}^{p} \mu_r^2 \\
&< \sum_{r=1}^{\rho^{\star}} \theta^2 + \sum_{r=\rho^{\star}+1}^{\rho} \mu_r^2 + \sum_{r=\rho+1}^{p} \mu_r^2 \\
&= \sum_{r=1}^{\rho^{\star}} \theta^2 + \sum_{r=\rho^{\star}+1}^{p} \mu_r^2 \ ,
\end{aligned}$$

where to derive the inequality above we used the fact that $\mu_r - \theta > 0$ for all $r \leq \rho$. We now need to analyze two cases depending on whether $\theta^\star$ is greater than θ or not. If $\theta^\star \geq \theta$ than we can further bound $\|\alpha - \mu\|^2$ from above as follows,

$$\|\alpha-\mu\|^2 \;<\; \sum_{r=1}^{\rho^\star}\theta^2 + \sum_{r=\rho^\star+1}^{p}\mu_r^2 \;\leq\; \sum_{r=1}^{\rho^\star}(\theta^\star)^2 + \sum_{r=\rho^\star+1}^{p}\mu_r^2 \;=\; \|\alpha^\star-\mu\|^2 \;\;,$$

which contradicts the optimality of $\alpha^\star$. We are thus left to show that the case $\theta > \theta^\star$ also leads to a contradiction. We do so by constructing a vector $\tilde{\alpha}$ from $\alpha^\star$. We show that this vector satisfies the constraints of Eq. (20) hence it is a feasible solution. Finally, we show that the objective function attained by $\tilde{\alpha}$ is strictly smaller than that of $\alpha^\star$. We define the vector $\tilde{\alpha} \in \mathbb{R}^k$ as follows,

$$\tilde{\alpha}_r = \begin{cases} \alpha^\star_{\rho^\star} - \varepsilon & r = \rho^\star \\ \varepsilon & r = \rho^\star + 1 \\ \alpha^\star_r & \text{otherwise} \end{cases} \;\;,$$

where $\varepsilon = \frac{1}{2}(\mu_{\rho^\star+1} - \theta^\star)$. Since we assume that $\theta > \theta^\star$ and $\rho > \rho^\star$ we know that $\alpha_{\rho^\star+1} = \mu_{\rho^\star+1} - \theta > 0$ which implies that

$$\tilde{\alpha}_{\rho^\star+1} = \frac{1}{2}(\mu_{\rho^\star+1} - \theta^\star) > \frac{1}{2}(\mu_{\rho^\star+1} - \theta) = \frac{1}{2}\alpha_{\rho^\star+1} > 0 \;\;.$$

Furthermore, we also get that,

$$\tilde{\alpha}_{\rho^\star} \;=\; \mu_{\rho^\star} - \frac{1}{2}\mu_{\rho^\star+1} - \frac{1}{2}\theta^\star \;>\; \frac{1}{2}\left(\mu_{\rho^\star+1} - \theta\right) \;=\; \frac{1}{2}\alpha_{\rho^\star+1} \;>\; 0 \;\;.$$

In addition, by construction we get that the rest of components of $\tilde{\alpha}$ are also non-negative. Our construction also preserves the norm, that is $\|\tilde{\alpha}\|_1 = \|\alpha^\star\|_1 = z$. Thus, the vector $\tilde{\alpha}$ is also a feasible solution for the set of constraints defined by Eq. (20). Alas, examining the difference in the objective functions attained by $\tilde{\alpha}$ and $\alpha^\star$ we get,

$$\begin{aligned} \|\alpha^\star-\mu\|^2 - \|\tilde{\alpha}-\mu\|^2 &= (\theta^\star)^2 + \mu^2_{\rho^\star+1} - \left((\theta^\star+\varepsilon)^2 + \left(\mu_{\rho^\star+1} - \varepsilon\right)^2\right) \\ &= 2\varepsilon\underbrace{(\mu_{\rho^\star+1} - \theta^\star)}_{=2\varepsilon} - 2\varepsilon^2 \;=\; 2\varepsilon^2 \;>\; 0 \;\;. \end{aligned}$$

We thus obtained the long desired contradiction which concludes the proof. ■

Proof of Thm. 4

Plugging the value of the optimal solution α from Eq. (23) into the objective $\|\alpha - \mu\|^2$ and using Lemma 3 give that,

$$g(z;\mu) \;=\; \frac{1}{\rho(z;\mu)}\left(\sum_{r=1}^{\rho(z;\mu)}\mu_r - z\right)^2 + \sum_{r=\rho(z;\mu)+1}\mu_r^2 \;\;,$$

where, to remind the reader, the number of strictly positive α's is,

$$\rho(z;\mu) = \max\left\{\rho : \mu_\rho - \frac{1}{\rho}\left(\sum_{r=1}^{\rho}\mu_r - z\right) \geq 0\right\} \;\;.$$

Throughout the proof μ is fixed and known. We therefore abuse our notation and use the shorthand $\rho(z)$ for $\rho(z;\mu)$. Recall that μ is given in a non-ascending order, $\mu_{i+1} \leq \mu_i$ for $i \in [p-1]$. Therefore, we get that

$$\begin{aligned} z_{i+1} &= \sum_{r=1}^{i+1} \mu_r - (i+1)\mu_{i+1} = \sum_{r=1}^{i} \mu_r + \mu_{i+1} - \mu_{i+1} - i\,\mu_{i+1} \\ &= \sum_{r=1}^{i} \mu_r - i\,\mu_{i+1} \geq \sum_{r=1}^{i} \mu_r - i\,\mu_i = z_i \ . \end{aligned}$$

Thus, the sequence $z_1, z_2, \ldots, z_p$ is monotonically non-decreasing and the intervals $[z_i, z_{i+1})$ are well defined. The definition of $\rho(z)$ implies that for all $z \in [z_i, z_{i+1})$ we have $\rho(z) = \rho(z_i) = i$. Hence, the value of $g(z;\mu)$ for each $z \in [z_i, z_{i+1})$ is,

$$g(z;\mu) = \frac{1}{i}\left(\sum_{r=1}^{i} \mu_r - z\right)^2 + \sum_{r=i+1}^{p} \mu_r^2 \ .$$

We have thus established the fact that $g(z;\mu)$ is a quadratic function in each interval (z_i, z_{i+1}) and in particular it is continuous in each such sub-interval. To show that g is continuous in $[0, C]$ we need to examine all of its knots z_i. Computing the left limit and the right limit of g at each knot we get that,

$$\begin{aligned} \lim_{z \downarrow z_i} g(z;\mu) &= \lim_{z \downarrow z_i} \frac{1}{i}\left(\sum_{r=1}^{i} \mu_r - z\right)^2 + \sum_{r=i+1}^{p} \mu_r^2 \\ &= \frac{1}{i}\left(\sum_{r=1}^{i} \mu_r - \sum_{r=1}^{i} \mu_r + i\mu_i\right)^2 + \sum_{r=i+1}^{p} \mu_r^2 \\ &= i\mu_i^2 + \sum_{r=i+1}^{p} \mu_r^2 \ , \end{aligned}$$

and

$$\begin{aligned} \lim_{z \uparrow z_i} g(z;\mu) &= \lim_{z \uparrow z_i} \frac{1}{i-1}\left(\sum_{r=1}^{i-1} \mu_r - z\right)^2 + \sum_{r=i}^{p} \mu_r^2 \\ &= \frac{1}{i-1}\left(\sum_{r=1}^{i-1} \mu_r - \sum_{r=1}^{i} \mu_r + i\mu_i\right)^2 + \sum_{r=i}^{p} \mu_r^2 \\ &= (i-1)\mu_i^2 + \sum_{r=i}^{p} \mu_r^2 = i\mu_i^2 + \sum_{r=i+1}^{p} \mu_r^2 \ . \end{aligned}$$

Therefore, $\lim_{z \downarrow z_i} g(z;\mu) = \lim_{z \uparrow z_i} g(z;\mu)$ and g is indeed continuous. The continuity of the derivative of g is shown by using the same technique of examining the right and left limits at each knot z_i for the function,

$$g'(z;\mu) = \frac{2}{i}\left(z - \sum_{r=1}^{i} \mu_r\right) \ .$$

Finally, we use the fact that a continuously differentiable function is convex iff its derivative is monotonically non-decreasing. Since g is quadratic in each segment $[z_i, z_{i+1}]$, g' is indeed monotonically non-decreasing in each segment. Furthermore, from the continuity of g' we get that g' is monotonically non-decreasing on the entire interval $[0,C]$. Thus, g is convex on $[0,C]$. ■

Appendix C. Derivation of Eq. (38)

In this section we derive conversion of the optimization problem from Eq. (6) to its reduced form given in Eq. (38). In Sec. 4 (Eq. (30)) we derived the dual of Eq. (6). Assuming that for each example, $\mathbf{E}(\gamma^i) = \{A^i \times B^i\}$, and using the definitions of $\mathbf{a}^{i,j}$, $b^{i,j}$, and $\bar{\mathbf{w}}$ from Sec. 4, we can rewrite the dual of Eq. (6) as

$$\begin{aligned} \max_{\tau} \quad & -\frac{1}{2}\sum_{r=1}^{k}\|\mathbf{w}_r\|^2 + \sum_{i=1}^{m}\sum_{a\in A^i}\sum_{b\in B^i}\lambda_{a,b}^i(\gamma_a^i - \gamma_b^i) \\ \text{s.t.} \quad & \forall i \in [m] : \forall (a,b) \in A^i \times B^i,\ \ \lambda_{a,b}^i \geq 0 \\ & \forall i \in [m] : \sum_{(a,b)\in A^i\times B^i}\lambda_{a,b}^i \leq C \quad , \end{aligned} \tag{40}$$

where

$$\mathbf{w}_r \ = \ \sum_{i:r\in A^i}\sum_{b\in B^i}\lambda_{r,b}^i\mathbf{x}^i - \sum_{i:r\in B^i}\sum_{a\in A^i}\lambda_{a,r}^i\mathbf{x}^i \ . \tag{41}$$

For each $a \in A^i$ define,

$$\alpha_a^i \ = \ \sum_{b\in B^i}\lambda_{a,b}^i \ , \tag{42}$$

and similarly, for each $b \in B^i$ define,

$$\beta_b^i \ = \ \sum_{a\in A^i}\lambda_{a,b}^i \ . \tag{43}$$

Using these definitions, we can rewrite Eq. (41) as,

$$\mathbf{w}_r \ = \ \sum_{i:r\in A^i}\alpha_r^i\mathbf{x}^i - \sum_{i:r\in B^i}\beta_r^i\mathbf{x}^i \ .$$

Therefore, the dual objective can be rewritten as,

$$D \ = \ -\frac{1}{2}\sum_{r=1}^{k}\left\|\sum_{i:r\in A^i}\alpha_r^i\mathbf{x}^i - \sum_{i:r\in B^i}\beta_r^i\mathbf{x}^i\right\|^2 + \sum_{r=1}^{k}\left(\sum_{i:r\in A^i}\alpha_r^i\gamma_r^i - \sum_{i:r\in B^i}\beta_r^i\gamma_r^i\right) \ .$$

As in Sec. 3, we need to enforce the additional constraints on α and β,

$$\begin{aligned} & \forall i \in [m] : \forall a \in A^i,\ \ \alpha_a^i \geq 0 \ \text{ and } \ \forall b \in B^i,\ \ \beta_b^i \geq 0 \\ & \forall i \in [m] : \sum_{a\in A^i}\alpha_a^i \ = \ \sum_{b\in B^i}\beta_b^i \ \leq \ C \ . \end{aligned}$$

Combining the dual definition with the above constraints gives the reduced problem from Eq. (38).

References

S. Agarwal and P. Niyogi. Stability and generalization of bipartite ranking algorithms. In *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory*, pages 32–47, 2005.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, New York, NY, USA, 1997.

S. Clemenon, G. Lugosi, and N. Vayatis. Ranking and scoring using empirical risk minimization. In *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory*, 2005.

W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

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

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Jornal of Machine Learning Research*, 2:265–292, 2001.

K. Crammer and Y. Singer. A new family of online algorithms for category ranking. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

K. Crammer and Y. Singer. Online ranking by projecting. *Neural Computation*, 17(1), 2005.

O. Dekel, C. Manning, and Y. Singer. Log-linear models for label ranking. In *Advances in Neural Information Processing Systems 16*, 2003.

A. Elisseeff and J. Weston. A kernel method for multi-labeled classification. In *Advances in Neural Information Processing Systems 14*, 2001.

G. Fung, R. Rosales, and B. Krishnapuram. Learning rankings via convex hull separation. In *Advances in Neural Information Processing Systems 18*, 2006.

R. Herbrich, T. Graepel, and K. Obermayer. Large marging rank boundaries for ordinal regression. In A. Smola, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2000.

C. Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85, 1957. Erratum, ibidem, p.361.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

C.-J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 13(5):1045–1052, Sept. 2002.

O. Mangasarian and D. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10, 1999.

J. C. Platt. Fast training of Support Vector Machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

C. Rudin, C. Cortes, M. Mohri, and R.E. Schapire. Margin-based ranking and boosting meet in the middle. In *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory*, pages 63–78, 2005.

R.J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Mthods and Software*, 12:451–484, 1999.

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

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.

Journal of Machine Learning Research 7 (2006) 1601–1626 Submitted 10/05; Published 7/06

Kernel-Based Learning of Hierarchical Multilabel Classification Models*

Juho Rousu JUHO.ROUSU@CS.HELSINKI.FI
Department of Computer Science
PO Box 68
FI-00014 University of Helsinki, Finland

Craig Saunders CJS@ECS.SOTON.AC.UK
Sandor Szedmak SS03V@ECS.SOTON.AC.UK
John Shawe-Taylor JST@ECS.SOTON.AC.UK
Electronics and Computer Science
University of Southampton
SO17 1BJ, United Kingdom

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

We present a kernel-based algorithm for hierarchical text classification where the documents are allowed to belong to more than one category at a time. The classification model is a variant of the Maximum Margin Markov Network framework, where the classification hierarchy is represented as a Markov tree equipped with an exponential family defined on the edges. We present an efficient optimization algorithm based on incremental conditional gradient ascent in single-example subspaces spanned by the marginal dual variables. The optimization is facilitated with a dynamic programming based algorithm that computes best update directions in the feasible set.

Experiments show that the algorithm can feasibly optimize training sets of thousands of examples and classification hierarchies consisting of hundreds of nodes. Training of the full hierarchical model is as efficient as training independent SVM-light classifiers for each node. The algorithm's predictive accuracy was found to be competitive with other recently introduced hierarchical multi-category or multilabel classification learning algorithms.

Keywords: kernel methods, hierarchical classification, text categorization, convex optimization, structured outputs

1. Introduction

In many application fields, taxonomies and hierarchies are natural ways to organize and classify objects, hence they are widely used for tasks such as text classification. In contrast, machine learning research has largely been focused on flat target prediction, where the output is a single binary or multivalued scalar variable. Naively encoding a large hierarchy either into a series of binary problems or a single multiclass problem with many possible class values suffers from the fact that dependencies between the classes cannot be represented well. For example, if a news article belongs to category MUSIC, it is very likely that the article belongs to category ENTERTAINMENT. The failure to represent these relationships leads to a steep decline of the predictive accuracy in the

*. A preliminary version of this paper appeared in Proceedings of 19th ICML, Bonn, Germany, 2005.

number of possible categories. In recent years, methods that utilize the hierarchy in learning the classification have been proposed by several authors (Koller and Sahami, 1997; McCallum et al., 1998; Dumais and Chen, 2000). Very recently, new hierarchical classification approaches utilizing kernel methods have been introduced (Hofmann et al., 2003; Cai and Hofmann, 2004; Dekel et al., 2004). The main idea behind these methods is to map the documents (or document–labeling pairs) into a potentially high-dimensional feature space where linear maximum margin separation of the documents becomes possible.

Most of the above mentioned methods assume that the object to be classified belongs to exactly one (leaf) node in the hierarchy. In this paper we consider the more general case where a single object can be classified into several categories in the hierarchy, to be specific, the multilabel is a *union of partial paths* in the hierarchy. For example, a news article about David and Victoria Beckham could belong to partial paths SPORT, FOOTBALL and ENTERTAINMENT, MUSIC but might not belong to any leaf categories such as CHAMPIONS LEAGUE. The problem of multiple partial paths was also considered in Cesa-Bianchi et al. (2004).

Recently Taskar et al. (2003) introduced a maximum margin technique which optimized an SVM-style objective function over structured outputs. This technique used a marginalization trick to obtain a polynomial sized quadratic program using marginal dual variables. This was an improvement over the exponentially-sized problem resulting from the dualization of the primal margin maximization problem, which only can be approximated with polynomial number of support vectors using a working set method (Altun et al., 2003; Tsochantaridis et al., 2004).

Even using marginal variables, however, the problem becomes infeasible for even medium sized data sets. Therefore, efficient optimization algorithms are needed. In this paper we present an algorithm for working with the marginal variables that is in the spirit of Taskar et al. (2003), however a reformulation of the objective allows a conditional-gradient method to be used which gains efficiency and also enables us to work with a richer class of loss functions.

The structure of this article is the following. In Section 2 we present the classification framework, review loss functions and derive a quadratic optimization problem for finding the maximum margin model parameters. In Section 3 we present an efficient learning algorithm relying a decomposition of the problem into single training example subproblems and conducting iterative conditional gradient ascent in marginal dual variable subspaces corresponding to single training examples. A dynamic programming algorithm is presented that used to efficiently find the best update directions. Extensions and variants are briefly discussed in Section 4. We compare the new algorithm in Section 5 to flat and hierarchical SVM learning approaches and the hierarchical regularized least squares algorithm recently proposed by Cesa-Bianchi et al. (2004). We conclude the article with discussion in Section 6.

2. Maximum Margin Hierarchical Multilabel Classification

We consider data from a domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is a set and $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$ is a Cartesian product of the sets $\mathcal{Y}_j = \{+1,-1\}, j = 1,\ldots,k$. A vector $\mathbf{y} = (y_1,\ldots,y_k) \in \mathcal{Y}$ is called the *multilabel* and the components y_j are called the *microlabels*.

We assume that a training set $\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$ has been given, consisting of training examples $(\mathbf{x}_i,\mathbf{y}_i)$ of a training pattern $\mathbf{x}_i$ and multilabel $\mathbf{y}_i$. A pair $(\mathbf{x}_i,\mathbf{y})$ where $\mathbf{x}_i$ is a training pattern and $\mathbf{y} \in \mathcal{Y}$ is arbitrary, is called a *pseudo-example*, to denote the fact that the pair may or may not be generated by the distribution generating the training examples.

A HUMAN NECESSITIES
  A 01 AGRICULTURE; FORESTRY; ANIMAL HUSBANDRY; ...
    A 01 B SOIL WORKING IN AGRICULTURE OR FORESTRY
      A 01 B 1/02 Spades; Shovels
      A 01 B 9/00 Ploughs with rotary driven tools
D TEXTILES; PAPER
  D 21 PAPER-MAKING; PRODUCTION OF CELLULOSE
    D 21 F PAPER-MAKING MACHINES
      D 21 F 1/00 Wet end of machines for making continuous webs of paper

E.C.1 Oxidoreductases
  E.C.1.1. Acting on the CH-OH group of donors.
    E.C.1.1.1 With NAD(+) or NADP(+) as acceptor.
      E.C.1.1.1.1 Alcohol dehydrogenase.
E.C.6 Ligases
  E.C.6.1 Forming carbon-oxygen bonds.
    E.C.6.1.1 Ligases forming aminoacyl-tRNA and related compounds.
      E.C.6.1.1.1 Tyrosine–tRNA ligase.

Figure 1: Examples of classification hierarchies: An excerpt from the WIPO patent classification hierarchy (top) and an excerpt from the Enzyme Classification scheme (bottom).

As the model class we use the exponential family

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x},\mathbf{w})} \prod_{e \in E} \exp\left(\mathbf{w}_e^T \phi_e(\mathbf{x}, \mathbf{y}_e)\right) = \frac{1}{Z(\mathbf{x},\mathbf{w})} \exp\left(\mathbf{w}^T \phi(\mathbf{x},\mathbf{y})\right)$$

defined on the edges of a Markov tree $T = (V,E)$, where node $j \in V$ corresponds to the j'th component of the multilabel and the edges $e = (j,j') \in E$ correspond to the classification hierarchy given as input. Above, $Z(\mathbf{x},\mathbf{w}) = \sum_{\mathbf{y}} \exp\left(\mathbf{w}^T \phi(\mathbf{x},\mathbf{y})\right)$ is the normalizing factor also called the partition function. By $\mathbf{y}_e = (y_j, y_{j'})$ we denote the restriction of the multilabel $\mathbf{y} = (y_1, \ldots, y_k)$ to the edge $e = (j,j')$. By $\mathcal{Y}_e = \mathcal{Y}_j \times \mathcal{Y}_{j'}$ we denote the set of labelings of an edge $e = (j,j')$.

In this work, we assume that the Markov tree T is given a priori. This is a reasonable assumption, as hand-made hierarchies and taxonomies are frequent in applications. The ability to learn the structure from data is an important and challenging question, which is out of scope of this article (See Lafferty et al. (2004) for a study to that direction).

Figure 1 depicts examples of two hierarchical classification domains, patent classification according to the World International Patent Organization (WIPO) that is used to classify patent texts, and enzyme classification scheme (EC) used by biologists to classify amino acid sequences for enzymatic proteins.

2.1 Loss Functions for Hierarchical Multilabel Classification

There are many ways to define loss functions for multilabel classification setting, and it depends on the application which loss function is the most suitable. A few general guidelines can be set,

though. The loss function between two multilabel vectors $\mathbf{y}$ and $\mathbf{u}$ should obviously fulfill some basic conditions: $\ell(\mathbf{u},\mathbf{y}) = 0$ if and only if $\mathbf{u} = \mathbf{y}$, $\ell(\mathbf{u},\mathbf{y})$ is maximum when $\mathbf{u}_j \neq \mathbf{y}_j$ for every $1 \leq j \leq k$, and ℓ should be monotonically non-decreasing with respect to the sets of incorrect microlabels. These conditions are satisfied by, for example, *zero-one loss* $\ell_{0/1}(\mathbf{y},\mathbf{u}) = [\mathbf{y} \neq \mathbf{u}]$. However, it gives loss of 1 if the complete hierarchy is not labeled correctly, even if only a single microlabel was predicted incorrectly.

In multilabel classification, we would like the loss to increase smoothly so that we can make a difference between 'nearly correct' and 'clearly incorrect' multilabel predictions. *Symmetric difference loss*

$$\ell_\Delta(\mathbf{y},\mathbf{u}) = \sum_j [y_j \neq u_j],$$

has this property and is an obvious first choice as the loss function in structured classification tasks. However, the classification hierarchy is not reflected in any way in the loss. For *uni-category* hierarchical classification (Hofmann et al., 2003; Cai and Hofmann, 2004; Dekel et al., 2004), where exactly one of the microlabels has value 1, Dekel et al. (2004) use as a loss function the length of the path $(i_1,\cdots,i_k)$ between the the true and predicted nodes with positive microlabels $\ell_{PATH}(\mathbf{y},\mathbf{u}) = |path(i : y_i = 1, j : u_j = 1)|$. Cai and Hofmann (2004) defined a weighted version of the loss that can take into account factors such as subscription loads of nodes.

In the union of partial paths model, where essentially we need to compare a predicted tree to the true one the concept of a path distance is not very natural. We would like to account for the incorrectly predicted subtrees—in the spirit of ℓ_Δ—but taking the hierarchy into account. Predicting the parent microlabel correctly is more important than predicting the child correctly, as the child may deal with some detailed concept that the user may not be interested in; for example whether a document was about CHAMPIONS LEAGUE football or not may not relevant to a person that is interested in FOOTBALL in general. Also, for the learners point of view, if the parent class was already predicted incorrectly, we don't want to penalize the mistake in the child. A loss function that has these properties was given by Cesa-Bianchi et al. (2004). It penalizes the first mistake along a path from root to a node

$$\ell_H(\mathbf{y},\mathbf{u}) = \sum_j c_j [y_j \neq u_j \ \ \& \ \ y_h = u_h \forall h \in anc(j)],$$

where $anc(j)$ denotes the set of ancestors of node j. The coefficients $0 \leq c_j \leq 1$ are used for down-scaling the loss when going deeper in the tree. These can be chosen in many ways. One can divide the maximum loss among the subtrees met along the path. This is done by defining

$$c_{root} = 1, c_j = c_{pa(j)}/|sibl(j)|,$$

where we denoted by $pa(j)$ the immediate parent and by $sibl(j)$ the set of siblings of node j (including j itself). Another possibility is to scale the loss by the proportion of the hierarchy that is in the subtree $T(j)$ rooted by j, that is, to define

$$c_j = |T(j)|/|T(root)|.$$

In our experiments we use both the sibling and subtree scaling to re-weight prediction errors on individual nodes, these are referred to as ℓ-*sibl* and ℓ-*subtree* respectively. If we just use a uniform weighting ($c_j = 1$) in conjunction with the hierarchical loss above this is denoted as ℓ-*unif*.

Using ℓ_H for learning a model has the drawback that it does not decompose very well: the labelings of the complete path are needed to compute the loss. Therefore, in this paper we consider a simplified version of ℓ_H, namely

$$\ell_{\tilde{H}}(\mathbf{y},\mathbf{u}) = \sum_j c_j [y_j \neq u_j \ \& \ y_{pa(j)} = u_{pa(j)}],$$

that penalizes a mistake in a child only if the label of the parent was correct. This choice leads the loss function to capture some of the hierarchical dependencies (between the parent and the child) but allows us define the loss in terms of edges, which is crucial for the efficiency of our learning algorithm.

Using the above, the per-microlabel loss is divided among the edges adjacent to the node. This is achieved by defining an *edge-loss* $\ell_e(\mathbf{y}_e,\mathbf{u}_e) = \ell_j(y_j,u_j)/\mathcal{N}(j) + \ell_{j'}(y_{j'},u_{j'})/\mathcal{N}(j')$ for each $e = (j,j')$, where ℓ_j is the term regarding microlabel j, $\mathbf{y}_e = (y_j,y_{j'})$ is a labeling of the edge e and $\mathcal{N}(j)$ denotes the neighbors of node j in the hierarchy (i.e. the children of a nodes and it's parent). Intuitively, the edges adjacent to node j 'share the blame' of the microlabel loss ℓ_j. The multilabel loss (ℓ_Δ or $\ell_{\tilde{H}}$) is then written as a sum over the edges: $\ell(\mathbf{y},\mathbf{u}) = \sum_{e\in E} \ell_e(\mathbf{y}_e,\mathbf{u}_e)$.

The above described loss functions do not represent an exhaustive list of the possible ones. With probabilistic models, it is common to employ KL-divergence or negative log likelihood as the loss function (Lafferty et al., 2004). In the max-margin learning framework these types of loss functions are not applicable, as they require estimating the underlying probability distribution, e.g. to compute the log-partition function. As our central theme is efficient computation of structured prediction models, we concentrate on the above simpler formulations of loss functions.

2.2 Feature Representations for Structured Inputs

When handling input data that already comes in vector form, there is no obligation to introduce a special kernel function. The inner product of the inputs $K(\mathbf{x},\mathbf{z}) = \mathbf{x}^T\mathbf{z}$, also called the 'linear kernel', can be used. However, when using structured data such as sequences, trees or graphs, one needs to convert the structured representation to a vector form. Feature representation for structured input data have been considered in many works already (c.f. Gartner (2003)), we will concentrate to the important case of hierarchical classification of text or, in general, sequence data.

For sequences the most common feature representation is to count or check the existence of subsequence occurrences, when the subsequences are taken from a fixed index set U. Different choices for the index set and accounting for occurrences give rise to a family of feature representations and kernels. Below we review the main forms of representation for sequences and the computation kernels for such representations.

Word spectrum (Bag-of-words) kernels. In the most widely used feature representation for strings in a natural language, informally called *bag-of-words* (BoW), the index set is taken as the set of words in the language, possibly excluding some frequently occurring stop words (Salton, 1989). The representation was brought to SVM learning by Joachims (1998).

In the case of a string s containing English text, for each English word u, we define the feature value

$$\phi_u(s) = |\{j | s_j \ldots s_{j+|u|-1} = u\}|,$$

as the number of times u occurs in some position j of s. For the example text $s =$ '`The cat was chased by the fat dog`' the BoW will contain the following non-zero entries: $\phi_{\texttt{the}}(s) = 2$,

$\phi_{\texttt{dog}}(s) = 1$, $\phi_{\texttt{was}}(s) = 1$, $\phi_{\texttt{chased}}(s) = 1$, $\phi_{\texttt{by}}(s) = 1$, $\phi_{\texttt{fat}}(s) = 1$, $\phi_{\texttt{cat}} = 1$. These occurrence counts can also be weighted, for example by scaling by the inverse document frequency as is done in TFIDF weighting (c.f. Salton (1989)):

$$\phi_u(s) = |\{j|s_j \dots s_{j+|u|-1} = u\}| \times \log_2 N/N_u,$$

where N_u is the number of documents where u occurs and N is the total number of documents in the collection.

Although the dimension of the feature space may be high, computation of the BoW kernel can be efficiently implemented by scanning the two strings, constructing lists $L(s)$ and $L(t)$ of pairs (u, c_u) of word u and occurrence count c_u ordered in the lexicographical order of the substrings u, and finally traversing the two lists to compute the dot product.

Substring spectrum kernels. For strings that do not encompass a crisply defined word-structure, for example, biological sequences, a different approach is more suitable. Given an alphabet Σ, a simple choice is to take $U = \Sigma^p$, the set of strings of length p. In some cases, using a range of substring lengths $q \leq l \leq p$ may be more appropriate than picking a single length. We can define

$$U = \Sigma^q \cup \Sigma^{q+1} \cdots \cup \Sigma^p \text{ for some } 1 \leq q \leq p.$$

The most efficient approaches, working in $O(p(|s| + |t|))$ time, to compute substring spectrum kernels are based on suffix trees (Leslie et al., 2002; Vishwanathan and Smola, 2002), although dynamic programming and approaches based on the *trie* data structure also can be used Shawe-Taylor and Cristianini (2004).

The substring kernels can be generalized in many ways, for example

- *Gapped substring spectrum kernels* allow gaps in the subsequence occurrences. *Gap-weighting* can be used to down-weight substring occurrences that contain many or long gaps (Lodhi et al., 2002; Rousu and Shawe-Taylor, 2005).
- *Word or syllable alphabets* can be used in place of characters (Saunders et al., 2002; Cancedda et al., 2003).

2.3 Feature Representations for Hierarchical Outputs

When the input features are used in hierarchical classification, they need to be associated with the labelings of the hierarchy. In our setting, this is done via constructing a joint feature map $\phi : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{F}_{xy}$. There are important design choices to be made in how the hierarchical structure should reflect in the feature representation.

There are two general types of features that can be distinguished:

Global features are given by the feature map $\phi^x : \mathcal{X} \mapsto \mathcal{F}_x$. They are not tied to a particular vertex or edge but represent the structured object as a whole. For example, the bag-of-words or the substring spectrum of a document is not tied to a single class of documents in a hierarchy, but a given word can relate to different classes with different importances.

Local features, are given by a feature map $\phi^x_j : \mathcal{X} \mapsto \mathcal{F}_{xj}$ tied to a particular vertex j or edge of the structure. For example, given a structured representation of a scientific article, we can make a difference between elements occurring within the title, abstract, article body and references, and construct local feature maps for each of the components.

Given the input features, there are two basic ways by which the joint feature vector can be constructed:

Orthogonal feature representation is defined as $\phi(\mathbf{x},\mathbf{y}) = (\phi_e(\mathbf{x},\mathbf{y}_e))_{e\in E}$, so that there is a block for each edge (or vertex), which, in turn, is divided into blocks for a specific edge-labeling pairs $(e,\mathbf{u}_e)$, i.e. $\phi_e(\mathbf{x},\mathbf{y}_e) = (\phi_e^{\mathbf{u}_e}(\mathbf{x},\mathbf{y}_e))_{\mathbf{u}_e\in\mathcal{Y}_e}$.

The vector ϕ_e^u should incorporate both the x-features relevant to the edge and encode the dependency on the labeling of the edge. A simple choice is to define

$$\phi_e^{\mathbf{u}_e}(\mathbf{x},\mathbf{y}_e) = [\mathbf{u}_e = \mathbf{y}_e]\left(\phi^x(\mathbf{x})^T, \phi_e^x(\mathbf{x})^T\right)^T$$

that incorporates both the global and local features if the edge is labeled $\mathbf{y}_e = \mathbf{u}_e$, and a zero vector otherwise. Intuitively, the features are turned 'on' only for the particular labeling of the edge that is consistent with $\mathbf{y}$.

Additive feature representation is defined as

$$\phi(\mathbf{x},\mathbf{y}) = \sum_{e\in E}\sum_{\mathbf{u}_e\in\mathcal{Y}_e}[\mathbf{y}_e = \mathbf{u}_e]\phi_e^{\mathbf{u}_e}(\mathbf{x}),$$

where $\phi_e^{\mathbf{u}_e}$ contains features specific to the pair $(e,\mathbf{u}_e)$.

The orthogonal and additive feature representations differ from each other in several respects. In the orthogonal representation, global features get weighted in a context-dependent manner: some features may be more important in labeling one edge than another. Thus, the global features will be 'localized' by the learning algorithm. The size of the feature vectors grow linearly in the number of edges, which requires careful implementation if solving the primal optimization problem (1) instead of the dual. The kernel induced by the above feature map decomposes as

$$K(\mathbf{x},\mathbf{y};\mathbf{x}',\mathbf{y}') = \sum_{e\in E}\phi_e(\mathbf{x},\mathbf{y}_e)^T\phi_e(\mathbf{x}',\mathbf{y}'_e) = \sum_{e\in E}K_e(\mathbf{x},\mathbf{y}_e;\mathbf{x}',\mathbf{y}'_e),$$

which means that there is no crosstalk between the edges:

$$\phi_e(\mathbf{x},\mathbf{y}_e)^T\phi_{e'}(\mathbf{x},\mathbf{y}_{e'}) = 0$$

if $e \neq e'$, hence the name 'orthogonal'. The number of terms in the sum when calculating the kernel obviously scales linearly in the number of edges.

The dimension of the feature vector using the additive feature representation is independent of the size of the hierarchy, thus optimization in primal representation (1) is more feasible for large structures. Second, as there are no feature weights depending on a particular part of the structure, the existence of local features is mandatory, otherwise the output structure is not reflected in the feature vector. Third, the kernel

$$\begin{aligned} K(\mathbf{x},\mathbf{y};\mathbf{x}',\mathbf{y}') &= \left(\sum_e\phi_e(\mathbf{x},\mathbf{y})\right)^T\left(\sum_e\phi_e(\mathbf{x}',\mathbf{y}')\right) \\ &= \sum_{e,e'}\phi_e(\mathbf{x},\mathbf{y}_e)^T\phi_{e'}(\mathbf{x},\mathbf{y}'_{e'}) = \sum_{e,e'}K_{ee'}(\mathbf{x},\mathbf{y}_e;\mathbf{x}',\mathbf{y}'_{e'}) \end{aligned}$$

induced by this representation typically has non-zero blocks $\mathbf{K}_{ee'} \neq 0$, representing cross-talk between edges. There are two consequences of this fact. First, the kernel does not exhibit the sparsity that is implied by the hierarchy, thus it creates the possibility of overfitting. Second, the complexity of the kernel will grow quadratically in the size of the hierarchy rather than linearly as is the case with orthogonal features. This is another reason why a primal optimization approach for this representation might be more justified than a dual approach.

In the sequel, we describe a method that relies on the orthogonal feature representation which will give us a dual formulation with complexity growing linearly in the number of edges in E. The kernel defined by the feature vectors, denoted by

$$K^x(\mathbf{x},\mathbf{x}') = \phi^x(\mathbf{x})^T \phi^x(\mathbf{x}'),$$

is referred to as *x*-kernel while $K(\mathbf{x},\mathbf{y};\mathbf{x},\mathbf{y}')$ is referred to as the *joint kernel*.

2.4 Maximum Margin Learning

Typically in learning probabilistic models, one aims to learn maximum likelihood parameters, which in the exponential CRF amounts to solving

$$\text{argmax}_{\mathbf{w}} \log\left(\prod_{i=1}^{m} P(\mathbf{y}_i|\mathbf{x}_i;\mathbf{w})\right) = \text{argmax}_{\mathbf{w}} \sum_{i=1}^{m} \left[\mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{y}_i) - \log Z(\mathbf{x}_i,\mathbf{w})\right].$$

This estimation problem is hampered by the need to compute the (logarithm of the) partition function Z. For a general graph this problem is hard to solve. Approximation methods for its computation is a subject of active research (c.f. Wainwright and Jordan 2003). Also, in the absence of regularization the max-likelihood model is likely to suffer from overfitting

An alternative formulation (c.f Altun et al. 2003; Taskar et al. 2003), inspired by support vector machines, is to estimate parameters that in some sense maximize the ratio

$$\frac{P(\mathbf{y}_i|\mathbf{x}_i;\mathbf{w})}{P(\mathbf{y}|\mathbf{x}_i;\mathbf{w})}$$

between the probability of the correct labeling $\mathbf{y}_i$ and the worst competing labeling $\mathbf{y}$. With the exponential family, the problem translates to the problem of maximizing the minimum linear margin

$$\mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{y}_i) - \mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{y})$$

in the log-space.

Furthermore, we would like the margin γ to scale as a function of the loss so that grossly incorrect pseudo-examples are pushed farther from the correct labeling than only slightly incorrect ones. Using the canonical hyperplane representation (c.f. Cristianini and Shawe-Taylor (2000)) this can be stated as the following minimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & \mathbf{w}^T\Delta\phi(x_i,\mathbf{y}) \geq \ell(\mathbf{y}_i,\mathbf{y}), \forall i,\mathbf{y} \end{aligned}$$

where $\Delta\phi(x_i,\mathbf{y}) = \phi(x_i,\mathbf{y}_i) - \phi(x_i,\mathbf{y})$. As with SVMs, a model satisfying margin constraints exactly rarely exists, hence it is necessary to add slack variables ξ_i to allow examples to deviate from the margin boundary. Altogether, this results in the following optimization problem

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & \mathbf{w}^T\Delta\phi(x_i,\mathbf{y}) \geq \ell(\mathbf{y}_i,\mathbf{y}) - \xi_i, \forall i,\mathbf{y}. \end{aligned} \tag{1}$$

This optimization problem suffers from the possible high-dimensionality of the feature vectors, for example with string kernels, and from the exponential-sized constraint set (in the length of the multilabel vector). A dual problem

$$\max_{\alpha\geq 0} \alpha^T\ell - \frac{1}{2}\alpha^T\mathbf{K}\alpha, \text{ s.t.} \sum_{\mathbf{y}}\alpha(i,\mathbf{y}) \leq C, \forall i, \tag{2}$$

where $\mathbf{K} = \Delta\Phi^T\Delta\Phi$ is the *joint* kernel matrix for *pseudo-examples* $(x_i,\mathbf{y})$ and $\ell = (\ell(\mathbf{y}_i,\mathbf{y}))_{i,\mathbf{y}}$ is the loss vector, allows us to circumvent the problem with feature vectors. However, in the dual problem there are exponentially many dual variables $\alpha(i,\mathbf{y})$, one for each pseudo-example.

There are a few basic routes by which the exponential complexity can be circumvented:

- Dual working set methods where the constraint set is grown incrementally by adding the worst margin violator

  $$\text{argmin}_{i,\mathbf{y}}\mathbf{w}^T\Delta\phi(\mathbf{x}_i,\mathbf{y}) - \ell(\mathbf{y}_i,\mathbf{y})$$

  to the dual problem. One can guarantee an approximate solution with a polynomial number of support vectors by this approach (Altun et al., 2003; Tsochantaridis et al., 2004).

- Primal methods where the solution above inference problem is integrated to the primal optimization problem, rather than writing down the exponential-sized constraint set (Taskar et al., 2004).

- Marginal dual methods, where the problem is translated to a polynomial-sized form via considering the marginals of the dual variables (Taskar et al., 2003).

The methodology presented in this article belongs to the third category.

2.5 Marginalized Dual Problem

The feasible set of the dual problem (2) is a Cartesian product

$$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_m \tag{3}$$

of identical closed polytopes

$$\mathcal{A}_i = \{\alpha_i \in \mathbb{R}^{|\mathcal{Y}|} \mid \alpha_i \geq 0, ||\alpha_i||_1 \leq C\}, \tag{4}$$

with a vertex set $\mathcal{V}_i = \{0, Ce_1, \ldots, Ce_{|\mathcal{Y}|}\} \subset \mathbb{R}^{|\mathcal{Y}|}$ consisting of the zero vector and the unit vectors of $\mathbb{R}^{|\mathcal{Y}|}$, scaled by C. The vertex set of $\mathcal{A}$ is the Cartesian product $\mathcal{V}_1 \times \cdots \times \mathcal{V}_m$.

The dimension of the set $\mathcal{A}$, $d_{\mathcal{A}} = m|\mathcal{Y}|$ is exponential in the length of the multilabel vectors. This means that optimizing directly over the the set $\mathcal{A}$ is not feasible. Fortunately by utilizing the structure of T, the set $\mathcal{A}$ can be mapped to a set $\mathcal{M}$ of polynomial dimension, called the marginal polytope of H, where optimization becomes more feasible (Taskar et al., 2003).

For an edge $e \in E$ of the Markov tree T, and an associated labeling $\mathbf{y}_e$, the marginal of $\alpha(i,\mathbf{y})$ for the pair $(e,\mathbf{y}_e)$ is given by

$$\mu_e(i,\mathbf{y}_e) = \sum_{\{\mathbf{u}\in\mathcal{Y}_i\}} [\mathbf{y}_e = \mathbf{u}_e]\alpha(i,\mathbf{u}) \tag{5}$$

where the sum picks up those dual variables $\alpha(i,\mathbf{y})$ that have equal value $\mathbf{u}_e = \mathbf{y}_e$ on the edge e. Single node marginals $\mu_j(i,y_j)$ are defined analogously.

For the hierarchy T, the vector containing the edge marginals of the example $\mathbf{x}_i$, the marginal dual vector, is given by

$$\mu_i = (\mu_e(i,\mathbf{u}_e))_{e\in E,\mathbf{u}_e\in\mathcal{Y}_e}.$$

The marginal vector of the whole training set is the concatenation of the single example marginal dual vectors $\mu = (\mu_i)_{i=1}^m$. The vector has dimension $d_{\mathcal{M}} = m\sum_{e\in E}|\mathcal{Y}_e| = O(m|E|\max_e|\mathcal{Y}_e|)$. Thus the dimension is linear in the number of the examples, edges and the maximum cardinality of set of labelings of a single edge.

The indicator functions in (5) can be collectively represented by the the matrix M_E, $M_E(e,\mathbf{u}_e;\mathbf{y}) = [\mathbf{u}_e = \mathbf{y}_e]$, and the relationship between a dual vector alpha and the corresponding marginal vector μ is given by the linear map $M_E\cdot\alpha_i = \mu_i$ and $\mu = (M_E\cdot\alpha_i)_{i=1}^m$. The image of the set $\mathcal{A}_i$, defined by

$$\mathcal{M}_i = \{\mu_i|\ \exists\alpha_i\in\mathcal{A}_i : M_E\alpha_i = \mu_i\}$$

is called the *marginal polytope* of α_i on T.

The following properties of the set $\mathcal{M}_i$ are immediate: Let $\mathcal{A}_i$ be the polytope of (4) and let $\mathcal{M}_i$ be the corresponding marginal polytope. Then

- the vertex set of $\mathcal{M}_i$ is the image of the vertex set of $\mathcal{A}_i$:

$$V_{\mu,i} = \{\mu_i|\ \exists\alpha_i\in V_i : M_E\alpha_i = \mu_i\}.$$

- As an image of a convex polytope $\mathcal{A}_i$ under the linear map M_E, $\mathcal{M}_i$ is a convex polytope.

These properties underlie the efficient solution of the dual problem on the marginal polytope.

The exponential size of the dual problem (2) can be tackled via the relationship between its feasible set $\mathcal{A} = \mathcal{A}_1\times\cdots\times\mathcal{A}_m$ and the marginal polytopes $\mathcal{M}_i$ of each $\mathcal{A}_i$.

Given a decomposable loss function

$$\ell(\mathbf{y}_i,\mathbf{y}) = \sum_{e\in E}\ell_e(i,\mathbf{y}_e)$$

the linear part of the objective satisfies

$$\begin{aligned}
\sum_{i=1}^m\sum_{\mathbf{y}\in\mathcal{Y}}\alpha(i,\mathbf{y})\ell(i,\mathbf{y}) &= \sum_{i=1}^m\sum_{\mathbf{y}}\alpha(i,\mathbf{y})\sum_e\ell_e(i,\mathbf{y}_e)\\
&= \sum_{i=1}^m\sum_{e\in E}\sum_{\mathbf{u}_e\in\mathcal{Y}_e}\sum_{\mathbf{y}:\mathbf{y}_e=\mathbf{u}_e}\alpha(i,\mathbf{y})\ell_e(i,\mathbf{u}_e) = \sum_{i=1}^m\sum_{e\in E}\sum_{\mathbf{u}_e\in\mathcal{Y}_e}\mu_e(i,\mathbf{u}_e)\ell_e(i,\mathbf{u}_e)\\
&= \sum_{i=1}^m\mu_i^T\ell_i = \mu^T\ell_E,
\end{aligned}$$

where $\ell_E = (\ell_i)_{i=1}^m = (\ell_e(i,\mathbf{u}_e))_{i=1,e\in E,\mathbf{u}_e\in\mathcal{Y}_e}^m$ is the marginal loss vector.

Given an orthogonal feature representation inducing a decomposable kernel $K(\mathbf{x},\mathbf{y};\mathbf{x}',\mathbf{y}') = \sum_{e\in E} K_e(\mathbf{x},\mathbf{y}_e;\mathbf{x}',\mathbf{y}'_e)$, the quadratic part of the objective becomes

$$\begin{aligned}
\alpha\mathbf{K}\alpha &= \sum_e \sum_{i,i'} \sum_{\mathbf{y},\mathbf{y}'} \alpha(i,\mathbf{y}) K_e(i,\mathbf{y}_e;i',\mathbf{y}'_e)\alpha(i',\mathbf{y}') \\
&= \sum_e \sum_{i,i'} \sum_{\mathbf{u}_e,\mathbf{u}'_e} K_e(i,\mathbf{u}_e;i',\mathbf{u}'_e) \sum_{\mathbf{y}:\mathbf{y}_e=\mathbf{u}_e} \sum_{\mathbf{y}':\mathbf{y}'_e=\mathbf{u}'_e} \alpha(i,\mathbf{y})\alpha(i',\mathbf{y}') \\
&= \sum_e \sum_{i,i'} \sum_{\mathbf{u}_e,\mathbf{u}'_e} \mu_e(i,\mathbf{u}_e) K_e(i,\mathbf{u}_e;i',\mathbf{u}'_e)\mu_e(i,\mathbf{u}'_e) \\
&= \mu^T \mathbf{K}_E \mu,
\end{aligned}$$

where $\mathbf{K}_E = \mathrm{diag}\,(\mathbf{K}_e, e\in E)$ is a block diagonal matrix with edge-specific kernel blocks $\mathbf{K}_e$.

The objective should be maximized with respect to μ whilst ensuring that there exist $\alpha\in\mathcal{A}$ satisfying $M\alpha_i = \mu_i$ for all i, so that the marginal dual solution represents a feasible solution of the original dual. By the properties outlined above, the feasible set of the marginalized problem is the marginal dual polytope, or to be exact the Cartesian product of the marginal polytopes of single examples (which are in fact equal):

$$\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_m$$

In summary, the marginalized optimization problem can be stated in implicit form as

$$\max_{\mu\in\mathcal{M}} \mu^T \ell_E - \frac{1}{2}\mu^T \mathbf{K}_E \mu$$

This problem is a quadratic programme with a linear number of variables in the number of training examples and in the number of edges.

For optimization algorithms, an explicit characterization of the feasible set is required. Characterizing the polytope $\mathcal{M}$ in terms of linear constraints defining the faces of the polytope, is for general graphs infeasible. Singly-connected graphs such as trees are an exception: for such graphs the marginal polytope is exactly reproduced by the box constraints

$$\sum_{\mathbf{u}_e} \mu_e(i,\mathbf{u}_e) \le C, \forall i, e\in E, \mu_e \ge 0 \tag{6}$$

and the local consistency constraints

$$\sum_{y_k} \mu_{kj}(i,y_k,y_j) = \mu_j(i,y_j); \sum_{y_j} \mu_{kj}(i,y_k,y_j) = \mu_k(i,y_k). \tag{7}$$

In this case the size of the resulting constraint set is linear in the number of vertices the graph. Thus for small hierarchies graphs it can be written down explicitly and the resulting optimization problem has linear size in both the number of examples and the size of the graph. Thus the approach can in principle be made to work, although not with off-the-shelf QP solvers (see sections 3 and 5).

For hierarchies, the consistency constraints (7), can be equivalently defined in terms of the edges: it suffices to pair up each edge with its parent which results in the set of edge pairs $E_2 =$

$\{(e,e') \in E \times E | e = (j',i), e' = (i,j)\}$. By introduction of these marginal consistency constraints the optimization problem gets the form

$$\begin{aligned} \max_{\mu \geq 0} & \sum_{e \in E} \mu_e^T \ell_e - \frac{1}{2} \sum_{e \in E} \mu_e^T \mathbf{K}_e \mu_e \\ \text{s.t} & \sum_{\mathbf{y}_e} \mu_e(i, \mathbf{y}_e) \leq C, \forall i, e \in E, \\ & \sum_{y'} \mu_e(i,(y',y)) = \sum_{y'} \mu_{e'}(i,(y,y')), \ \forall i, y, (e,e') \in E_2, \\ , & \end{aligned} \tag{8}$$

While the above formulation is closely related to that described in Taskar et al. (2003), there are a few differences to be pointed out. Firstly, as we assign the loss to the edges rather than the microlabels, we are able to use richer loss functions than the simple ℓ_Δ. Secondly, single-node marginal dual variables—the μ_j's in (7)—become redundant when the constraints are given in terms of the edges. Thirdly, we have utilized the fact that in our feature representation the 'cross-edge' values $\Delta\phi_e(x, \mathbf{y}_e)^T \Delta\phi_{e'}(x', \mathbf{y}'_{e'})$, where $e \neq e'$, do not contribute to the kernel, hence we have a block-diagonal kernel $\mathbf{K}_E = \text{diag}\,(\mathbf{K}_{e_1}, \ldots, \mathbf{K}_{e_{|E|}}), K_E(i,e,\mathbf{u}_e; j,e,\mathbf{v}_e) = K_e(i,\mathbf{u}_e; j,\mathbf{v}_e)$ with the number of non-zero entries thus scaling linearly rather than quadratically in the number of edges. Finally, we write the box constraint (6) as an inequality as we want the algorithm to be able to inactivate training examples (see Section 3.2).

Like that of Taskar et al. (2003), our approach can be generalized to non-tree structures. However, for a general graph, the feasible region in (8) will only approximate that of (2), which will give rise to a approximate solution to the primal. To arrive at an exact solution, one should construct the junction tree for the graph and to write down the corresponding constraints for the junction tree. As a caveat, one should note that for dense graphs, the junction tree may be significantly larger than the size of the original structure. Also, in tractable time, finding the maximum likelihood multilabel can only be approximated.

3. Efficient Optimization of the Marginalized Dual Problem

While the above quadratic program is polynomial-sized—and considerably smaller than that described in Taskar et al. (2003)—it is still easily too large in practice to fit in main memory or to solve by off-the-shelf QP solvers. To arrive at a more tractable problem, we notice from (3) and (4) that the constraint set decomposes by the examples: to satisfy a single box constraint (6) or a marginal consistency constraint (7) one only needs to change the marginal dual variables of a single example. Moreover, the structure of the feasible set only depends on the edge set E, not on the training example in question: we have $\mathcal{A}_1 = \cdots = \mathcal{A}_m$.

However, the kernel matrix only decomposes by the edges as most pairs of examples have non-positive kernel value between them. Thus there does not seem to be a straightforward way to decompose the quadratic programme.

A decomposition becomes possible when considering gradient-based approaches. Let us consider optimizing the dual variables $\mu_i = (\mu_e(i, \mathbf{y}_e))_{e \in E, \mathbf{y}_e \in \mathcal{Y}_e}$ of example x_i where ℓ_i denotes the corresponding loss vector and $\mathbf{K}_{ij} = \left(K_e(i,\mathbf{u}_e; j, \mathbf{v}_e)_{e \in E, \mathbf{u}_e, \mathbf{v}_e \in \mathcal{Y}_e}\right)$ denotes the block of kernel values between examples i and j, and by $\mathbf{K}_{i\cdot} = (\mathbf{K}_{ij})_{j \in \{1,\ldots,m\}}$ the columns of the kernel matrix $\mathbf{K}_E$ referring to example i.

Obtaining the gradient for the x_i-subspace requires computing the corresponding part of the gradient of the objective function in (8) which is $\mathbf{g}_i = \ell_i - \mathbf{K}_{i\cdot}\mu$ where $\ell_i = (\ell_e(i, \mathbf{u}_e))_{e\in E, \mathbf{u}_e\in\mathcal{Y}_e}$ is the corresponding loss vector for x_i. However, when updating μ_i only, evaluating the change in objective and updating the gradient can be done more cheaply: $\Delta\mathbf{g}_i = -\mathbf{K}_{ii}\Delta\mu_i$ and $\Delta obj = \mathbf{g}_i^T\Delta\mu_i - 1/2\Delta\mu_i\mathbf{K}_{ii}\Delta\mu_i$. Thus local optimization in a subspace of a single training example can be done without consulting the other training examples. On the other hand, we do not want to spend too much time in optimizing a single example: When the dual variables of the other examples are non-optimal, so is the initial gradient $\mathbf{g}_i$. Thus the optimum we would arrive at would not be the global optimum of the quadratic objective. It makes more sense to optimize all examples more or less in tandem so that the full gradient approaches its optimum as quickly as possible.

Before presenting the pseudocode of our method some notations have to be introduced. The function $f()$ denotes the objective function and $\mathcal{F}$ stands for the set of the feasible solutions in (8). The feasibility domain for μ_i when all other components in μ are fixed is denoted by $\mathcal{F}_i$.

In our approach, we have chosen to conduct a few optimization steps for each training example using a conditional gradient ascent (see Algorithm 2) before moving on to the next example. The iteration limit for each example is set by using the Karush-Kuhn-Tucker(KKT) conditions as a guideline (see Section 3.2).

The pseudocode of our algorithm is given in Algorithm 1. It takes as input the training data, the edge set of the hierarchy, the loss vector $\ell = (\ell_i)_{i=1}^m$ and the constraints defining the feasible region. The algorithm chooses a chunk of examples as the working set, computes the kernel for each x_i and makes an optimization pass over the chunk. After one pass, the gradient, slacks and the duality gap are computed and a new chunk is picked. The process is iterated until the duality gap gets below given threshold.

Note in particular, that the joint kernel is not explicitly computed, although evaluating the gradient requires computing the product $\mathbf{K}_E\mu$. However, we are able to take advantage of the special structure of the feature vectors, repeating the same feature vector in different contexts, see the definition of the edge marginal dual variables (5) and the explanation after, to facilitate the computation using the x-kernel $K^x(i,j) = \Delta\phi(x_i)^T\Delta\phi(x_j)$ and the dual variables only.

3.1 Conditional Subspace Gradient Ascent

The optimization algorithm used for a single example is a variant of conditional gradient ascent (or descent) algorithms (Bertsekas, 1999). The algorithms in this family solve a constrained quadratic problem by iteratively stepping to the best feasible direction with respect to the current gradient. It exploits the fact if μ^* is an optimum solution of a maximization problem with objective function f above the feasibility domain $\mathcal{F}_i$ then it has to satisfy the first order optimality condition, i.e., the inequality

$$\nabla f(\mu_i)(\mu_i - \mu^*) \geq 0 \tag{9}$$

has to hold for any feasible μ_i chosen from $\mathcal{F}_i$.

The pseudocode of our variant CSGA is given in Algorithm 2. The algorithm takes as input the current dual variables, gradient, constraints and the kernel block for the example x_i, and an iteration limit. It outputs new values for the dual variables μ_i and the change in objective value. As discussed above, the iteration limit is set very tight so that only a few iterations will be typically conducted.

Algorithm 1 Maximum margin optimization algorithm for the H-M^3 hierarchical classification model.

H-M^3($S, E, \ell, \mathcal{F}$)

Require: Training data $S = ((x_i, \mathbf{y}_i))_{i=1}^m$, edge set E of the hierarchy, a loss vector ℓ, and the feasibility domain $\mathcal{F}$.

Ensure: Dual variable vector μ and objective value $f(\mu)$.

```
Initialize g = ℓ, ξ = ℓ, dg = ∞ and OBJ = 0.
while dg > dg_min & iter < max_iter do
    [WS, Freq] = UpdateWorkingSet(μ, g, ξ);
    Compute x-kernel values K_{X,WS} with respect to the working set;
    for i ∈ WS do
        Compute joint kernel block K_ii and subspace gradient g_i;
        [μ_i, Δobj] = CSGA(μ_i, g_i, K_ii, F_i, Freq_i);
    end for
    Compute gradient g, slacks ξ and duality gap dg;
end while
```

First we need to find a feasible μ^* which maximizes the first order feasibility condition (9) at a fixed μ_i. It gives a direction potentially increasing the value of objective function f. Then we have to choose a step length, τ that gives the optimal feasible solution as a stationary point along the line segment $\mu_i(\tau) = \mu_i + \tau\Delta\mu,\ \tau \in (0,1]$, where $\Delta\mu = \mu^* - \mu_i$, starting on the known feasible solution μ_i.

The stationary point is found by solving the equation

$$\frac{\mathbf{d}}{\mathbf{d}\tau}\left[\ell_i^T \mu_i(\tau) - 1/2\mu_i(\tau)^T \mathbf{K}_{ii}\mu_i(\tau)\right] = 0, \tag{10}$$

expressing the optimality condition with respect to τ. If $\tau > 1$, the stationary point is infeasible and the feasible maximum is obtained at $\tau = 1$. In our experience, the time taken to compute the stationary point was typically significantly smaller than time taken to find μ_i^*, depending on the dataset characteristics and the actual algorithm (see Section 3.3) that was used to find μ_i^*.

3.2 Working Set Maintenance

We wish to maintain the working set so that the most promising examples to be updated are contained there at all times to minimize the amount of computation used for unsuccessful updates. Our working set update is based on the Karush-Kuhn-Tucker(KKT) conditions which at the optimum hold for all x_i:

1. $(C - \sum_{e,\mathbf{y}_e} \mu_e(i,\mathbf{y}_e))\xi_i = 0$, and
2. $\alpha(i,\mathbf{y})(\mathbf{w}^T\phi(x_i,\mathbf{y}) - \ell(\mathbf{y}_i,\mathbf{y}) + \xi_i) = 0$.

The first condition states that, at optimum, only examples that saturate the box constraint can have positive slack, and consequently a pseudo-example that has a negative margin. The second condition states that pseudo-examples with non-zero dual variables are those that have the minimum margin, that is, need the full slack ξ_i. Consequently, if all pseudo-examples of x_i have positive margin, all dual variables satisfy $\alpha(i,\mathbf{y}) = 0$. This observation leads to the following heuristics for the working set update:

Algorithm 2 Conditional subspace gradient ascent optimization step.

CSGA($\mu_i, \mathbf{g}_i, \mathbf{K}_{ii}, \mathcal{F}_i, maxiter_i$)

Require: Initial dual variable vector μ_i, gradient $\mathbf{g}_i$, constraints of the feasible region $\mathcal{F}_i$, a joint kernel block $\mathbf{K}_{ii}$ for the subspace, and an iteration limit $maxiter_i$.

Ensure: New values for dual variables μ_i and change in objective Δobj.

$\Delta obj = 0; iter = 0;$
while $iter < maxiter$ **do**
  % find highest feasible point given $\mathbf{g}_i$
  $\mu^* = \text{argmax}_{v \in \mathcal{F}_i} \mathbf{g}_i^T v;$
  $\Delta\mu = \mu^* - \mu_i;$
  $q = \mathbf{g}_i^T \Delta\mu,\ r = \Delta\mu^T \mathbf{K}_{ii} \Delta\mu;$ % taken from the solution of (10)
  $\tau = \min(q/r, 1);$ % clip to remain feasible
  if $\tau \leq 0$ **then**
    break; % no progress, stop
  else
    $\mu_i = \mu_i + \tau\Delta\mu;$ % update
    $\mathbf{g}_i = \mathbf{g}_i - \tau\mathbf{K}_{ii}\Delta\mu;$
    $\Delta obj = \Delta obj + \tau q - \tau^2 r/2;$
  end if
  $iter = iter + 1;$
end while

- Non-saturated ($\sum_{e,\mathbf{y}_e} \mu_e(i,\mathbf{y}_e) < C$) examples are given priority as they certainly will need to be updated to reach the optimum.

- Saturated examples ($\sum_{e,\mathbf{y}_e} \mu_e(i,\mathbf{y}_e) = C$) are added if there are not enough non-saturated ones. The rationale is that the even though an example is saturated, the individual dual variable values may still be suboptimal being equal to 0.

- Inactive ($\sum_{e,\mathbf{y}_e} \mu_e(i,\mathbf{y}_e) = 0$) non-violators ($\xi_i = 0$) are removed from the working set, as they do not constrain the objective.

Another heuristic technique to concentrate computational effort to most promising examples is to favor examples with a large duality gap

$$\Delta obj(\mu, \xi) = \sum_i C\xi_i + \mu_i^T \mathbf{g}_i.$$

As feasible primal solutions always are least as large as feasible dual solutions, the duality gap gives an upper bound to the distance from the dual solution to the optimum. We use the quantity $\Delta_i = C\xi_i + \mu_i^T \mathbf{g}_i$ as a heuristic measure of the work needed for that particular example in order to reach the optimum. Examples are then chosen to the chunk to be updated with probability proportional to $p_i \propto \Delta_i - \min_j \Delta_j$. An example that is drawn more than once will be set a higher iteration limit for the next optimization step.

3.3 Finding Update Directions Efficiently

The optimization algorithm described above relies on efficient computation of update directions μ_i^* in the single example subspaces, that is, to solve the constrained linear program

$$\text{argmax}_{\mathbf{v}\in\mathcal{F}_i}\ \mathbf{g}_i^T\mathbf{v}. \tag{11}$$

A straightforward approach would be to use a linear programming solver, such as the LIPSOL interior point solver. However, a such black-box approach does not utilize the special structure of the problem in any way.

In order to solve this problem efficiently, we first notice two things:

1. A vertex of the feasible set is always among the optimal solutions.

2. Vertices correspond to consistent labelings of the hierarchy. This can be seen from the fact that at the vertex, for each edge $\mu_e(i,\mathbf{y}_e) = C$ for exactly one $\mathbf{y}_e$ and $\mu_e(i,\mathbf{u}_e) = 0$ for $\mathbf{u}_e \neq \mathbf{y}_e$, and that the marginal consistency constraints require that for two adjacent edges $e' = (j', j), e'' = (j, j'')$ we have $\mu_{e'}(i,\mathbf{y}'_e) = C = \mu_{e''}(i,\mathbf{y}''_e)$ with matching edge-labelings $\mathbf{y}'_e = (y_{j'}, y_j)$ and $\mathbf{y}''_e = (y_j, y_{j''})$.

Thus instead of solving (11) directly, we can search for the labeling $\mathbf{y}_*$ of the hierarchy corresponding to an optimal vertex
$vmu(\mathbf{y}_*)$ of the feasible set:

$$\text{argmax}_{\mathbf{y}\in\mathcal{Y}}\ \mathbf{g}_i^T\mu(\mathbf{y}) \tag{12}$$

This problem can be solved efficiently using a dynamic programming inference algorithm, reviewed in the next section.

3.4 Solving the Inference Problem in Linear Time

When dealing with structured output models, one needs to solve the inference problem

$$\text{argmax}_{\mathbf{y}\in\mathcal{Y}}\ \mathbf{g}_i^T\mu(\mathbf{y}) \tag{13}$$

to find a multilabel $\mathbf{y}$ maximizing the inner product between some (gradient) vector h and the marginal dual variables $\mu(\mathbf{y})$ corresponding to $\mathbf{y}$. In our learning scheme this problem is found in two situations,

- when predicting multilabels given a learned model, and

- to find update directions (12).

The algorithm described below can be used for both problems, the only quantity that changes is the gradient $\mathbf{g}_i$.

Inference algorithms solving problems of the above form have been well-studied in the literature of probabilistic models, under the names of belief propagation and generalized distributive law (Aji and McEliece, 2000; Kschischang et al., 2001; Wainwright and Jordan, 2003). It is known that, for general graphs, solving (13) is not any easier than solving (11). However, for a hierarchical model dynamic programming can be used: starting from the leaves of the hierarchy we compute bottom-up

for each subtree the optimal labeling of the subtree, conditioned on fixing the label of the subtree root to $+1$ or -1.

We denote by $T_j = (V_j, E_j)$ the subtree of T rooted at node j. We need to maintain two quantities during the bottom-up pass:

- The best objective value that can be obtained for the example i in the subtree rooted at node j when the label y_j has been fixed. We denote this value by $S_{y_j}(i, j)$.
- The best objective value that can be obtained for the subtree rooted by the edge $e = (j, j')$ when the root node j is fixed to y_j. We denote this value by $G_{y_j}(i, e)$.

The two quantities are computed from the recurrences

$$S_{y_j}(i,j) = \begin{cases} \sum_{e=(j,j')\in E_j} G_{y_j}(i,e), & \text{if } E_j \neq \emptyset, \text{and} \\ 0, & \text{otherwise ,} \end{cases}$$

and

$$G_{y_j}(i,e) = \max_{y_{j'}} g_e(i,y_j,y_{j'})\mu_e(i,y_j,y_{j'}) + S_{y_{j'}}(i,j')$$

At the root node of the hierarchy, $\max_y S_y(i, root)$ finally gives the optimum. The corresponding vertex $v(\mathbf{y}_*)$ is found in making a top-down pass over the hierarchy: one looks for best label for a child of a node given the parent has been fixed. It should be noted that although in principle the best conditional labeling—how to label a subtree when the root is fixed to one of the possible labels—could be computed already during the bottom-up pass, the two pass algorithm, where the labeling is worked out only after the label of the global root of the hierarchy has been found out, is much easier to implement and works just as fast.

The dynamic programming scheme can be implemented in vectorized form so that all examples and all nodes on a level of the hierarchy are handled at the same time, thus eliminating the need for loops going over examples and nodes, which in MATLAB implementation are to be avoided.

All in all, the above described inference algorithm works in linear time in the number of dual variables, which can be seen from the fact that each example is processed once, each edge is visited twice (once in the bottom-up pass, once in the top-down pass) and the max operations are taken over the dual variables belonging to the current edge.

3.5 Computing Stationary Points in Linear Time

The conditional gradient ascent requires us to iteratively solve (10) for τ, which gives $\tau = \Delta\mu / \Delta\mu \mathbf{K}_{ii} \Delta\mu$. The potentially expensive part is evaluating the matrix-vector product $\mathbf{K}_{ii}\Delta\mu = \mathbf{K}_{ii}\mu^* - \mathbf{K}_{ii}\mu_i$, which trivially could take quadratic time in the number of variables. However, we can keep in memory the vector $\mathbf{K}_{ii}\mu_i$ during the computation, thus it remains to compute $\mathbf{K}_{ii}\mu^*$. Firstly, we notice that for a normalized x-kernel, the entries of the joint kernel are given as sums of indicators $K_{ii}(e, \mathbf{u}_e; e, \mathbf{u}'_e) = 1 - [\mathbf{y}_{ie} = \mathbf{u}'_e] - [\mathbf{y}'_{ie} = \mathbf{u}_e] + [\mathbf{u}_e = \mathbf{u}'_e]$. Secondly, since μ^* is an extreme point of the feasible set, $\mu^*(e, \mathbf{u}_e) = C$ for exactly one of the components $\mathbf{u}_e \in \mathcal{Y}_e$. By these facts and some arithmetic manipulation we obtain $\mathbf{K}_{ii}\mu^* = [\mathbf{1} - \mathbf{y}_i]C - \mathbf{y}_i \cdot \mu^* + \mu^*$. Thus, instead of matrix-vector product we only need to compute a single vector-vector product and a sum of three vectors. Finally, the update for $\mathbf{K}_{ii}\mu_i$ is given as a convex combination of vectors $\mathbf{K}_{ii}\mu_i^{new} = \tau\mathbf{K}_{ii}\mu^* + (1-\tau)\mathbf{K}_{ii}\mu_i$. The total number of operations to compute the stationary point remains linear in the number of variables.

4. Extensions and Variants

There are several variations of the multilabel classification models described above.

Slack variables were defined as non-negative and a single variable was allocated per example. Allowing negative slack (c.f.Taskar et al. (2003); Tsochantaridis et al. (2004)) results in the dual equality constraint $\sum_{\mathbf{y}_e} \mu_e(i,\mathbf{y}_e) = C$ instead of the box constraint. This results in non-sparse models as training points are very likely to have non-zero slack.

Allotting a separate slack variable for each edge is a possibility when the data for some edges can be considered less reliable than the data for others; in such case the unreliable edge can consume required slack without affecting the other edges. From an optimization point of view, edge-based slack variables make the model decompose into separate edge-based quadratic programs and may allow larger models to be optimized.

Partial paths could be used as the basis of the classification model instead of the edges. For each partial path $p = (j_1,\ldots,j_d)$ one defines a feature vector $\phi_p(i,\mathbf{y}) = [\mathbf{y}_p = \mathbf{1}_{|p|}]\phi(x)$, where $\mathbf{y}_p = (y_{j_1},\ldots,y_{j_d})$ is the restriction of the multilabel to the partial path. As the number of partial paths in the hierarchy equals the number of nodes, the resulting feature vectors are actually smaller than the ones defined by edge-labelings. The marginalization of the model by the partial paths works in an analogous way to the edge-marginalization and the same optimization algorithms can be used. The price of the more compact feature representation comes in the form of slightly more complicated consistency constraints and inference: For consistency one needs to ensure that if a partial path p has non-zero path-marginal $\mu_p(i,\mathbf{y}_p)$, no prefix p' of p has non-zero marginal $\mu_{p'}(i,\mathbf{y}_p)$. Correspondingly, the inference algorithms need to make comparisons between a partial path and its prefixes.

Non-hierarchical models can also be tackled with the above described framework, with a few caveats. First, ensuring global consistency of the marginalized dual is more involved as local consistency of edge-marginals does not guarantee existence of a dual variable $\alpha(i,\mathbf{y})$ with those marginals. If the graph is not too dense this problem can be circumvented by computing the clique tree of the graph and making the clique tree locally consistent, and the conditional gradient optimization will work unmodified. However, inference for general graphs is NP-hard so both computing predictions of the model and finding the update directions in the optimization becomes hard. Several schemes to find approximate solutions exist, including loopy belief propagation, semi-definite relaxations and tree-based approximations (Wainwright and Jordan (2003); Wainwright et al. (2003)). Depending on the application, also considering the model in a decomposed form via definition of edge-slack variables (see above) may be justified.

5. Experiments

We tested the presented learning approach on three datasets that have an associated classification hierarchy:

- REUTERS Corpus Volume 1, RCV1 (Lewis et al., 2004). 2500 documents were used for training and 5000 for testing. As the label hierarchy we used the 'CCAT' family of categories (Corporate/Industrial news articles), which had a total of 34 nodes, organized in a tree with maximum depth 3. The tree is quite unbalanced, half of the nodes residing in depth 1, and very few nodes in depth 3.

- WIPO-alpha patent dataset (WIPO, 2001). The dataset consisted of the 1372 training and 358 testing document comprising the D section of the hierarchy. The number of nodes in the hierarchy was 188, with maximum depth 3.

- ENZYME classification dataset. The training data consisted of 7700 protein sequences with hierarchical classification given by the Enzyme Classification (EC) system. The hierarchy consisted of 236 nodes organized into a tree of depth three. Test data consisted of 1755 sequences.

In all datasets, the membership of examples in the nodes of the hierarchy is indicated by binary vectors $\mathbf{y} \in \{+1,-1\}^k$. Multiple paths were actually present in one of the datasets, REUTERS, approximately 8 percent of examples were classified into more than one category.

The two first datasets were processed into bag-of-words representation with TFIDF weighting. No word stemming or stop-word removal was performed. For the ENZYME sequences a length-4 subsequence kernel was used.

We compared the performance of the presented learning approach—below denoted by H-M^3—to three algorithms: SVM denotes an SVM trained for each microlabel separately, H-SVM denotes the case where the SVM for a microlabel is trained only with examples for which the ancestor labels are positive.

The SVM and H-SVM were run using the SVM-light package. After pre-computation of the kernel these algorithms are as fast as one could expect, as they just involve solving an SVM for each node in the graph (with the full training set for SVM and usually a much smaller subset for H-SVM).

H-RLS is a batch version of the hierarchical least squares algorithm described in Cesa-Bianchi et al. (2004). It essentially solves for each node i a least squares style problem $\mathbf{w}_i = (I + S_i S_i^T + \mathbf{x}\mathbf{x}^T)^{-1} S_i \mathbf{y}_i$, where S_i is a matrix consisting of all training examples for which the parent of node i was classified as positive, $\mathbf{y}_i$ is a microlabel vector for node i of those examples and I is the identity matrix. Predictions for a node i for a new example x is -1 if the parent of the node was classified negatively and $\text{sign}(\mathbf{w}_i^T \mathbf{x})$ otherwise.

H-RLS requires a matrix inversion for each prediction of each example, at each node along a path for which errors have not already been made. No optimization of the algorithm was done, except to use extension approaches to efficiently compute the matrix inverse (for each example an inverted matrix needs to be extended by one row/column, so a straightforward application of the Sherman-Morrison formula to efficiently update the inverse can be used).

The H-RLS and H-M^3 algorithms were implemented in MATLAB. The tests were run on a high-end PC. For SVM,H-SVM and H-M^3, the regularization parameter value $C = 1$ was used in all experiments.

Obtaining consistent labelings. As the learning algorithms compared here all decompose the hierarchy for learning, the multilabel composed of naively combining the microlabel predictions may be inconsistent, that is, they may predict a document as part of the child but not as part of the parent. For SVM and H-SVM consistent labelings were produced by post-processing the predicted labelings as follows: start at the root and traverse the tree in a breadth-first fashion. If the label of a node is predicted as -1 then all descendants of that node are also labeled negatively. This post-processing turned out to be crucial to obtain good accuracy, thus we only report results with the postprocessed labelings. Note that H-RLS performs essentially the same procedure (see above). For H-M^3 models, we computed by dynamic programming the consistent multilabel with maximum

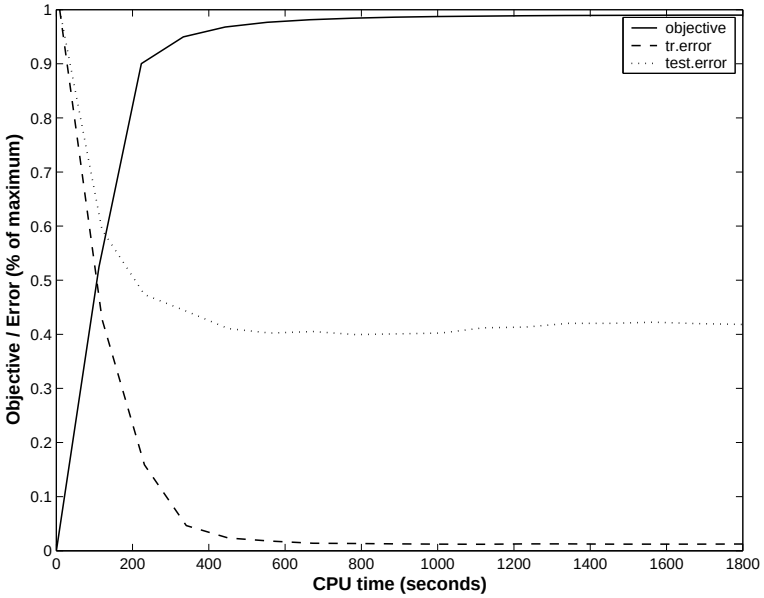


Figure 2: The objective function (% of optimum) and ℓ_Δ losses for H-M^3 on training and test sets (WIPO-alpha)

likelihood

$$\mathbf{u}(\mathbf{x}) = \text{argmax}_{\mathbf{y}\in\mathcal{Y}_T} P(\mathbf{y}|x) = \text{argmax}_{\mathbf{y}} \mathbf{w}^T \phi(\mathbf{x},\mathbf{y}),$$

where $\mathcal{Y}_T$ is the set of multilabels that correspond to unions of partial paths in T. The algorithm is otherwise the same as the one in 3.4, but the inconsistent edge-labelings are not taken into account in the maximization.

Efficiency of optimization. To give an indication of the efficiency of the H-M^3 algorithm, Figure 2 shows an example learning curve on WIPO-alpha dataset. The number of dual variables for this training set is just over one million with a joint kernel matrix with approx 5 billion entries. Note that the solutions for this optimization are not sparse, typically less than 25% of the marginal dual variables are zero. Training and test losses (ℓ_Δ) are all close to their optima within 10 minutes of starting the training, and the objective is within 2 percent of the optimum in 30 minutes.

To put these results in perspective, for the WIPO data set SVM (SVM-light) takes approximately 50 seconds per node, resulting in a total running time of about 2.5 hours, which makes it significantly slower than H-M^3, in these tests. It is possible that using early stopping for SVM the training time could be pushed down to the level of H-M^3, however, we have not explored this question. We also suspect that early stopping for SVM may be more costly than for H-M^3, due to the fact that the latter predicts whole labelings for the trees where the weight of a single microlabel is small, and in fact the predicted multilabels may contain microlabels that are not locally optimal. In other words, the inference procedure for multilabels may correct poor microlabel predictions.

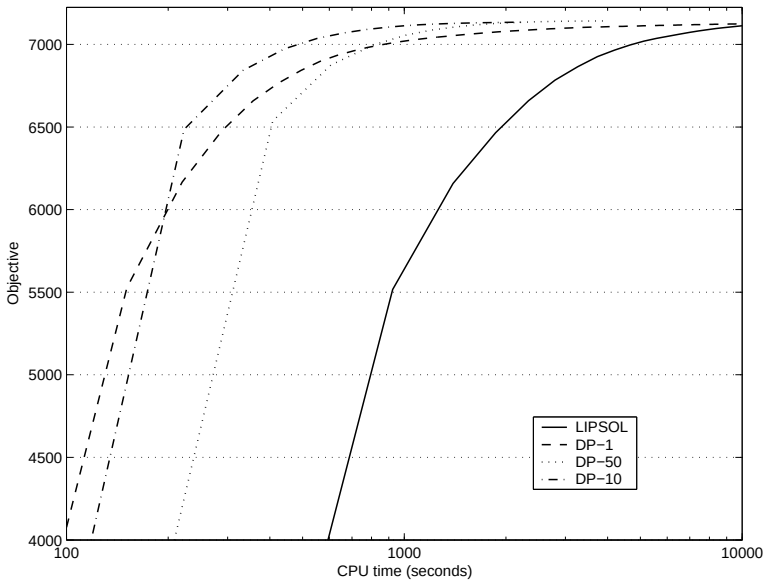


Figure 3: Learning curves for H-M^3 using LIPSOL and dynamic programming (DP) to compute update directions (WIPO-alpha). Curves with iteration limits 1,10 and 50 are shown for DP. The LIPSOL curve is computed with iteration limit set to 1.

The running time of H-RLS was slower than the other methods, however this could be due to our unoptimized implementation. It is our expectation that it would be very close to the time taken by H-SVM if coded more efficiently.

Therefore, the methods presented in this paper are very competitive from a computational efficiency point of view to other methods which do not operate in the large feature/output spaces of H-M^3.

Figure 3 shows on WIPO-alpha the efficiency of the dynamic programming (DP) based computation of update directions as compared to solving the update directions with MATLAB's linear interior point solver LIPSOL. The DP based updates result in an order of magnitude faster optimization than using LIPSOL.

In addition for DP the effect of the iteration limit for optimization speed is depicted. Setting the iteration limit too low (1) or too high (50) slows down the optimization, for different reasons. A too tight iteration limit makes the overhead in moving from one example to the other dominate the running time. A too high iteration limit makes the the algorithm spend too much time optimizing the dual variables of a single example. Unfortunately, it is not straightforward to suggest a iteration limit that would be universally the best, as the optimal value depends on the dataset.

Effect of choice of the loss function. In order to show the effect of training the H-M^3 algorithm using the different loss functions described in Section 2.1, we studied the performance of the algorithm on Reuters and WIPO data sets. The results can be seen in Table 5. The WIPO dataset gives an indication that using a hierarchical loss function during training (e.g. either $\ell_{\tilde{H}}$-sibl. or

	Test loss				
	$\ell_{0/1}$	ℓ_Δ	$\ell_{\tilde{H}}$ +scaling		
Tr. loss	%		unif	sibl.	subtree
ℓ_Δ	27.1	0.574	0.344	0.114	0.118
$\ell_{\tilde{H}}$-unif	26.8	0.590	0.338	0.118	0.122
$\ell_{\tilde{H}}$-sibl.	28.2	0.608	0.381	0.109	0.114
$\ell_{\tilde{H}}$-subtree	27.9	0.588	0.373	0.109	0.109
	$\ell_{0/1}$	ℓ_Δ	$\ell_{\tilde{H}}$ +scaling		
Tr. loss	%		unif	sibl.	subtree
ℓ_Δ	70.9	1.670	0.891	0.050	0.070
$\ell_{\tilde{H}}$-unif.	70.1	1.721	0.888	0.052	0.074
$\ell_{\tilde{H}}$-sibl.	64.8	1.729	0.927	0.048	0.071
$\ell_{\tilde{H}}$-subtree	65.0	1.709	0.919	0.048	0.072

Table 1: Prediction losses obtained using different training losses on Reuter's (top) and WIPO-alpha data (bottom). The loss $\ell_{0/1}$ is given as a percentage, the other losses as averages per-example.

$\ell_{\tilde{H}}$-subtree) may lead to a reduced $0/1$ loss on the test set. On Reuters dataset this effect is not observed, however this is due to the fact that the label tree of the Reuters data set is very shallow.

Comparison of predictive accuracies of different algorithms. In our final test we compare the predictive accuracy of H-M^3 to other learning methods. For H-M^3 we include the results for training with ℓ_Δ and $\ell_{\tilde{H}}$-*subtree* losses. For training SVM and H-SVM, these losses produce the same learned model.

Table 2 depicts the different test losses, as well as the standard information retrieval statistics precision (P), recall (R) and F1 statistic ($F1 = 2PR/(P+R)$). Precision and recall were computed over all microlabel predictions in the test set. Flat SVM is expectedly inferior to the competing algorithms with respect to most statistics, as it cannot utilize the dependencies between the microlabels in any way. The two variants of H-M^3 are the most efficient in getting the complete tree correct as shown by the lower zero-one loss. With respect to other statistics, the hierarchical methods are quite evenly matched overall.

Finally, to highlight the differences between the predicted labelings, we computed level-wise precision and recall values, that is, the set of predictions contained all test instances and microlabels on a given level of the tree (Table 3). On all datasets, recall of all methods, especially with SVM and H-SVM, diminishes when going farther from the root. H-M^3 is the most efficient method in fighting the recall decline, and is still able to obtain reasonable precision on REUTERS and WIPO-alpha, especially when trained with the hierarchical loss.

The results on ENZYME data are generally not good for any of the methods, this is most probably due to the subsequence kernel used not being able to pick out the subsequences corresponding to the active centers of the enzymes. Nevertheless, the effect of H-M^3 obtaining better recall in deep nodes than the competition can be observed.

REUTERS	$\ell_{0/1}$	ℓ_Δ	P	R	F1
SVM	32.9	0.61	94.6	58.4	72.2
H-SVM	29.8	0.57	92.3	63.4	75.1
H-RLS	28.1	**0.55**	91.5	65.4	**76.3**
H-M^3-ℓ_Δ	**27.1**	0.58	91.0	64.1	75.2
H-M^3-$\ell_{\tilde{H}}$	27.9	0.59	85.4	68.3	75.9
WIPO-alpha	$\ell_{0/1}$	ℓ_Δ	P	R	F1
SVM	87.2	1.84	93.1	58.2	71.6
H-SVM	76.2	1.74	90.3	63.3	74.4
H-RLS	72.1	1.69	88.5	66.4	75.9
H-M^3-ℓ_Δ	70.9	**1.67**	90.3	65.3	75.8
H-M^3-$\ell_{\tilde{H}}$	**65.0**	1.73	84.1	70.6	**76.7**
ENZYME	$\ell_{0/1}$	ℓ_Δ	P	R	F1
SVM	99.7	1.3	99.6	41.1	58.2
H-SVM	98.5	**1.2**	98.9	41.7	58.7
H-RLS	95.6	2.0	51.9	54.7	53.3
H-M^3-ℓ_Δ	95.7	**1.2**	87.0	49.8	**63.3**
H-M^3-$\ell_{\tilde{H}}$	**85.5**	2.5	44.5	66.7	53.4

Table 2: Prediction losses $\ell_{0/1}$ and ℓ_Δ, precision, recall and F1 values obtained using different learning algorithms. All figures are given as percentages. Precision and recall are computed in terms of totals of microlabel predictions in the test set.

6. Conclusions and Future Work

In this paper we have proposed a new method for training variants of the Maximum Margin Markov Network framework for hierarchical multi-category text classification models.

Our method relies on a decomposition of the problem into single-example sub problems and conditional gradient ascent for optimisation of the subproblems. The method scales well to medium-sized datasets with label matrix (examples $\times$ microlabels) size upto hundreds of thousands, and via kernelization, very large feature vectors for the examples can be used. Experimental results on three classification tasks show that using the hierarchical structure of multi-category labelings leads to improved performance over the more traditional approach of combining individual binary classifiers.

Our future work includes generalization of the approach to general graph structures and looking for ways to scale up the method further.

REUTERS	Level 0	Level 1	Level 2	Level 3
SVM	92.4/89.4/90.9	96.8/38.7/55.3	98.1/49.3/65.6	81.8/46.2/59.0
H-SVM	92.4/89.4/90.9	93.7/43.6/59.5	91.1/61.5/73,4	72.0/46.2/56,3
H-RLS	93.2/89.1/**91.1**	90.9/46.8/61.8	89.7/64.8/**75.2**	76.0/48.7/59.4
H-M^3-ℓ_Δ	94.1/83.0/88.2	87.3/48.9/62.7	91.1/63.2/74.6	79.4/69.2/73.9
H-M^3-$\ell_{\tilde{H}}$	91.1/87.8/89.4	79.2/53.1/**63.6**	85.4/66.6/74.8	77.9/76.9/**77.4**
WIPO-alpha	Level 0	Level 1	Level 2	Level 3
SVM	100/100/100	92.1/77.7/84.3	84.4/42.5/56.5	82.1/12.8/22.1
H-SVM	100/100/100	92.1/77.7/84.3	79.6/51.1/62.2	77.0/24.3/36.9
H-RLS	100/100/100	91.3/79.1/84.8	78.2/57.0/65.9	72.6/29.6/42.1
H-M^3-ℓ_Δ	100/100/100	90.8/80.2/85.2	86.1/50.0/63.3	72.1/31.0/43.4
H-M^3-$\ell_{\tilde{H}}$	100/100/100	90.9/80.4/**85.3**	76.4/62.3/**68.6**	60.4/39.7/**47.9**
ENZYME	Level 0	Level 1	Level 2	Level 3
SVM	100/100/100	84.3/4.9/9.3	100/0.4/0.8	100/0.3/0.6
H-SVM	100/100/100	84.3/4.9/9.3	72.3/1.9/3.7	67.5/1.5/2.9
H-RLS	100/97.4/98.7	33.0/39.3/35.9	22.4/22.6/22.5	15.2/17.0/16.0
H-M^3-ℓ_Δ	100/100/100	61.2/30.8/41.0	49.8/13.3/21.0	52.9/4.7/8.6
H-M^3-$\ell_{\tilde{H}}$	100/100/100	49.3/56.0/**52.4**	21.5/42.5/**28.6**	14.7/35.2/**20.7**

Table 3: Precision/Recall/F1 statistics for each level of the hierarchy for different algorithms on Reuters RCV1 (top), WIPO-alpha (middle), and ENZYME datasets (bottom).

Acknowledgments

The authors gratefully acknowledge the insightful comments by the anonymous referees. We also wish to thank Esa Pitkänen for his help in preparing the datasets. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. Juho Rousu has been supported by the European Union Marie Curie Fellowship grant HPMF-CT-2002-02110 and the work was partly conducted when he was visiting Royal Holloway University of London.

References

S. M. Aji and R. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *International Conference of Machine Learning*, 2003.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *13 ACM CIKM*, 2004.

N. Cancedda, E. Gaussier, C. Goutte, and J.-M. Renders. Word-sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082, 2003.

N. Cesa-Bianchi, C. Gentile, A. Tironi, and L. Zaniboni. Incremental algorithms for hierarchical classification. In *Neural Information Processing Systems*, 2004.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *ICML'04*, pages 209–216, 2004.

S. T. Dumais and H. Chen. Hierarchical classification of web content. In *SIGIR'00*, pages 256–263, 2000.

T. Gartner. A survey of kernels for structured data. *ACM SIGKDD Explorations*, pages 49–58, 2003.

T. Hofmann, L. Cai., and M. Ciaramita. Learning with taxonomies: Classifying documents and words. In *NIPS Workshop on Syntax, Semantics, and Statistics*, 2003.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137 – 142, Berlin, 1998. Springer.

D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *ICML'97*, pages 170–178, 1997.

F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 2001.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: representation and clique selection. In *Proc. 21th International Conference on Machine Learning*, pages 504–511, 2004.

C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564 – 575, 2002.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, Apr 2004.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, February 2002.

A. McCallum, R. Rosenfeld, T. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML'98*, pages 359–367, 1998.

J. Rousu and J. Shawe-Taylor. Efficient computation of gapped substring kernels on large alphabets. *JMLR*, 6:1323–1344, 2005.

G. Salton. *Automatic Text Processing*. Addison-Wesley, Massachusetts, 1989.

C. Saunders, H. Tschach, and J. Shawe-Taylor. Syllables and other string kernel extensions. In *ICML'02*, pages 530–537, 2002.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Neural Information Processing Systems 2003*, 2003.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proc. 21th International Conference on Machine Learning*, pages 807–814, 2004.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y.n Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21th International Conference on Machine Learning*, pages 823–830, 2004.

S. V. N. Vishwanathan and A. J. Smola. Fast kernels on strings and trees. In *Proceedings of Neural Information Processing Systems 2002*, 2002. URL `http://users.rsise.anu.edu.au/ vishy/papers/VisSmo02.pdf`.

M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, Department of Statistics, University of California, Berkeley, 2003.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on information theory*, 49:1120–1146, May 2003.

WIPO. *World Intellectual Property Organization. http://www.wipo.int/classifications/en*. 2001.

Journal of Machine Learning Research 7 (2006) 1627–1653 Submitted 10/05; Published 7/06

Structured Prediction, Dual Extragradient and Bregman Projections

Ben Taskar TASKAR@CS.BERKELEY.EDU
Simon Lacoste-Julien SLACOSTE@CS.BERKELEY.EDU
Computer Science
University of California
Berkeley, CA 94720, USA

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science and Statistics
University of California
Berkeley, CA 94720, USA

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

We present a simple and scalable algorithm for maximum-margin estimation of structured output models, including an important class of Markov networks and combinatorial models. We formulate the estimation problem as a convex-concave saddle-point problem that allows us to use simple projection methods based on the dual extragradient algorithm (Nesterov, 2003). The projection step can be solved using dynamic programming or combinatorial algorithms for min-cost convex flow, depending on the structure of the problem. We show that this approach provides a memory-efficient alternative to formulations based on reductions to a quadratic program (QP). We analyze the convergence of the method and present experiments on two very different structured prediction tasks: 3D image segmentation and word alignment, illustrating the favorable scaling properties of our algorithm.[1]

Keywords: Markov networks, large-margin methods, structured prediction, extragradient, Bregman projections

1. Introduction

Structured prediction problems are classification or regression problems in which the output variables (the class labels or regression responses) are interdependent. These dependencies may reflect sequential, spatial, recursive or combinatorial structure in the problem domain, and capturing these dependencies is often as important for the purposes of prediction as capturing input-output dependencies. In addition to modeling output correlations, we may wish to incorporate hard constraints between variables. For example, we may seek a model that maps descriptions of pairs of structured objects (shapes, strings, trees, etc.) into alignments of those objects. Real-life examples of such problems include bipartite matchings in alignment of 2D shapes (Belongie et al., 2002) and word alignment of sentences from a source language to a target language in machine translation (Matusov et al., 2004) or non-bipartite matchings of residues in disulfide connectivity prediction for proteins (Baldi et al., 2005). In these examples, the output variables encode presence of edges in the matching and may obey hard one-to-one matching constraints. The prediction problem in such situ-

1. Preliminary versions of some of this work appeared in the proceedings of Advances in Neural Information Processing Systems 19, 2006 and Empirical Methods in Natural Language Processing, 2005.

ations is often solved via efficient combinatorial optimization such as finding the maximum weight matching, where the model provides the appropriate edge weights.

Thus in this paper we define the term *structured output model* very broadly, as a compact scoring scheme over a (possibly very large) set of combinatorial structures and a method for finding the highest scoring structure. For example, when a probabilistic graphical model is used to capture dependencies in a structured output model, the scoring scheme is specified via a factorized probability distribution for the output variables conditional on the input variables, and the search involves some form of generalized Viterbi algorithm. More broadly, in models based on combinatorial problems, the scoring scheme is usually a simple sum of weights associated with vertices, edges, or other components of a structure; these weights are often represented as parametric functions of the inputs. Given training data consisting of instances labeled by desired structured outputs and a set of features that parameterize the scoring function, the (discriminative) learning problem is to find parameters such that the highest scoring outputs are as close as possible to the desired outputs.

In the case of structured prediction based on graphical models, which encompasses most work to date on structured prediction, two major approaches to discriminative learning have been explored: (1) maximum conditional likelihood (Lafferty et al., 2001, 2004) and (2) maximum margin (Collins, 2002; Altun et al., 2003; Taskar et al., 2004b). Both approaches are viable computationally for restricted classes of graphical models. In the broader context of the current paper, however, only the maximum-margin approach appears to be viable. In particular, it has been shown that maximum-margin estimation can be formulated as a tractable convex problem — a polynomial-size quadratic program (QP) — in several cases of interest (Taskar et al., 2004a, 2005a); such results are not available for conditional likelihood. Moreover, it is possible to find interesting subfamilies of graphical models for which maximum-margin methods are provably tractable whereas likelihood-based methods are not. For example, for the Markov random fields that arise in object segmentation problems in vision (Kumar and Hebert, 2004; Anguelov et al., 2005) the task of finding the most likely assignment reduces to a min-cut problem. In these prediction tasks, the problem of finding the highest scoring structure is tractable, while computing the partition function is #P-complete. Essentially, maximum-likelihood estimation requires the partition function, while maximum-margin estimation does not, and thus remains tractable. Polynomial-time sampling algorithms for approximating the partition function for some models do exist (Jerrum and Sinclair, 1993), but have high-degree polynomial complexity and have not yet been shown to be effective for conditional likelihood estimation.

While the reduction to a tractable convex program such as a QP is a significant step forward, it is unfortunately not the case that off-the-shelf QP solvers necessarily provide practical solutions to structured prediction problems. Indeed, despite the reduction to a polynomial number of variables, off-the-shelf QP solvers tend to scale poorly with problem and training sample size for these models. The number of variables is still large and the memory needed to maintain second-order information (for example, the inverse Hessian) is a serious practical bottleneck.

To solve the largest-scale machine learning problems, researchers have often found it expedient to consider simple gradient-based algorithms, in which each individual step is cheap in terms of computation and memory (Platt, 1999; LeCun et al., 1998). Examples of this approach in the structured prediction setting include the Structured Sequential Minimal Optimization algorithm (Taskar et al., 2004b; Taskar, 2004) and the Structured Exponentiated Gradient algorithm (Bartlett et al., 2005). These algorithms are first-order methods for solving QPs arising from low-treewidth Markov random fields and other decomposable models. In these restricted settings these methods can be used to solve significantly larger problems than can be solved with off-the-shelf QP solvers. These

methods are, however, limited in scope in that they rely on dynamic programming to compute essential quantities such as gradients. They do not extend to models where dynamic programming is not applicable, for example, to problems such as matchings and min-cuts. Another line of work in learning structured prediction models aims to approximate the arising QPs via constraint generation (Altun et al., 2003; Tsochantaridis et al., 2004). This approach only requires finding the highest scoring structure in the inner loop and incrementally solving a growing QP as constraints are added.

In this paper, we present a solution methodology for structured prediction that encompasses a broad range of combinatorial optimization problems, including matchings, min-cuts and other network flow problems. There are two key aspects to our methodology. The first is that we take a novel approach to the formulation of structured prediction problems, formulating them as saddle-point problems. This allows us to exploit recent developments in the optimization literature, where simple gradient-based methods have been developed for solving saddle-point problems (Nesterov, 2003). Moreover, we show that the key computational step in these methods—a certain projection operation—inherits the favorable computational complexity of the underlying optimization problem. This important result makes our approach viable computationally. In particular, for decomposable graphical models, the projection step is solvable via dynamic programming. For matchings and min-cuts, projection involves a min-cost quadratic flow computation, a problem for which efficient, highly-specialized algorithms are available.

The paper is organized as follows. In Section 2 we present an overview of structured prediction, focusing on three classes of tractable optimization problems. Section 3 shows how to formulate the maximum-margin estimation problem for these models as a saddle-point problem. In Section 4 we discuss the dual extragradient method for solving saddle-point problems and show how it specializes to our setting. We derive a memory-efficient version of the algorithm that requires storage proportional to the number of parameters in the model and is independent of the number of examples in Section 5. In Section 6 we illustrate the effectiveness of our approach on two very different large-scale structured prediction tasks: 3D image segmentation and word alignment in natural language translation. Finally, Section 7 presents our conclusions.

2. Structured Output Models

We begin by discussing three special cases of the general framework that we present subsequently: (1) tree-structured Markov networks, (2) Markov networks with submodular potentials, and (3) a bipartite matching model. Despite significant differences in the formal specification of these models, they share the property that in all cases the problem of finding the highest-scoring output can be formulated as a linear program (LP).

2.1 Tree-Structured Markov Networks

For simplicity of notation, we focus on tree networks, noting in passing that the extension to hypertrees is straightforward. Given N variables, $\mathbf{y} = \{y_1, \ldots, y_N\}$, with discrete domains $y_j \in \mathcal{D}_j = \{\alpha_1, \ldots, \alpha_{|\mathcal{D}_j|}\}$, we define a joint distribution over $\mathcal{Y} = \mathcal{D}_1 \times \ldots \times \mathcal{D}_N$ via

$$P(\mathbf{y}) \propto \prod_{j \in \mathcal{V}} \phi_j(y_j) \prod_{jk \in \mathcal{E}} \phi_{jk}(y_j, y_k),$$

where $(\mathcal{V} = \{1, \ldots, N\}, \mathcal{E} \subset \{jk : j < k, j \in \mathcal{V}, k \in \mathcal{V}\})$ is an undirected graph, and where $\{\phi_j(y_j), j \in \mathcal{V}\}$ are the node potentials and $\{\phi_{jk}(y_j, y_k), jk \in \mathcal{E}\}$ are the edge potentials. We can find the most

likely assignment, $\arg\max_{\mathbf{y}} P(\mathbf{y})$, using the Viterbi dynamic programming algorithm for trees. We can also find it using a standard linear programming formulation as follows. We introduce variables $z_{j\alpha}$ to denote indicators $\mathbb{1}(y_j = \alpha)$ for all variables $j \in \mathcal{V}$ and their values $\alpha \in \mathcal{D}_j$. Similarly, we introduce variables $z_{jk\alpha\beta}$ to denote indicators $\mathbb{1}(y_j = \alpha, y_k = \beta)$ for all edges $jk \in \mathcal{E}$ and the values of their nodes, $\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k$. We can formulate the problem of finding the maximal probability configuration as follows:

$$\max_{0 \le \mathbf{z} \le 1} \quad \sum_{j \in \mathcal{V}} \sum_{\alpha \in \mathcal{D}_j} z_{j\alpha} \log \phi_j(\alpha) + \sum_{jk \in \mathcal{E}} \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk\alpha\beta} \log \phi_{jk}(\alpha, \beta) \tag{1}$$

$$\text{s.t.} \quad \sum_{\alpha \in \mathcal{D}_j} z_{j\alpha} = 1, \ \forall j \in \mathcal{V}; \qquad \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk\alpha\beta} = 1, \quad \forall jk \in \mathcal{E}; \tag{2}$$

$$\sum_{\alpha \in \mathcal{D}_j} z_{jk\alpha\beta} = z_{k\beta}, \ \forall jk \in \mathcal{E}, \beta \in \mathcal{D}_k; \qquad \sum_{\beta \in \mathcal{D}_k} z_{jk\alpha\beta} = z_{j\alpha}, \ \forall jk \in \mathcal{E}, \alpha \in \mathcal{D}_j, \tag{3}$$

where (2) expresses normalization constraints and (3) captures marginalization constraints. This LP has integral optimal solutions if $\mathcal{E}$ is a forest (Chekuri et al., 2001; Wainwright et al., 2002; Chekuri et al., 2005). In networks of general topology, however, the optimal solution can be fractional (as expected, since the problem is NP-hard). Other important exceptions can be found, however, specifically by focusing on constraints on the potentials rather than constraints on the topology. We discuss one such example in the following section.

2.2 Markov Networks with Submodular Potentials

We consider a special class of Markov networks, common in vision applications, in which inference reduces to a tractable min-cut problem (Greig et al., 1989; Kolmogorov and Zabih, 2004). We assume that (1) all variables are binary ($\mathcal{D}_j = \{0,1\}$), and (2) all edge potentials are "regular" (i.e., submodular):

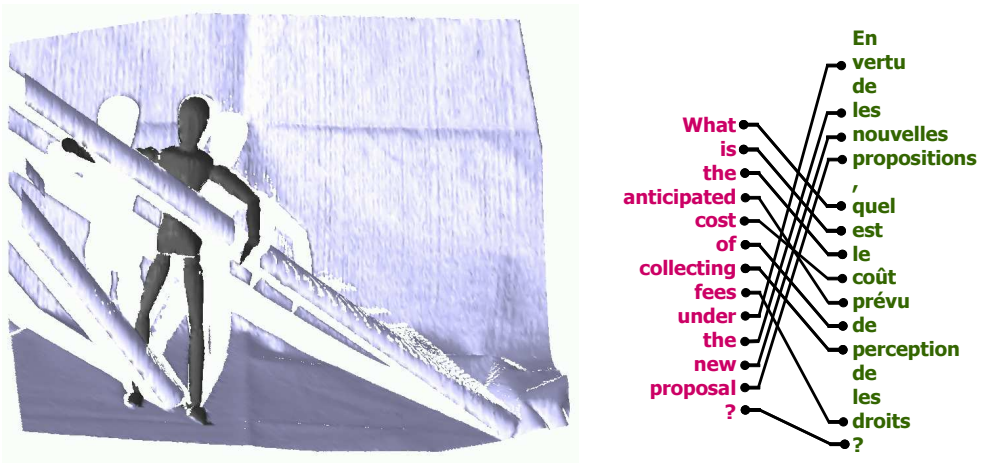
$$\log \phi_{jk}(0,0) + \log \phi_{jk}(1,1) \ge \log \phi_{jk}(1,0) + \log \phi_{jk}(0,1), \qquad \forall jk \in \mathcal{E}. \tag{4}$$

Such potentials prefer assignments where connected nodes have the same label, that is, $y_j = y_k$. This notion of regularity can be extended to potentials over more than two variables (Kolmogorov and Zabih, 2004). These assumptions ensure that the LP in Eq. (1) has integral optimal solutions (Chekuri et al., 2001; Kolmogorov and Wainwright, 2005; Chekuri et al., 2005). Similar kinds of networks (defined also for non-binary variables and non-pairwise potentials) were called "associative Markov networks" by Taskar et al. (2004a) and Anguelov et al. (2005), who used them for object segmentation and hypertext classification.

In figure-ground segmentation (see Fig. 1a), the node potentials capture local evidence about the label of a pixel or range scan point. Edges usually connect nearby pixels in an image, and serve to correlate their labels. Assuming that such correlations tend to be *positive* (connected nodes tend to have the same label) leads us to consider simplified edge potentials of the form $\phi_{jk}(y_j, y_k) = \exp\{-s_{jk}\mathbb{1}(y_j \ne y_k)\}$, where s_{jk} is a nonnegative penalty for assigning y_j and y_k different labels. Note that such potentials are regular if $s_{jk} \ge 0$. Expressing node potentials as $\phi_j(y_j) = \exp\{s_j y_j\}$, we have $P(\mathbf{y}) \propto \exp\left\{\sum_{j \in \mathcal{V}} s_j y_j - \sum_{jk \in \mathcal{E}} s_{jk}\mathbb{1}(y_j \ne y_k)\right\}$. Under this restriction on the potentials, we can obtain the following (simpler) LP:

$$\max_{0 \le \mathbf{z} \le 1} \quad \sum_{j \in \mathcal{V}} s_j z_j - \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} \tag{5}$$

$$\text{s.t.} \quad z_j - z_k \le z_{jk}, \ z_k - z_j \le z_{jk}, \ \forall jk \in \mathcal{E},$$



(a) (b)

Figure 1: Structured prediction applications: (a) 3D figure-ground segmentation; (b) Word alignment in machine translation.

where the continuous variables z_j correspond to a relaxation of the binary variables y_j, and the constraints encode $z_{jk} = \mathbb{1}(z_j \neq z_k)$. To see this, note that the constraints can be equivalently expressed as $|z_j - z_k| \leq z_{jk}$. Because s_{jk} is positive, $z_{jk} = |z_k - z_j|$ at the maximum, which is equivalent to $\mathbb{1}(z_j \neq z_k)$ if the z_j, z_k variables are binary. An integral optimal solution always exists, since the constraint matrix is totally unimodular (Schrijver, 2003).

We can parameterize the node and edge potentials in terms of user-provided features $\mathbf{x}_j$ and $\mathbf{x}_{jk}$ associated with the nodes and edges. In particular, in 3D range data, $\mathbf{x}_j$ might involve spin-image features or spatial occupancy histograms of a point j, while $\mathbf{x}_{jk}$ might include the distance between points j and k, the dot-product of their normals, etc. The simplest model of dependence is a linear combination of features: $s_j = \mathbf{w}_n^\top \mathbf{f}_n(\mathbf{x}_j)$ and $s_{jk} = \mathbf{w}_e^\top \mathbf{f}_e(\mathbf{x}_{jk})$, where $\mathbf{w}_n$ and $\mathbf{w}_e$ are node and edge parameters, and $\mathbf{f}_n$ and $\mathbf{f}_e$ are node and edge feature mappings, of dimension d_n and d_e, respectively. To ensure non-negativity of s_{jk}, we assume that the edge features $\mathbf{f}_e$ are nonnegative and we impose the restriction $\mathbf{w}_e \geq 0$. This constraint is incorporated into the learning formulation we present below. We assume that the feature mappings $\mathbf{f}$ are provided by the user and our goal is to estimate parameters $\mathbf{w}$ from labeled data. We abbreviate the score assigned to a labeling $\mathbf{y}$ for an input $\mathbf{x}$ as $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_j y_j \mathbf{w}_n^\top \mathbf{f}_n(\mathbf{x}_j) - \sum_{jk \in \mathcal{E}} y_{jk} \mathbf{w}_e^\top \mathbf{f}_e(\mathbf{x}_{jk})$, where $y_{jk} = \mathbb{1}(y_j \neq y_k)$.

2.3 Matchings

Consider modeling the task of word alignment of parallel bilingual sentences (Fig. 1b) as a maximum weight bipartite matching problem in a graph, where the nodes $\mathcal{V} = \mathcal{V}^s \cup \mathcal{V}^t$ correspond to the words in the "source" sentence ($\mathcal{V}^s$) and the "target" sentence ($\mathcal{V}^t$) and the edges $\mathcal{E} = \{jk : j \in \mathcal{V}^s, k \in \mathcal{V}^t\}$ correspond to possible alignments between the words. For simplicity, assume that each word aligns to one or zero words in the other sentence. The edge weight s_{jk} represents the degree to which word j in one sentence can translate into the word k in the other sentence. Our objective is to find an alignment that maximizes the sum of edge scores. We represent a matching using a set of

binary variables y_{jk} that are set to 1 if word j is assigned to word k in the other sentence, and 0 otherwise. The score of an assignment is the sum of edge scores: $s(\mathbf{y}) = \sum_{jk \in \mathcal{E}} s_{jk} y_{jk}$. The maximum weight bipartite matching problem, $\arg\max_{\mathbf{y} \in \mathcal{Y}} s(\mathbf{y})$, can be found by solving the following LP:

$$\begin{array}{ll} \max\limits_{0 \leq \mathbf{z} \leq 1} & \sum\limits_{jk \in \mathcal{E}} s_{jk} z_{jk} \\ \text{s.t.} & \sum\limits_{j \in \mathcal{V}^s} z_{jk} \leq 1, \ \forall k \in \mathcal{V}^t; \qquad \sum\limits_{k \in \mathcal{V}^t} z_{jk} \leq 1, \ \forall j \in \mathcal{V}^s. \end{array} \tag{6}$$

where again the continuous variables z_{jk} correspond to the relaxation of the binary variables y_{jk}. As in the min-cut problem, this LP is guaranteed to have integral solutions for any scoring function $s(\mathbf{y})$ (Schrijver, 2003).

For word alignment, the scores s_{jk} can be defined in terms of the word pair jk and input features associated with $\mathbf{x}_{jk}$. We can include the identity of the two words, the relative position in the respective sentences, the part-of-speech tags, the string similarity (for detecting cognates), etc. We let $s_{jk} = \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$ for a user-provided feature mapping $\mathbf{f}$ and abbreviate $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{jk} y_{jk} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$.

2.4 General Structure

More generally, we consider prediction problems in which the input $\mathbf{x} \in \mathcal{X}$ is an arbitrary structured object and the output is a vector of values $\mathbf{y} = (y_1, \ldots, y_{L_\mathbf{x}})$ encoding, for example, a matching or a cut in the graph. We assume that the length $L_\mathbf{x}$ and the structure encoded by $\mathbf{y}$ depend deterministically on the input $\mathbf{x}$. In our word alignment example, the output space is defined by the length of the two sentences. Denote the output space for a given input $\mathbf{x}$ as $\mathcal{Y}(\mathbf{x})$ and the entire output space as $\mathcal{Y} = \bigcup_{\mathbf{x} \in \mathcal{X}} \mathcal{Y}(\mathbf{x})$.

Consider the class of structured prediction models $\mathcal{H}$ defined by the linear family:

$$h_\mathbf{w}(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}), \tag{7}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is a vector of functions $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^n$. This formulation is very general. Indeed, it is too general for our purposes—for many $(\mathbf{f}, \mathcal{Y})$ pairs, finding the optimal $\mathbf{y}$ is intractable. We specialize to the class of models in which the optimization problem in Eq. (7) can be solved in polynomial time via convex optimization; this is still a very large class of models. Beyond the examples discussed here, it includes weighted context-free grammars and dependency grammars (Manning and Schütze, 1999) and string edit distance models for sequence alignment (Durbin et al., 1998).

3. Large Margin Estimation

We assume a set of training instances $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, where each instance consists of a structured object $\mathbf{x}_i$ (such as a graph) and a target solution $\mathbf{y}_i$ (such as a matching). Consider learning the parameters $\mathbf{w}$ in the conditional likelihood setting. We can define $P_\mathbf{w}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_\mathbf{w}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}$, where $Z_\mathbf{w}(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}')\}$, and maximize the conditional log-likelihood $\sum_i \log P_\mathbf{w}(\mathbf{y}_i \mid \mathbf{x}_i)$, perhaps with additional regularization of the parameters $\mathbf{w}$. As we have noted earlier, however, the problem of computing the partition function $Z_\mathbf{w}(\mathbf{x})$ is computationally intractable for many of the problems we are interested in. In particular, it is #P-complete for matchings and min-cuts (Valiant, 1979; Jerrum and Sinclair, 1993).

We thus retreat from conditional likelihood and consider the max-margin formulation developed in several recent papers (Collins, 2002; Altun et al., 2003; Taskar et al., 2004b). In this formulation, we seek to find parameters $\mathbf{w}$ such that:

$$\mathbf{y}_i = \arg\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}'_i), \qquad \forall i,$$

where $\mathcal{Y}_i = \mathcal{Y}(\mathbf{x}_i)$. The solution space $\mathcal{Y}_i$ depends on the structured object $\mathbf{x}_i$; for example, the space of possible matchings depends on the precise set of nodes and edges in the graph.

As in univariate prediction, we measure the error of prediction using a loss function $\ell(\mathbf{y}_i, \mathbf{y}'_i)$. To obtain a convex formulation, we upper bound the loss $\ell(\mathbf{y}_i, h_\mathbf{w}(\mathbf{x}_i))$ using the hinge function: $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)]$, where $\ell_i(\mathbf{y}'_i) = \ell(\mathbf{y}_i, \mathbf{y}'_i)$, and $\mathbf{f}_i(\mathbf{y}'_i) = \mathbf{f}(\mathbf{x}_i, \mathbf{y}'_i)$. Minimizing this upper bound will force the true structure $\mathbf{y}_i$ to be optimal with respect to $\mathbf{w}$ for each instance i:

$$\min_{\mathbf{w} \in \mathcal{W}} \quad \sum_i \max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i), \tag{8}$$

where $\mathcal{W}$ is the set of allowed parameters $\mathbf{w}$. We assume that the parameter space $\mathcal{W}$ is a convex set, typically a norm ball $\{\mathbf{w} : ||\mathbf{w}||_p \leq \gamma\}$ with $p = 1, 2$ and a regularization parameter γ. In the case that $\mathcal{W} = \{\mathbf{w} : \ ||\mathbf{w}||_2 \leq \gamma\}$, this formulation is equivalent to the standard large margin formulation using slack variables ξ and slack penalty C (cf. Taskar et al., 2004b), for some suitable values of C depending on γ. The correspondence can be seen as follows: let $\mathbf{w}^*(C)$ be a solution to the optimization problem with slack penalty C and define $\gamma(C) = ||\mathbf{w}^*(C)||$. Then $\mathbf{w}^*$ is also a solution to Eq. (8). Conversely, we can invert the mapping $\gamma(\cdot)$ to find those values of C (possibly non-unique) that give rise to the same solution as Eq. (8) for a specific γ. In the case of submodular potentials, there are additional linear constraints on the edge potentials. In the setting of Eq. (5), we simply constrain $w_e \geq 0$. For general submodular potentials, we can parameterize the log of the edge potential using four sets of edge parameters, $\mathbf{w}_{e00}, \mathbf{w}_{e01}, \mathbf{w}_{e10}, \mathbf{w}_{e11}$, as follows: $\log \phi_{jk}(\alpha, \beta) = \mathbf{w}_{e\alpha\beta}^\top \mathbf{f}(\mathbf{x}_{jk})$. Assuming, as before, that the edge features are nonnegative, the regularity of the potentials can be enforced via a linear constraint: $\mathbf{w}_{e00} + \mathbf{w}_{e11} \geq \mathbf{w}_{e10} + \mathbf{w}_{e01}$, where the inequality should be interpreted componentwise.

The key to solving Eq. (8) efficiently is the *loss-augmented inference problem*,

$$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)]. \tag{9}$$

This optimization problem has precisely the same form as the prediction problem whose parameters we are trying to learn—$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$—but with an additional term corresponding to the loss function. Tractability of the loss-augmented inference thus depends not only on the tractability of $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$, but also on the form of the loss term $\ell_i(\mathbf{y}'_i)$. A natural choice in this regard is the Hamming distance, which simply counts the number of variables in which a candidate solution $\mathbf{y}'_i$ differs from the target output $\mathbf{y}_i$. In general, we need only assume that the loss function decomposes over the variables in $\mathbf{y}_i$.

In particular, for word alignment, we use weighted Hamming distance, which counts the number of variables in which a candidate matching $\mathbf{y}'_i$ differs from the target alignment $\mathbf{y}_i$, with different cost for false positives (c^+) and false negatives (c^-):

$$\begin{aligned} \ell(\mathbf{y}_i, \mathbf{y}'_i) &= \sum_{jk \in \mathcal{E}_i} \left[c^- y_{i,jk}(1 - y'_{i,jk}) + c^+ y'_{i,jk}(1 - y_{i,jk}) \right] \\ &= \sum_{jk \in \mathcal{E}_i} c^- y_{i,jk} + \sum_{jk \in \mathcal{E}_i} [c^+ - (c^- + c^+) y_{i,jk}] y'_{i,jk}, \end{aligned} \tag{10}$$

where $y_{i,jk}$ indicates the presence of edge jk in example i and $\mathcal{E}_i$ is the set of edges in example i. The loss-augmented matching problem can then be written as an LP similar to Eq. (6) (without the constant term $\sum_{jk} c^- y_{i,jk}$):

$$\begin{aligned} \max_{0 \le \mathbf{z}_i \le 1} \quad & \sum_{jk \in \mathcal{E}_i} z_{i,jk} [\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{i,jk}) + c^+ - (c^- + c^+) y_{i,jk}] \\ \text{s.t.} \quad & \sum_{j \in \mathcal{V}_i^s} z_{i,jk} \le 1, \ \forall k \in \mathcal{V}_i^t; \qquad \sum_{k \in \mathcal{V}_i^t} z_{i,jk} \le 1, \ \forall j \in \mathcal{V}_i^s, \end{aligned}$$

where $\mathbf{f}(\mathbf{x}_{i,jk})$ is the vector of features of the edge jk in example i and $\mathcal{V}_i^s$ and $\mathcal{V}_i^t$ are the nodes in example i. As before, the continuous variables $z_{i,jk}$ correspond to the binary values $y'_{i,jk}$.

Generally, suppose we can express the prediction problem as an LP:

$$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) = \max_{\mathbf{z}_i \in \mathcal{Z}_i} \mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i,$$

where

$$\mathcal{Z}_i = \{\mathbf{z}_i : \mathbf{A}_i \mathbf{z}_i \le \mathbf{b}_i, \ 0 \le \mathbf{z}_i \le 1\}, \tag{11}$$

for appropriately defined $\mathbf{F}_i, \mathbf{A}_i$ and $\mathbf{b}_i$. Then we have a similar LP for the loss-augmented inference for each example i:

$$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i) = d_i + \max_{\mathbf{z}_i \in \mathcal{Z}_i} (\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i)^\top \mathbf{z}_i, \tag{12}$$

for appropriately defined d_i and $\mathbf{c}_i$. For the matching case, $d_i = \sum_{jk} c^- y_{i,jk}$ is the constant term, $\mathbf{F}_i$ is a matrix that has a column of features $\mathbf{f}(\mathbf{x}_{i,jk})$ for each edge jk in example i, and $\mathbf{c}_i$ is the vector of the loss terms $c^+ - (c^- + c^+) y_{i,jk}$. Let $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ and $\mathcal{Z} = \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_m$. With these definitions, we have the following saddle-point problem:

$$\min_{\mathbf{w} \in \mathcal{W}} \max_{\mathbf{z} \in \mathcal{Z}} \quad \sum_i \left(\mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i) \right). \tag{13}$$

where we have omitted the constant term $\sum_i d_i$. The only difference between this formulation and our initial formulation in Eq. (8) is that we have created a concise continuous optimization problem by replacing the discrete $\mathbf{y}'_i$'s with continuous $\mathbf{z}_i$'s.

When the prediction problem is intractable (for example, in general Markov networks or tripartite matchings), we can use a convex relaxation (for example, a linear or semidefinite program) to upper bound $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$ and obtain an approximate maximum-margin formulation. This is the approach taken in Taskar et al. (2004b) for general Markov networks using the LP in Eq. (1).

To solve (13), we could proceed by making use of Lagrangian duality. This approach, explored in Taskar et al. (2004a, 2005a), yields a joint convex optimization problem. If the parameter space $\mathcal{W}$ is described by linear and convex quadratic constraints, the result is a convex quadratic program which can be solved using a generic QP solver.

We briefly outline this approach below, but in this paper, we take a different tack, solving the problem in its natural saddle-point form. As we discuss in the following section, this approach allows us to exploit the structure of $\mathcal{W}$ and $\mathcal{Z}$ *separately*, allowing for efficient solutions for a wider range of parameterizations and structures. It also opens up alternatives with respect to numerical algorithms.

Before moving on to solution of the saddle-point problem, we consider the joint convex form when the feasible set has the form of (11) and the loss-augmented inference problem is a LP, as in (12). Using commercial convex optimization solvers for this formulation will provide us with a comparison point for our saddle-point solver. We now proceed to present this alternative form.

To transform the saddle-point form of (13) into a standard convex optimization form, we take the dual of the individual loss-augmented LPs (12):

$$\max_{\mathbf{z}_i \in \mathcal{Z}_i} (\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i)^\top \mathbf{z}_i = \min_{(\lambda_i, \mu_i) \in \Lambda_i(\mathbf{w})} \mathbf{b}_i^\top \lambda_i + \mathbf{1}^\top \mu_i \tag{14}$$

where $\Lambda_i(\mathbf{w}) = \{(\lambda_i, \mu_i) \geq 0 : \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i \leq \mathbf{A}_i^\top \lambda_i + \mu_i\}$ defines the feasible space for the dual variables λ_i and μ_i. Substituting back in equation (13) and writing $\lambda = (\lambda_1, \ldots, \lambda_m)$, $\mu = (\mu_1, \ldots, \mu_m)$, we obtain (omitting the constant $\sum_i d_i$):

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{W}, (\lambda, \mu) \geq 0} \quad & \sum_i \left(\mathbf{b}_i^\top \lambda_i + \mathbf{1}^\top \mu_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i) \right) \\ \text{s.t.} \quad & \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i \leq \mathbf{A}_i^\top \lambda_i + \mu_i \quad i = 1, \ldots, m. \end{aligned} \tag{15}$$

If $\mathcal{W}$ is defined by linear and convex quadratic constraints, the above optimization problem can be solved using standard commercial solvers. The number of variables and constraints in this problem is linear in the number of the parameters *and* the training data (for example nodes and edges).

4. Saddle-Point Problems and the Dual Extragradient Method

We begin by establishing some notation and definitions. Denote the objective of the saddle-point problem in (13) by:

$$\mathcal{L}(\mathbf{w}, \mathbf{z}) \equiv \sum_i \mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i).$$

$\mathcal{L}(\mathbf{w}, \mathbf{z})$ is bilinear in $\mathbf{w}$ and $\mathbf{z}$, with gradient given by: $\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{z}) = \sum_i \mathbf{F}_i \mathbf{z}_i - \mathbf{f}_i(\mathbf{y}_i)$ and $\nabla_{\mathbf{z}_i} \mathcal{L}(\mathbf{w}, \mathbf{z}) = \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i$.

We can view this problem as a zero-sum game between two players, $\mathbf{w}$ and $\mathbf{z}$. Consider a simple iterative improvement method based on gradient projections:

$$\mathbf{w}^{t+1} = \boldsymbol{\pi}_{\mathcal{W}}(\mathbf{w}^t - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^t, \mathbf{z}^t)); \quad \mathbf{z}_i^{t+1} = \boldsymbol{\pi}_{\mathcal{Z}_i}(\mathbf{z}_i^t + \eta \nabla_{\mathbf{z}_i} \mathcal{L}(\mathbf{w}^t, \mathbf{z}^t)), \tag{16}$$

where η is a step size and $\boldsymbol{\pi}_{\mathcal{V}}(\mathbf{v}) = \arg\min_{\mathbf{v}' \in \mathcal{V}} ||\mathbf{v} - \mathbf{v}'||_2$ denotes the Euclidean projection of a vector $\mathbf{v}$ onto a convex set $\mathcal{V}$. In this simple iteration, each player makes a small best-response improvement without taking into account the effect of the change on the opponent's strategy. This usually leads to oscillations, and indeed, this method is generally not guaranteed to converge for bilinear objectives for any step size (Korpelevich, 1976; He and Liao, 2002). One way forward is to attempt to average the points $(\mathbf{w}^t, \mathbf{z}^t)$ to reduce oscillation. We pursue a different approach that is based on the dual extragradient method of Nesterov (2003). In our previous work (Taskar et al., 2006), we used a related method, the extragradient method due to Korpelevich (1976). The dual extragradient is, however, a more flexible and general method, in terms of the types of projections and feasible sets that can be used, allowing a broader range of structured problems and parameter regularization schemes. Before we present the algorithm, we introduce some notation which will be useful for its description.

Let us combine $\mathbf{w}$ and $\mathbf{z}$ into a single vector, $\mathbf{u} = (\mathbf{w}, \mathbf{z})$, and define the joint feasible space $\mathcal{U} = \mathcal{W} \times \mathcal{Z}$. Note that $\mathcal{U}$ is convex since it is a direct product of convex sets.

We denote the (affine) gradient operator on this joint space as

$$\begin{pmatrix} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{z}) \\ -\nabla_{\mathbf{z}_1} \mathcal{L}(\mathbf{w}, \mathbf{z}) \\ \vdots \\ -\nabla_{\mathbf{z}_m} \mathcal{L}(\mathbf{w}, \mathbf{z}) \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & \mathbf{F}_1 \cdots \mathbf{F}_m \\ -\mathbf{F}_1^\top & \\ \vdots & 0 \\ -\mathbf{F}_m^\top & \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} \mathbf{w} \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_m \end{pmatrix}}_{\mathbf{u}} - \underbrace{\begin{pmatrix} \sum_i \mathbf{f}_i(\mathbf{y}_i) \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_m \end{pmatrix}}_{\mathbf{a}} = \mathbf{F}\mathbf{u} - \mathbf{a}.$$

4.1 Dual Extragradient

We first present the dual extragradient algorithm of Nesterov (2003) using the Euclidean geometry induced by the standard 2-norm, and consider a non-Euclidean setup in Sec. 4.2.

As shown in Fig. 2, the dual extragradient algorithm proceeds using very simple gradient and projection calculations.

Initialize: Choose $\hat{\mathbf{u}} \in \mathcal{U}$, set $\mathbf{s}^{-1} = 0$.
Iteration t, $0 \le t \le \tau$:

$$\begin{aligned} \mathbf{v} &= \boldsymbol{\pi}_{\mathcal{U}}(\hat{\mathbf{u}} + \eta \mathbf{s}^{t-1}); \\ \mathbf{u}^t &= \boldsymbol{\pi}_{\mathcal{U}}(\mathbf{v} - \eta(\mathbf{F}\mathbf{v} - \mathbf{a})); \\ \mathbf{s}^t &= \mathbf{s}^{t-1} - (\mathbf{F}\mathbf{u}^t - \mathbf{a}). \end{aligned} \tag{17}$$

Output: $\overline{\mathbf{u}}^\tau = \frac{1}{\tau+1} \sum_{t=0}^{\tau} \mathbf{u}^t$.

Figure 2: Euclidean dual extragradient.

To relate this generic algorithm to our setting, recall that $\mathbf{u}$ is composed of subvectors $\mathbf{w}$ and $\mathbf{z}$; this induces a commensurate decomposition of the $\mathbf{v}$ and $\mathbf{s}$ vectors into subvectors. To refer to these subvectors we will abuse notation and use the symbols $\mathbf{w}$ and $\mathbf{z}_i$ as indices. Thus, we write $\mathbf{v} = (\mathbf{v}_{\mathbf{w}}, \mathbf{v}_{\mathbf{z}_1}, \ldots, \mathbf{v}_{\mathbf{z}_m})$, and similarly for $\mathbf{u}$ and $\mathbf{s}$. Using this notation, the generic algorithm in Eq. (17) expands into the following dual extragradient algorithm for structured prediction (where the brackets represent gradient vectors):

$$\mathbf{v}_{\mathbf{w}} = \boldsymbol{\pi}_{\mathcal{W}}(\hat{\mathbf{u}}_{\mathbf{w}} + \eta \mathbf{s}_{\mathbf{w}}^{t-1}); \qquad \mathbf{v}_{\mathbf{z}_i} = \boldsymbol{\pi}_{\mathcal{Z}_i}(\hat{\mathbf{u}}_{\mathbf{z}_i} + \eta \mathbf{s}_{\mathbf{z}_i}^{t-1}), \ \forall i;$$

$$\mathbf{u}_{\mathbf{w}}^t = \boldsymbol{\pi}_{\mathcal{W}}(\mathbf{v}_{\mathbf{w}} - \eta \left[\sum_i \mathbf{F}_i \mathbf{v}_{\mathbf{z}_i} - \mathbf{f}_i(\mathbf{y}_i) \right]); \qquad \mathbf{u}_{\mathbf{z}_i}^t = \boldsymbol{\pi}_{\mathcal{Z}_i}(\mathbf{v}_{\mathbf{z}_i} + \eta \left[\mathbf{F}_i^\top \mathbf{v}_{\mathbf{w}} + \mathbf{c}_i \right]), \ \forall i;$$

$$\mathbf{s}_{\mathbf{w}}^t = \mathbf{s}_{\mathbf{w}}^{t-1} - \left[\sum_i \mathbf{F}_i \mathbf{u}_{\mathbf{z}_i}^t - \mathbf{f}_i(\mathbf{y}_i) \right]; \qquad \mathbf{s}_{\mathbf{z}_i}^t = \mathbf{s}_{\mathbf{z}_i}^{t-1} + \left[\mathbf{F}_i^\top \mathbf{u}_{\mathbf{w}}^t + \mathbf{c}_i \right], \ \forall i.$$

In the convergence analysis of dual extragradient (Nesterov, 2003), the stepsize η is set to the inverse of the Lipschitz constant (with respect to the 2-norm) of the gradient operator:

$$1/\eta = L \equiv \max_{\mathbf{u}, \mathbf{u}' \in \mathcal{U}} \frac{||\mathbf{F}(\mathbf{u} - \mathbf{u}')||_2}{||\mathbf{u} - \mathbf{u}'||_2} \le ||\mathbf{F}||_2,$$

where $||\mathbf{F}||_2$ is the largest singular value of the matrix $\mathbf{F}$. In practice, various simple heuristics can be considered for setting the stepsize, including search procedures based on optimizing the gap merit function (see, e.g., He and Liao, 2002).

4.1.1 CONVERGENCE

One measure of quality of a saddle-point solution is via the gap function:

$$\mathcal{G}(\mathbf{w},\mathbf{z}) = \left[\max_{\mathbf{z}'\in\mathcal{Z}} \mathcal{L}(\mathbf{w},\mathbf{z}') - \mathcal{L}^*\right] + \left[\mathcal{L}^* - \min_{\mathbf{w}'\in\mathcal{W}} \mathcal{L}(\mathbf{w}',\mathbf{z})\right], \tag{18}$$

where the optimal loss is denoted $\mathcal{L}^* = \min_{\mathbf{w}'\in\mathcal{W}} \max_{\mathbf{z}\in\mathcal{Z}} \mathcal{L}(\mathbf{w},\mathbf{z})$. For non-optimal points $(\mathbf{w},\mathbf{z})$, the gap $\mathcal{G}(\mathbf{w},\mathbf{z})$ is positive and serves as a useful merit function, a measure of accuracy of a solution found by the extragradient algorithm. At an optimum we have

$$\mathcal{G}(\mathbf{w}^*,\mathbf{z}^*) = \max_{\mathbf{z}'\in\mathcal{Z}} \mathcal{L}(\mathbf{w}^*,\mathbf{z}') - \min_{\mathbf{w}'\in\mathcal{W}} \mathcal{L}(\mathbf{w}',\mathbf{z}^*) = 0.$$

Define the Euclidean divergence function as

$$d(\mathbf{v},\mathbf{v}') = \frac{1}{2}||\mathbf{v}-\mathbf{v}'||_2^2,$$

and define a restricted gap function parameterized by positive divergence radii $D_\mathbf{w}$ and $D_\mathbf{z}$

$$\mathcal{G}_{D_\mathbf{w},D_\mathbf{z}}(\mathbf{w},\mathbf{z}) = \max_{\mathbf{z}'\in\mathcal{Z}} \left[\mathcal{L}(\mathbf{w},\mathbf{z}') : d(\hat{\mathbf{z}},\mathbf{z}') \le D_\mathbf{z}\right] - \min_{\mathbf{w}'\in\mathcal{W}} \left[\mathcal{L}(\mathbf{w}',\mathbf{z}) : d(\hat{\mathbf{w}},\mathbf{w}') \le D_\mathbf{w}\right],$$

where the point $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_\mathbf{w},\hat{\mathbf{u}}_\mathbf{z}) \in \mathcal{U}$ is an arbitrary point that can be thought of as the "center" of $\mathcal{U}$. Assuming there exists a solution $\mathbf{w}^*,\mathbf{z}^*$ such that $d(\hat{\mathbf{w}},\mathbf{w}^*) \le D_\mathbf{w}$ and $d(\hat{\mathbf{z}},\mathbf{z}^*) \le D_\mathbf{z}$, this restricted gap function coincides with the unrestricted function defined in Eq. (18). The choice of the center point $\hat{\mathbf{u}}$ should reflect an expectation of where the "average" solution lies, as will be evident from the convergence guarantees presented below. For example, we can take $\hat{\mathbf{u}}_\mathbf{w} = 0$ and let $\hat{\mathbf{u}}_{\mathbf{z}_i}$ correspond to the encoding of the target $\mathbf{y}_i$.

By Theorem 2 of Nesterov (2003), after τ iterations, the gap of $(\overline{\mathbf{w}}^\tau,\overline{\mathbf{z}}^\tau) = \overline{\mathbf{u}}^\tau$ is upper bounded by:

$$\mathcal{G}_{D_\mathbf{w},D_\mathbf{z}}(\overline{\mathbf{w}}^\tau,\overline{\mathbf{z}}^\tau) \le \frac{(D_\mathbf{w}+D_\mathbf{z})L}{\tau+1}. \tag{19}$$

This implies that $O(\frac{1}{\varepsilon})$ steps are required to achieve a desired accuracy of solution ε as measured by the gap function. Note that the exponentiated gradient algorithm (Bartlett et al., 2005) has the same $O(\frac{1}{\varepsilon})$ convergence rate. This sublinear convergence rate is slow compared to interior point methods, which enjoy superlinear convergence (Boyd and Vandenberghe, 2004). However, the simplicity of each iteration, the locality of key operations (projections), and the linear memory requirements make this a practical algorithm when the desired accuracy ε is not too small, and, in particular, these properties align well with the desiderata of large-scale machine learning algorithms. We illustrate these properties experimentally in Section 6.

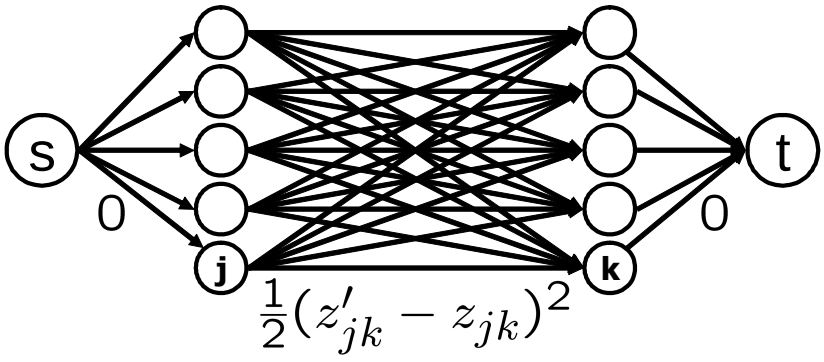


Figure 3: Euclidean projection onto the matching polytope using min-cost quadratic flow. Source s is connected to all the "source" nodes and target t connected to all the "target" nodes, using edges of capacity 1 and cost 0. The original edges jk have a quadratic cost $\frac{1}{2}(z'_{jk} - z_{jk})^2$ and capacity 1.

4.1.2 PROJECTIONS

The efficiency of the algorithm hinges on the computational complexity of the Euclidean projection onto the feasible sets $\mathcal{W}$ and $\mathcal{Z}_i$. In the case of $\mathcal{W}$, projections are cheap when we have a 2-norm ball $\{\mathbf{w} : ||\mathbf{w}||_2 \leq \gamma\}$: $\boldsymbol{\pi}_{\mathcal{W}}(\mathbf{w}) = \gamma\mathbf{w}/\max(\gamma, ||\mathbf{w}||_2)$. Additional non-negativity constraints on the parameters (e.g., $\mathbf{w}_e \geq 0$) can also be easily incorporated by clipping negative values. Projections onto the 1-norm ball are not expensive either (Boyd and Vandenberghe, 2004), but may be better handled by the non-Euclidean setup we discuss below.

We turn to the consideration of the projections onto $\mathcal{Z}_i$. The complexity of these projections is the key issue determining the viability of the extragradient approach for our class of problems. In fact, for both alignment and matchings these projections turn out to reduce to classical network flow problems for which efficient solutions exist. In case of alignment, $\mathcal{Z}_i$ is the convex hull of the bipartite matching polytope and the projections onto $\mathcal{Z}_i$ reduce to the much-studied minimum cost quadratic flow problem (Bertsekas, 1998). In particular, the projection problem $\mathbf{z} = \boldsymbol{\pi}_{\mathcal{Z}_i}(\mathbf{z}'_i)$ can be computed by solving

$$\begin{aligned} \min_{0 \leq \mathbf{z}_i \leq 1} \quad & \sum_{jk \in \mathcal{E}_i} \frac{1}{2}(z'_{i,jk} - z_{i,jk})^2 \\ \text{s.t.} \quad & \sum_{j \in \mathcal{V}_i^s} z_{i,jk} \leq 1, \ \forall j \in \mathcal{V}_i^t; \qquad \sum_{k \in \mathcal{V}_i^t} z_{i,jk} \leq 1, \ \forall k \in \mathcal{V}_i^s. \end{aligned}$$

We use a standard reduction of bipartite matching to min-cost flow (see Fig. 3) by introducing a source node s connected to all the words in the "source" sentence, $\mathcal{V}_i^s$, and a target node t connected to all the words in the "target" sentence, $\mathcal{V}_i^t$, using edges of capacity 1 and cost 0. The original edges jk have a quadratic cost $\frac{1}{2}(z'_{i,jk} - z_{i,jk})^2$ and capacity 1. Since the edge capacities are 1, the flow conservation constraints at each original node ensure that the (possibly fractional) degree of each node in a valid flow is at most 1. Now the minimum cost flow from the source s to the target t computes projection of $\mathbf{z}'_i$ onto $\mathcal{Z}_i$.

The reduction of the min-cut polytope projection to a convex network flow problem is more complicated; we present this reduction in Appendix A. Algorithms for solving convex network flow problems (see, for example, Bertsekas et al., 1997) are nearly as efficient as those for solving linear min-cost flow problems, bipartite matchings and min-cuts. In case of word alignment, the running time scales with the cube of the sentence length. We use standard, publicly-available code for solving this problem (Guerriero and Tseng, 2002).[2]

4.2 Non-Euclidean Dual Extragradient

Euclidean projections may not be easy to compute for many structured prediction problems or parameter spaces. The non-Euclidean version of the algorithm of Nesterov (2003) affords flexibility to use other types of (Bregman) projections. The basic idea is as follows. Let $d(\mathbf{u},\mathbf{u}')$ denote a suitable divergence function (see below for a definition) and define a proximal step operator:

$$T_\eta(\mathbf{u},\mathbf{s}) \equiv \arg\max_{\mathbf{u}'\in\mathcal{U}} [\mathbf{s}^\top(\mathbf{u}'-\mathbf{u}) - \frac{1}{\eta} d(\mathbf{u},\mathbf{u}')].$$

Intuitively, the operator tries to make a large step from $\mathbf{u}$ in the direction of $\mathbf{s}$ but not too large as measured by $d(\cdot,\cdot)$. Then the only change to the algorithm is to switch from using a Euclidean projection of a gradient step $\boldsymbol{\pi}_\mathcal{U}(\mathbf{u}+\frac{1}{\eta}\mathbf{s})$ to a proximal step in a direction of the gradient $T_\eta(\mathbf{u},\mathbf{s})$ (see Fig. 4):

Initialize: Choose $\hat{\mathbf{u}} \in \widetilde{\mathcal{U}}$, set $\mathbf{s}^{-1} = 0$.
Iteration t, $0 \le t \le \tau$:

$$\mathbf{v}_\mathbf{w} = T_\eta(\hat{\mathbf{u}}_\mathbf{w}, \mathbf{s}_\mathbf{w}^{t-1}); \qquad \mathbf{v}_{\mathbf{z}_i} = T_\eta(\hat{\mathbf{u}}_{\mathbf{z}_i}, \mathbf{s}_{\mathbf{z}_i}^{t-1}),\ \forall i;$$

$$\mathbf{u}_\mathbf{w}^t = T_\eta(\mathbf{v}_\mathbf{w}, -\left[\sum_i \mathbf{F}_i \mathbf{v}_{\mathbf{z}_i} - \mathbf{f}_i(\mathbf{y}_i)\right]); \qquad \mathbf{u}_{\mathbf{z}_i}^t = T_\eta(\mathbf{v}_{\mathbf{z}_i}, \left[\mathbf{F}_i^\top \mathbf{v}_\mathbf{w} + \mathbf{c}_i\right]),\ \forall i;$$

$$\mathbf{s}_\mathbf{w}^t = \mathbf{s}_\mathbf{w}^{t-1} - \left[\sum_i \mathbf{F}_i \mathbf{u}_{\mathbf{z}_i}^t - \mathbf{f}_i(\mathbf{y}_i)\right]; \qquad \mathbf{s}_{\mathbf{z}_i}^t = \mathbf{s}_{\mathbf{z}_i}^{t-1} + \left[\mathbf{F}_i^\top \mathbf{u}_\mathbf{w}^t + \mathbf{c}_i\right],\ \forall i.$$

Output: $\bar{\mathbf{u}}^\tau = \frac{1}{\tau+1}\sum_{t=0}^{\tau} \mathbf{u}^t$.

Figure 4: Non-Euclidean dual extragradient.

To define the range of possible divergence functions and to state convergence properties of the algorithm, we will need some more definitions. We follow the development of Nesterov (2003). Given a norm $||\cdot||^\mathcal{W}$ on $\mathcal{W}$ and norms $||\cdot||^{\mathcal{Z}_i}$ on $\mathcal{Z}_i$, we combine them into a norm on $\mathcal{U}$ as

$$||\mathbf{u}|| = \max(||\mathbf{w}||^\mathcal{W}, ||\mathbf{z}_1||^{\mathcal{Z}_1}, \ldots, ||\mathbf{z}_m||^{\mathcal{Z}_m}).$$

We denote the dual of $\mathcal{U}$ (the vector space of linear functions on $\mathcal{U}$) as $\mathcal{U}^*$. The norm $||\cdot||$ on the space $\mathcal{U}$ induces the dual norm $||\cdot||_*$ for all $\mathbf{s} \in \mathcal{U}^*$:

$$||\mathbf{s}||_* \equiv \max_{\mathbf{u}\in\mathcal{U}, ||\mathbf{u}||\le 1} \mathbf{s}^\top \mathbf{u}.$$

2. Available from http://www.math.washington.edu/∼tseng/netflowg_nl/.

The Lipschitz constant with respect to this norm (used to set $\eta = 1/L$) is

$$L \equiv \max_{\mathbf{u},\mathbf{u}' \in \mathcal{U}} \frac{||\mathbf{F}(\mathbf{u}-\mathbf{u}')||_*}{||\mathbf{u}-\mathbf{u}'||}.$$

The dual extragradient algorithm adjusts to the geometry induced by the norm by making use of Bregman divergences. We assume a strongly convex function $h(\mathbf{u})$:

$$h(\alpha\mathbf{u}+(1-\alpha)\mathbf{u}') \leq \alpha h(\mathbf{u}) + (1-\alpha)h(\mathbf{u}') - \alpha(1-\alpha)\frac{\sigma}{2}||\mathbf{u}-\mathbf{u}'||^2, \qquad \forall \mathbf{u},\mathbf{u}',\alpha \in [0,1],$$

for some $\sigma > 0$, the convexity parameter of $h(\cdot)$. This function is constructed from strongly convex functions on each of the spaces $\mathcal{W}$ and $\mathcal{Z}_i$ by a simple sum: $h(\mathbf{u}) = h(\mathbf{w}) + \sum_i h(\mathbf{z}_i)$. Its conjugate is defined as:

$$h^*(\mathbf{s}) \equiv \max_{\mathbf{u} \in \mathcal{U}} [\mathbf{s}^\top \mathbf{u} - h(\mathbf{u})].$$

Since $h(\cdot)$ is strongly convex, $h^*(\mathbf{u})$ is well-defined and differentiable at any $\mathbf{s} \in \mathcal{U}^*$. We define

$$\widetilde{\mathcal{U}} \equiv \{\nabla h^*(\mathbf{s}) : \mathbf{s} \in \mathcal{U}^*\}.$$

We further assume that $h(\cdot)$ is differentiable at any $\mathbf{u} \in \widetilde{\mathcal{U}}$; since it is also strongly convex, for any two points $\mathbf{u} \in \widetilde{\mathcal{U}}$ and $\mathbf{u}' \in \mathcal{U}$ we have

$$h(\mathbf{u}') \geq h(\mathbf{u}) + \nabla h(\mathbf{u})^\top (\mathbf{u}'-\mathbf{u}) + \frac{\sigma}{2}||\mathbf{u}'-\mathbf{u}||^2,$$

and we can define the Bregman divergence:

$$d(\mathbf{u},\mathbf{u}') = h(\mathbf{u}') - h(\mathbf{u}) - \nabla h(\mathbf{u})^\top (\mathbf{u}'-\mathbf{u}).$$

Note that when $||\cdot||$ is the 2-norm, we can use $h(\mathbf{u}) = \frac{1}{2}||\mathbf{u}||_2^2$, which has convexity parameter $\sigma = 1$, and induces the usual squared Euclidean distance $d(\mathbf{u},\mathbf{u}') = \frac{1}{2}||\mathbf{u}-\mathbf{u}'||_2^2$. When $||\cdot||$ is the 1-norm, we can use the negative entropy $h(\mathbf{u}) = -\mathrm{H}(\mathbf{u})$ (say if $\mathcal{U}$ is a simplex), which also has $\sigma = 1$ and recovers the Kullback-Leibler divergence $d(\mathbf{u},\mathbf{u}') = \mathrm{KL}(\mathbf{u}'||\mathbf{u})$.

With these definitions, the convergence bound in Eq. (19) applies to the non-Euclidean setup, but now the divergence radii are measured using Bregman divergence and the Lipschitz constant is computed with respect to a different norm.

EXAMPLE 1: L_1 REGULARIZATION

Suppose $\mathcal{W} = \{\mathbf{w} : ||\mathbf{w}||_1 \leq \gamma\}$. We can transform this constraint set into a simplex constraint by the following variable transformation. Let $\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^-$, $v_0 = 1 - ||\mathbf{w}||_1/\gamma$, and $\mathbf{v} \equiv (v_0, \mathbf{w}^+/\gamma, \mathbf{w}^-/\gamma)$. Then $\mathcal{V} = \{\mathbf{v} : \mathbf{v} \geq 0; \mathbf{1}^\top \mathbf{v} = 1\}$ corresponds to $\mathcal{W}$. We define $h(\mathbf{v})$ as the negative entropy of $\mathbf{v}$:

$$h(\mathbf{v}) = \sum_d v_d \log v_d.$$

The resulting conjugate function and its gradient are given by

$$h^*(\mathbf{s}) = \log \sum_d e^{s_d}; \qquad \frac{\partial h^*(\mathbf{s})}{\partial s_d} = \frac{e^{s_d}}{\sum_d e^{s_d}}.$$

Hence, the gradient space of $h^*(\mathbf{s})$ is the interior of the simplex, $\widetilde{\mathcal{V}} = \{\mathbf{v} : \mathbf{v} > 0; \mathbf{1}^\top \mathbf{v} = 1\}$. The corresponding Bregman divergence is the standard Kullback-Leibler divergence

$$d(\mathbf{v}, \mathbf{v}') = \sum_d v'_d \log \frac{v'_d}{v_d}, \quad \forall \mathbf{v} \in \widetilde{\mathcal{V}}, \mathbf{v}' \in \mathcal{V},$$

and the Bregman proximal step or projection, $\tilde{\mathbf{v}} = T_\eta(\mathbf{v}, \mathbf{s}) = \arg\max_{\mathbf{v}' \in \mathbf{v}} [\mathbf{s}^\top \mathbf{v}' - \frac{1}{\eta} d(\mathbf{v}, \mathbf{v}')]$ is given by a multiplicative update:

$$\tilde{v}_d = \frac{v_d e^{\eta s_d}}{\sum_d v_d e^{\eta s_d}}.$$

Note that we cannot choose $\hat{\mathbf{u}}_\mathbf{v} = (1, \mathbf{0}, \mathbf{0})$ as the center of $\widetilde{\mathcal{V}}$—given that the updates are multiplicative the algorithm will not make any progress in this case. In fact, this choice is precluded by the constraint that $\hat{\mathbf{u}}_\mathbf{v} \in \widetilde{\mathcal{V}}$, not just $\hat{\mathbf{u}}_\mathbf{v} \in \mathcal{V}$. A reasonable choice is to set $\hat{\mathbf{u}}_\mathbf{v}$ to be the center of the simplex $\mathcal{V}$, $\hat{\mathbf{u}}_{v_d} = \frac{1}{|\mathcal{V}|} = \frac{1}{2|\mathcal{W}|+1}$.

EXAMPLE 2: TREE-STRUCTURED MARGINALS

Consider the case in which each example i corresponds to a tree-structured Markov network, and $\mathcal{Z}_i$ is defined by the normalization and marginalization constraints in Eq. (2) and Eq. (3) respectively. These constraints define the space of valid marginals. For simplicity of notation, we assume that we are dealing with a single example i and drop the explicit index i. Let us use a more suggestive notation for the components of $\mathbf{z}$: $z_j(\alpha) = z_{j\alpha}$ and $z_{jk}(\alpha, \beta) = z_{jk\alpha\beta}$. We can construct a natural joint probability distribution via

$$P_\mathbf{z}(\mathbf{y}) = \prod_{jk \in \mathcal{E}} z_{jk}(y_j, y_k) \prod_{j \in \mathcal{V}} (z_j(y_j))^{1-q_j},$$

where q_j is the number of neighbors of node j. Now $\mathbf{z}$ defines a point on the simplex of joint distributions over $\mathcal{Y}$, which has dimension $|\mathcal{Y}|$. One natural measure of complexity in this enlarged space is the 1-norm. We define $h(\mathbf{z})$ as the negative entropy of the distribution represented by $\mathbf{z}$:

$$h(\mathbf{z}) = \sum_{jk \in \mathcal{E}} \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk}(\alpha, \beta) \log z_{jk}(\alpha, \beta) + (1 - q_j) \sum_{j \in \mathcal{V}} \sum_{\alpha \in \mathcal{D}_j} z_j(\alpha) \log z_j(\alpha).$$

The resulting $d(\mathbf{z}, \mathbf{z}')$ is the Kullback-Leibler divergence $\mathrm{KL}(P_{\mathbf{z}'} || P_\mathbf{z})$. The corresponding Bregman step or projection operator, $\tilde{\mathbf{z}} = T_\eta(\mathbf{z}, \mathbf{s}) = \arg\max_{\mathbf{z}' \in \mathcal{Z}} [\mathbf{s}^\top \mathbf{z}' - \frac{1}{\eta} \mathrm{KL}(P_{\mathbf{z}'} || P_\mathbf{z})]$ is given by a multiplicative update on the space of distributions:

$$P_{\tilde{\mathbf{z}}}(\mathbf{y}) = \frac{1}{Z} P_\mathbf{z}(\mathbf{y}) e^{\eta [\sum_{jk} s_{jk}(y_j, y_k) + \sum_j s_j(y_j)]} = \frac{1}{Z} \prod_{jk} z_{jk}(y_j, y_k) e^{\eta s_{jk}(y_j, y_k)} \prod_j (z_j(y_j))^{1-q_j} e^{\eta s_j(y_j)},$$

where we use the same indexing for the dual space vector $\mathbf{s}$ as for $\mathbf{z}$ and Z is a normalization constant. Hence, to obtain the projection $\tilde{\mathbf{z}}$, we compute the node and edge marginals of the distribution $P_{\tilde{\mathbf{z}}}(\mathbf{y})$ via the standard sum-product dynamic programming algorithm using the node and edge potentials defined above. Note that the form of the multiplicative update of the projection resembles that of exponentiated gradient. As in the example above, we cannot let $\hat{\mathbf{u}}_\mathbf{z}$ be a corner (or any boundary point) of the simplex since $\widetilde{\mathcal{Z}}$ does not include it. A reasonable choice for $\hat{\mathbf{u}}_\mathbf{z}$ would be either the center of the simplex or a point near the target structure but in the interior of the simplex.

5. Memory-Efficient Formulation

Consider the memory requirements of the algorithm. The algorithm maintains the vector $\mathbf{s}^\tau$ as well as the running average, $\bar{\mathbf{u}}^\tau$, a total dimensionality of $|\mathcal{W}|+|\mathcal{Z}|$. Note, however, that these vectors are related very simply by:

$$\mathbf{s}^\tau = -\sum_{t=0}^{\tau}(\mathbf{F}\mathbf{u}^t - \mathbf{a}) = -(\tau+1)(\mathbf{F}\bar{\mathbf{u}}^\tau - \mathbf{a}).$$

So it suffices to only maintain the running average $\bar{\mathbf{u}}^\tau$ and reconstruct $\mathbf{s}$ as needed.

In problems in which the number of examples, m, is large we can take advantage of the fact that the memory needed to store the target structure $\mathbf{y}_i$ is often much smaller than the corresponding vector $\mathbf{z}_i$. For example, for word alignment, we need $O(|\mathcal{V}_i^s|\log|\mathcal{V}_i^t|)$ bits to encode a matching $\mathbf{y}_i$ by using roughly $\log\mathcal{V}_i^t$ bits per node in $\mathcal{V}_i^s$ to identify its match. By contrast, we need $|\mathcal{V}_i^s||\mathcal{V}_i^t|$ floating numbers to maintain $\mathbf{z}_i$. The situation is worse in context-free parsing, where a parse tree $\mathbf{y}_i$ requires space linear in the sentence length and logarithmic in grammar size, while $|\mathcal{Z}_i|$ is the product of the grammar size and the cube of the sentence length.

Note that from $\bar{\mathbf{u}}^\tau = (\bar{\mathbf{u}}_\mathbf{w}^\tau, \bar{\mathbf{u}}_\mathbf{z}^\tau)$, we only care about $\bar{\mathbf{u}}_\mathbf{w}^\tau$, the parameters of the model, while the other component, $\bar{\mathbf{u}}_\mathbf{z}^\tau$, maintains the state of the algorithm. Fortunately, we can eliminate the need to store $\bar{\mathbf{u}}_\mathbf{z}$ by maintaining it implicitly, at the cost of storing a vector of size $|\mathcal{W}|$. This allows us to essentially have the same small memory footprint of online-type learning methods, where a single example is processed at a time and only a vector of parameters is maintained. In particular, instead of maintaining the entire vector $\bar{\mathbf{u}}^t$ and reconstructing $\mathbf{s}^t$ from $\bar{\mathbf{u}}^t$, we can instead store only $\bar{\mathbf{u}}_\mathbf{w}^t$ and $\mathbf{s}_\mathbf{w}^t$ between iterations, since

$$\mathbf{s}_{\mathbf{z}_i}^t = (t+1)(\mathbf{F}_i^\top \bar{\mathbf{u}}_\mathbf{w}^t + \mathbf{c}_i).$$

The diagram in Fig. 5 illustrates the process and the algorithm is summarized in Fig. 6. We use two "temporary" variables $\mathbf{v}_\mathbf{w}$ and $\mathbf{r}_\mathbf{w}$ of size $|\mathcal{W}|$ to maintain intermediate quantities. The additional vector $\mathbf{q}_\mathbf{w}$ shown in Fig. 5 is introduced only to allow the diagram to be drawn in a simplified manner; it can be eliminated by using $\mathbf{s}_\mathbf{w}$ to accumulate the gradients as shown in Fig. 6. The total amount of memory needed is thus four times the number of parameters plus memory for a single example $(\mathbf{v}_{\mathbf{z}_i}, \mathbf{u}_{\mathbf{z}_i})$. We assume that we do not need to store $\hat{\mathbf{u}}_{\mathbf{z}_i}$ explicitly but can construct it efficiently from $(\mathbf{x}_i, \mathbf{y}_i)$.

Note that in case the dimensionality of the parameter space is much larger than the dimensionality of $\mathcal{Z}$, we can use a similar trick to only store variables of the size of $\mathbf{z}$. In fact, if $\mathcal{W} = \{\mathbf{w} : ||\mathbf{w}||_2 \leq \gamma\}$ and we use Euclidean projections onto $\mathcal{W}$, we can exploit kernels to define infinite-dimensional feature spaces and derive a kernelized version of the algorithm.

6. Experiments

In this section we describe experiments focusing on two of the structured models we described earlier: bipartite matchings for word alignments and restricted potential Markov nets for 3D segmentation.[3] We compared three algorithms: the dual extragradient (`dual-ex`), the averaged projected gradient (`proj-grad`) defined in Eq. (16), and the averaged perceptron (Collins, 2002). For

3. Software implementing our dual extragradient algorithm can be found at `http://www.cs.berkeley.edu/~slacoste/research/dualex`.

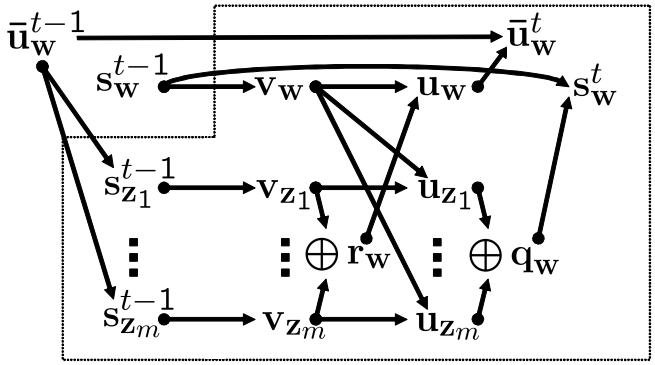


Figure 5: Dependency diagram for memory-efficient dual extragradient. The dotted box represents the computations of an iteration of the algorithm. Only $\bar{\mathbf{u}}_{\mathbf{w}}^t$ and $\mathbf{s}_{\mathbf{w}}^t$ are kept between iterations. Each example is processed one by one and the intermediate results are accumulated as $\mathbf{r}_{\mathbf{w}} = \mathbf{r}_{\mathbf{w}} - \mathbf{F}_i \mathbf{v}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i)$ and $\mathbf{q}_{\mathbf{w}} = \mathbf{q}_{\mathbf{w}} - \mathbf{F}_i \mathbf{u}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i)$. Details shown in Fig. 6, except that intermediate variables $\mathbf{u}_{\mathbf{w}}$ and $\mathbf{q}_{\mathbf{w}}$ are only used here for pictorial clarity.

Initialize: Choose $\hat{\mathbf{u}} \in \widetilde{\mathcal{U}}$, $\mathbf{s}_{\mathbf{w}} = 0$, $\bar{\mathbf{u}}_{\mathbf{w}} = 0$, $\eta = 1/L$.
Iteration t, $0 \le t \le \tau$:
$\mathbf{v}_{\mathbf{w}} = T_\eta(\hat{\mathbf{u}}_{\mathbf{w}}, \mathbf{s}_{\mathbf{w}})$; $\mathbf{r}_{\mathbf{w}} = 0$.
Example i, $1 \le i \le m$:

$$\mathbf{v}_{\mathbf{z}_i} = T_\eta(\hat{\mathbf{u}}_{\mathbf{z}_i}, t(\mathbf{F}_i^\top \bar{\mathbf{u}}_{\mathbf{w}} + \mathbf{c}_i)); \qquad \mathbf{r}_{\mathbf{w}} = \mathbf{r}_{\mathbf{w}} - \mathbf{F}_i \mathbf{v}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i);$$
$$\mathbf{u}_{\mathbf{z}_i} = T_\eta(\mathbf{v}_{\mathbf{z}_i}, \mathbf{F}_i^\top \mathbf{v}_{\mathbf{w}} + \mathbf{c}_i); \qquad \mathbf{s}_{\mathbf{w}} = \mathbf{s}_{\mathbf{w}} - \mathbf{F}_i \mathbf{u}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i).$$

$\bar{\mathbf{u}}_{\mathbf{w}} = \frac{t\bar{\mathbf{u}}_{\mathbf{w}} + T_\eta(\mathbf{v}_{\mathbf{w}}, \mathbf{r}_{\mathbf{w}})}{t+1}$.
Output $\mathbf{w} = \bar{\mathbf{u}}_{\mathbf{w}}$.

Figure 6: Memory-efficient dual extragradient.

`dual-ex` and `proj-grad`, we used Euclidean projections, which can be formulated as min-cost quadratic flow problems. We used $\mathbf{w} = 0$ and $\mathbf{z}_i$ corresponding to $\mathbf{y}_i$ as the centroid $\hat{\mathbf{u}}$ in `dual-ex` and as the starting point of `proj-grad`.

In our experiments, we consider standard L_2 regularization, $\{||\mathbf{w}||_2 \le \gamma\}$. A question which arises in practice is how to choose the regularization parameter γ. The typical approach is to run the algorithm for several values of the regularization parameter and pick the best model using a validation set. This can be quite expensive, though, and several recent papers have explored techniques for obtaining the whole regularization path, either exactly (Hastie et al., 2004), or approximately using path following techniques (Rosset, 2004). Instead, we run the algorithm without regularization ($\gamma = \infty$) and track its performance on the validation set, selecting the model with best performance. For comparison, whenever feasible with the available memory, we used commercial software to compute points on the regularization path. As we discuss below, the dual extragradient algorithm approximately follows the regularization path in our experiments (in terms of the training objective

and test error) in the beginning and the end of the range of γ and often performs better in terms of generalization error in the mid-range.

6.1 Object Segmentation

We tested our algorithm on a 3D scan segmentation problem using the class of Markov networks with regular potentials that were described above. The dataset is a challenging collection of cluttered scenes containing articulated wooden puppets (Anguelov et al., 2005). It contains eleven different single-view scans of three puppets of varying sizes and positions, with clutter and occluding objects such as rope, sticks and rings. Each scan consists of around $7,000$ points. Our goal was to segment the scenes into two classes—puppet and background. We use five of the scenes for our training data, three for validation and three for testing. Sample scans from the training and test set can be seen at `http://www.cs.berkeley.edu/~taskar/3DSegment/`. We computed spin images of size 10×5 bins at two different resolutions, then scaled the values and performed PCA to obtain 45 principal components, which comprised our node features. We used the surface links output by the scanner as edges between points and for each edge only used a single feature, set to a constant value of 1 for all edges. This results in all edges having the same potential. The training data contains approximately $37,000$ nodes and $88,000$ edges. We used standard Hamming distance for our loss function $\ell(\mathbf{y}_i, \mathbf{y}'_i)$.

We compared the performance of the dual extragradient algorithm along its unregularized path to solutions of the regularized problems for different settings of the norm.[4] For dual extragradient, the stepsize is set to $\eta = 1/||\mathbf{F}||_2 \approx 0.005$. We also compared to a variant of the averaged perceptron algorithm (Collins, 2002), where we use the batch updates to stabilize the algorithm, since we only have five training examples. We set the learning rate to 0.0007 by trying several values and picking the best value based on the validation data.

In Fig. 7(a) we track the hinge loss on the training data:

$$\sum_i \max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i). \tag{20}$$

The hinge loss of the regularization path (`reg-path`) is the minimum loss for a given norm, and hence is always lower than the hinge loss of the other algorithms. However, as the norm increases and the model approaches the unregularized solution, `dual-ex` loss tends towards that of `reg-path`. Note that `proj-grad` behaves quite erratically in the range of the norms shown. Fig. 7(b) shows the growth of the norm as a function of iteration number for `dual-ex` and `ave-perc`. The unregularized dual extragradient seems to explore the range of models (in terms on their norm) on the regularization path more thoroughly than the averaged perceptron and eventually asymptotes to the unregularized solution, while `proj-grad` quickly achieves very large norm.

Fig. 7(c) and Fig. 7(d) show validation and test error for the three algorithms. The best validation and test error achieved by the `dual-ex` and `ave-perc` algorithms as well as `reg-path` are fairly close, however, this error level is reached at very different norms. Since the number of scenes in the validation and test data is very small (three), because of variance, the best norm on validation is not very close to the best norm on the test set. Selecting the best model on the validation set leads to test errors of 3.4% for `dual-ex`, 3.5% for `ave-perc`, 3.6% for `reg-path` and 3.8% for `proj-grad`

4. We used CPLEX to solve the regularized problems and also to find the projections onto the min-cut polytope, since the min-cost quadratic flow code we used (Guerriero and Tseng, 2002) does not support negative flows on edges, which are needed in the formulation presented in Appendix A.

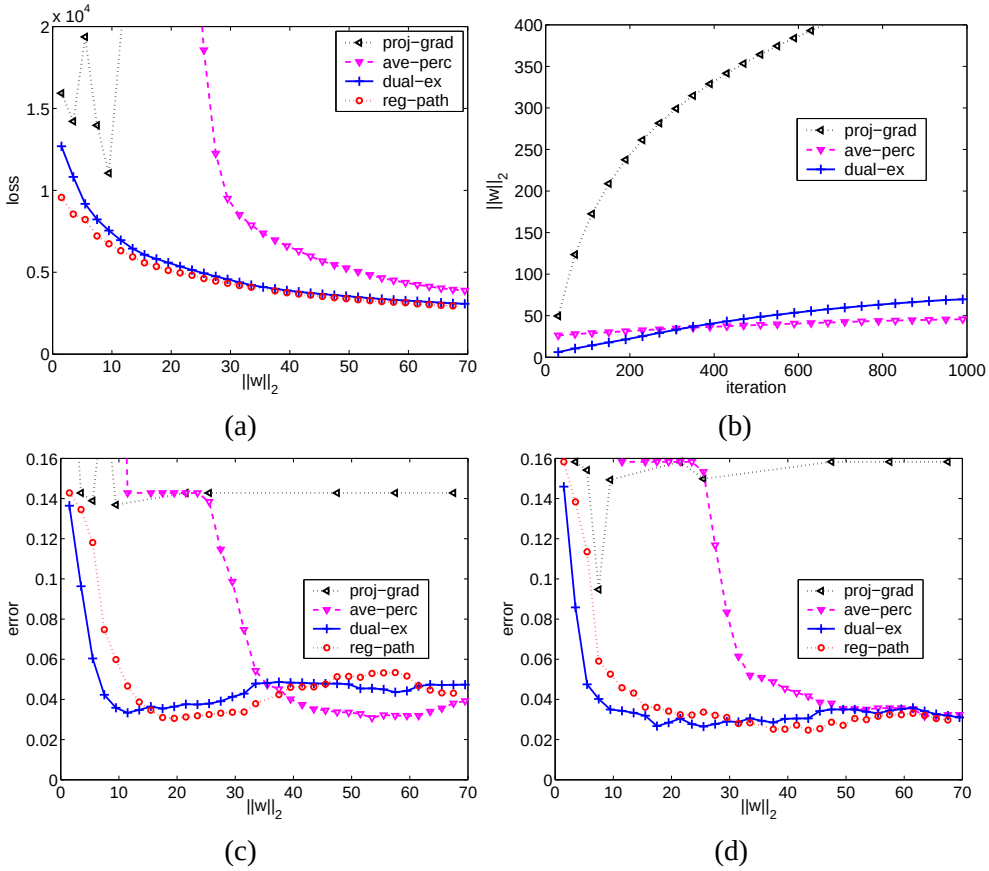


Figure 7: Object segmentation results: (a) Training hinge loss for the regularization path (`reg-path`), the averaged projected gradient (`proj-grad`), the averaged perceptron (`ave-perc`) and unregularized dual extragradient (`dual-ex`) vs. the norm of the parameters. (b) Norm of the parameters vs. iteration number for the three algorithms. (c) Validation error vs. the norm of the parameters. (d) Test error vs. the norm of the parameters.

(`proj-grad` actually improves performance after the model norm is larger than 500, which is not shown in the graphs).

6.2 Word Alignment

We also tested our algorithm on word-level alignment using a data set from the 2003 NAACL set (Mihalcea and Pedersen, 2003), the English-French Hansards task. This corpus consists of 1.1M pairs of sentences, and comes with a validation set of 37 sentence pairs and a test set of 447 word-aligned sentences. The validation and test sentences have been hand-aligned (see Och and Ney, 2003) and are marked with both *sure* and *possible* alignments. Using these alignments, the *alignment error rate* (AER) is calculated as:

$$AER(A,S,P) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|},$$

where A is a set of proposed alignment pairs, S is the set of sure gold pairs, and P is the set of possible gold pairs (where $S \subseteq P$).

We experimented with two different training settings. In the first one, we split the original test set into 100 training examples and 347 test examples—this dataset is called the 'Gold' dataset. In the second setting, we used GIZA++ (Och and Ney, 2003) to produce IBM Model 4 alignments for the unlabeled sentence pairs. We took the intersection of the predictions of the English-to-French and French-to-English Model 4 alignments on the first 5000 sentence pairs from the 1.1M sentences in order to experiment with the scaling of our algorithm (training on 500, 1000 and 5000 sentences). The number of edges for 500, 1000 and 5000 sentences of GIZA++ were about 31,000, 99,000 and 555,000 respectively. We still tested on the 347 Gold test examples, and used the validation set to select the stopping point. The stepsize for the dual extragradient algorithm was chosen to be $1/||\mathbf{F}||_2$.

We used statistics computed on the 1.1M sentence pairs as the edge features for our model. A detailed analysis of the constructed features and corresponding error analysis is presented in Taskar et al. (2005b). Example features include: a measure of mutual information between the two words computed from their co-occurrence in the aligned sentences (Dice coefficient); the difference between word positions; character-based similarity features designed to capture cognate (and exact match) information; and identity of the top five most frequently occurring words. We used the structured loss $\ell(\mathbf{y}_i, \mathbf{y}'_i)$ defined in Eq. (10) with $(c^+, c^-) = (1, 3)$ (where 3 was selected by testing several values on the validation set). We obtained low recall when using equal cost for both type of errors because the number of positive edges is significantly smaller than the number of negative edges, and so it is safe (precision-wise) for the model to predict fewer edges, hurting the recall. Increasing the cost for false negatives solves this problem.

Fig. 8(a) and Fig. 8(e) compare the hinge loss of the regularization path with the evolution of the objective for the unregularized dual extragradient, averaged projected gradient and averaged perceptron algorithms when trained on the Gold data set, 500 sentences and 1000 sentences of the GIZA++ output respectively.[5] The dual extragradient path appears to follow the regularization path closely for $||\mathbf{w}|| \leq 2$ and $||\mathbf{w}|| \geq 12$. Fig. 8(b) compares the AER on the test set along the dual extragradient path trained on the Gold dataset versus the regularization path AER. The results on the validation set for each path are also shown. On the Gold data set, the minimum AER was reached roughly after 200 iterations.

Interestingly, the unregularized dual extragradient path seems to give better performance on the test set than that obtained by optimizing along the regularization path. The dominance of the dual extragradient path over the regularization path is more salient in figure 8(f) for the case where both are trained on 1000 sentences from the GIZA++ output. We conjecture that the dual extragradient method provides additional statistical regularization (compensating for the noisier labels of the GIZA++ output) by enforcing local smoothness of the path in parameter space.

The averaged projected gradient performed much better for this task than segmentation, getting somewhat close to the dual extragradient path as is shown in Fig. 8(c). The online version of the averaged perceptron algorithm varied significantly with the order of presentation of examples (up to five points of difference in AER between two orders). To alleviate this, we randomize the order of the points at each pass over the data. Fig. 8(d) shows that a typical run of averaged perceptron does somewhat worse than dual extragradient. The variance of the averaged perceptron performance for

5. The regularization path is obtained by using the commercial optimization software Mosek with the QCQP formulation of Eq. (15). We did not obtain the path in the case of 5000 sentences, as Mosek runs out of memory.

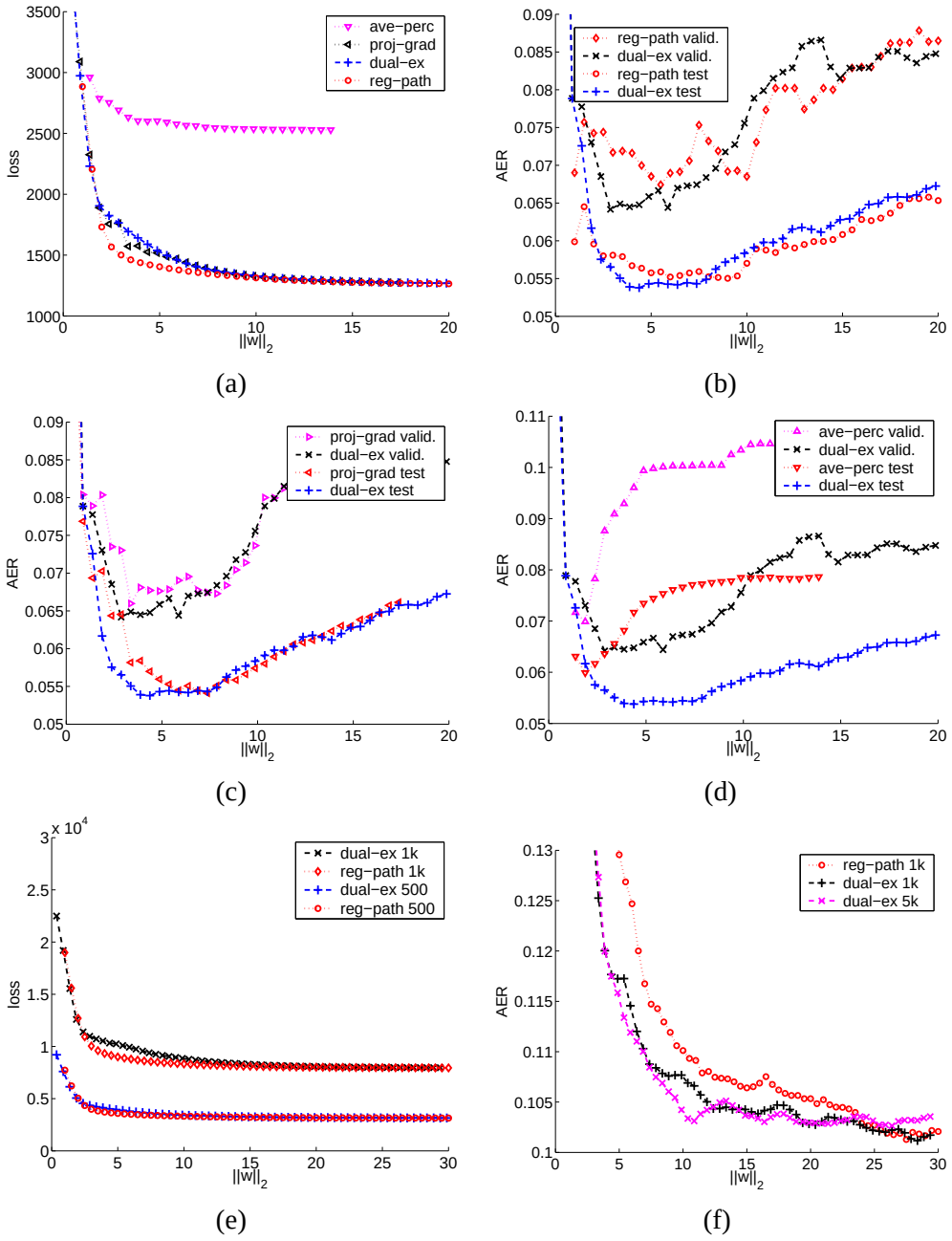


Figure 8: Word alignment results: (a) Training hinge loss for the three different algorithms and the regularization path on the Gold dataset. (b) AER for the unregularized dual extragradient (`dual-ex`) and the regularization path (`reg-path`) on the 347 Gold sentences (`test`) and the validation set (`valid`) when trained on the 100 Gold sentences; (c) Same setting as in (b), comparing `dual-ex` with the averaged projected-gradient (`proj-grad`); (d) Same setting as in (b), comparing `proj-grad` with the averaged perceptron (`ave-perc`); (e) Training hinge loss for `dual-ex` and `reg-path` on 500 and 1000 GIZA++ labeled sentences. (f) AER for `dual-ex` and `reg-path` tested on the Gold test set and trained on 1000 and 5000 GIZA++ sentences. The graph for 500 sentences is omitted for clarity.

different datasets and learning rate choices was also significantly higher than for dual extragradient, which is more stable numerically. The online version of the averaged perceptron converged very quickly to its minimum AER score; converging in as few as five iterations for the Gold training set. Selecting the best model on the validation set leads to test errors of 5.6% for `dual-ex`, 5.6% for `reg-path`, 5.8% for `proj-grad` and 6.1% for `ave-perc` on the Gold data training set.

The running time for 500 iterations of dual extragradient on a 3.4 Ghz Intel Xeon CPU with 4G of memory was roughly 10 minutes, 30 minutes and 3 hours for 500, 1000 and 5000 sentences, respectively, showing the favorable linear scaling of the algorithm (linear in the number of edges). Note, by way of comparison, that Mosek ran out of memory for more than 1500 training sentences.

The framework we have presented here supports much richer models for word alignment; for example, allowing finer-grained, feature-based fertility control (number of aligned words for each word) as well as inclusion of positive correlations between adjacent edges in alignments. These extensions are developed in Lacoste-Julien et al. (2006).

7. Conclusions

We have presented a general and simple solution strategy for large-scale structured prediction problems. Using a saddle-point formulation of the problem, we exploit the dual extragradient algorithm, a simple gradient-based algorithm for saddle-point problems (Nesterov, 2003). The factoring of the problem into optimization over the feasible parameter space $\mathcal{W}$ and feasible structured output space $\mathcal{Z}$ allows easy integration of complex parameter constraints that arise in estimation of restricted classes of Markov networks and other models.

Key to our approach is the recognition that the projection step in the extragradient algorithm can be solved by network flow algorithms for matchings and min-cuts (and dynamic programming for decomposable models). Network flow algorithms are among the most well-developed algorithms in the field of combinatorial optimization, and yield stable, efficient algorithmic platforms.

One of the key bottlenecks of large learning problems is the memory requirement of the algorithm. We have derived a version of the algorithm that only uses storage proportional to the number of parameters in the model, and is independent of the number of examples. We have exhibited the favorable scaling of this overall approach in two concrete, large-scale learning problems. It is also important to note that the general approach extends and adopts to a much broader class of problems by allowing the use of Bregman projections suitable to particular problem classes.

Acknowledgments

We would like to thank Paul Tseng for kindly answering our questions about the minimum cost quadratic flow implementation. We also wish to acknowledge support from the Defense Advanced Research projects Agency under contract NBCHD030010, and grants from Google and Microsoft Research.

Appendix A. Min-Cut Polytope Projections

Consider projection for a single example i:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \sum_{j\in\mathcal{V}} \frac{1}{2}(z'_j - z_j)^2 + \sum_{jk\in\mathcal{E}} \frac{1}{2}(z'_{jk} - z_{jk})^2 \\ \text{s.t.} \quad & 0 \le z_j \le 1, \quad \forall j \in \mathcal{V}; \quad z_j - z_k \le z_{jk}, \quad z_k - z_j \le z_{jk}, \quad \forall jk \in \mathcal{E}. \end{aligned} \tag{21}$$

Let $h_j^+(z_j) = \frac{1}{2}(z'_j - z_j)^2$ if $0 \le z_j$, else ∞. We introduce non-negative Lagrangian variables $\lambda_{jk}, \lambda_{kj}$ for the two constraints for each edge jk and λ_{j0} for the constraint $z_j \le 1$ each node j.

The Lagrangian is given by:

$$\begin{aligned} L(\mathbf{z},\lambda) \quad = \quad & \sum_{j\in\mathcal{V}} h_j^+(z_j) + \sum_{jk\in\mathcal{E}} \frac{1}{2}(z'_{jk} - z_{jk})^2 - \sum_{j\in\mathcal{V}} (1 - z_j)\lambda_{j0} \\ & - \sum_{jk\in\mathcal{E}} (z_{jk} - z_j + z_k)\lambda_{jk} \; - \sum_{jk\in\mathcal{E}} (z_{jk} - z_k + z_j)\lambda_{kj} \end{aligned}$$

Letting $\lambda_{0j} = \lambda_{j0} + \sum_{k:jk\in\mathcal{E}} (\lambda_{jk} - \lambda_{kj}) + \sum_{k:kj\in\mathcal{E}} (\lambda_{jk} - \lambda_{kj})$, note that

$$\sum_{j\in\mathcal{V}} z_j\lambda_{0j} = \sum_{j\in\mathcal{V}} z_j\lambda_{j0} + \sum_{jk\in\mathcal{E}} (z_j - z_k)\lambda_{jk} + \sum_{jk\in\mathcal{E}} (z_k - z_j)\lambda_{kj}.$$

So the Lagrangian becomes:

$$L(\mathbf{z},\lambda) = \sum_{j\in\mathcal{V}} \left[h_j^+(z_j) + z_j\lambda_{0j} - \lambda_{j0} \right] + \sum_{jk\in\mathcal{E}} \left[\frac{1}{2}(z'_{jk} - z_{jk})^2 - z_{jk}(\lambda_{jk} + \lambda_{kj}) \right].$$

Now, minimizing $L(\mathbf{z},\lambda)$ with respect to $\mathbf{z}$, we have

$$\min_{\mathbf{z}} L(\mathbf{z},\lambda) = \sum_{jk\in\mathcal{E}} q_{jk}(\lambda_{jk} + \lambda_{kj}) + \sum_{j\in\mathcal{V}} [q_{0j}(\lambda_{0j}) - \lambda_{j0}],$$

where $q_{jk}(\lambda_{jk} + \lambda_{kj}) = \min_{z_{jk}} \left[\frac{1}{2}(z'_{jk} - z_{jk})^2 - z_{jk}(\lambda_{jk} + \lambda_{kj}) \right]$ and $q_{0j}(\lambda_{0j}) = \min_{z_j} [h_j^+(z_j) + z_j\lambda_{0j}]$. The minimizing values of $\mathbf{z}$ are:

$$\begin{aligned} z_j^* \quad &= \quad \arg\min_{z_j} \left[h_j^+(z_j) + z_j\lambda_{0j} \right] = \begin{cases} 0 & \lambda_{0j} \ge z'_j; \\ z'_j - \lambda_{0j} & \lambda_{0j} \le z'_j; \end{cases} \\ z_{jk}^* \quad &= \quad \arg\min_{z_{jk}} \left[\frac{1}{2}(z'_{jk} - z_{jk})^2 - z_{jk}(\lambda_{jk} + \lambda_{kj}) \right] = z'_{jk} + \lambda_{jk} + \lambda_{kj}. \end{aligned}$$

Hence, we have:

$$\begin{aligned} q_{jk}(\lambda_{jk} + \lambda_{kj}) \quad &= \quad -z'_{jk}(\lambda_{jk} + \lambda_{kj}) - \frac{1}{2}(\lambda_{jk} + \lambda_{kj})^2 \\ q_{0j}(\lambda_{0j}) \quad &= \quad \begin{cases} \frac{1}{2}{z'_j}^2 & \lambda_{0j} \ge z'_j; \\ z'_j\lambda_{0j} - \frac{1}{2}\lambda_{0j}^2 & \lambda_{0j} \le z'_j. \end{cases} \end{aligned}$$

The dual of the projection problem is thus:

$$\begin{aligned} \max_{\lambda} \quad & \sum_{j\in\mathcal{V}} [q_{0j}(\lambda_{0j}) - \lambda_{j0}] + \sum_{jk\in\mathcal{E}} \left[-z'_{jk}(\lambda_{jk}+\lambda_{kj}) - \frac{1}{2}(\lambda_{jk}+\lambda_{kj})^2 \right] \\ \text{s.t.} \quad & \lambda_{j0} - \lambda_{0j} + \sum_{jk\in\mathcal{E}} (\lambda_{jk} - \lambda_{kj}) = 0, \ \forall j \in \mathcal{V}; \\ & \lambda_{jk}, \lambda_{kj} \geq 0, \ \forall jk \in \mathcal{E}; \ \ \lambda_{j0} \geq 0, \ \forall j \in \mathcal{V}. \end{aligned} \tag{22}$$

Interpreting λ_{jk} as flow from node j to node k, and λ_{kj} as flow from k to j and $\lambda_{j0}, \lambda_{0j}$ as flow from and to a special node 0, we can identify the constraints of Eq. (22) as conservation of flow constraints. The last transformation we need is to address the presence of cross-terms $\lambda_{jk}\lambda_{kj}$ in the objective. Note that in the flow conservation constraints, λ_{jk}, λ_{kj} always appear together as $\lambda_{jk} - \lambda_{kj}$. Since we are minimizing $(\lambda_{jk}+\lambda_{kj})^2$ subject to constraints on $\lambda_{jk} - \lambda_{kj}$, at least one of λ_{jk}, λ_{kj} will be zero at the optimum and the cross-terms can be ignored. Note that all λ variables are non-negative except for λ_{0j}'s. Many standard flow packages support this problem form, but we can also transform the problem to have all non-negative flows by introducing extra variables. The final form has a convex quadratic cost for each edge:

$$\begin{aligned} \min_{\lambda} \quad & \sum_{j\in\mathcal{V}} [-q_{0j}(\lambda_{0j}) + \lambda_{j0}] + \sum_{jk\in\mathcal{E}} \left[z'_{jk}\lambda_{jk} + \frac{1}{2}\lambda_{jk}^2 \right] + \sum_{jk\in\mathcal{E}} \left[z'_{jk}\lambda_{kj} + \frac{1}{2}\lambda_{kj}^2 \right] \\ \text{s.t.} \quad & \lambda_{j0} - \lambda_{0j} + \sum_{jk\in\mathcal{E}} (\lambda_{jk} - \lambda_{kj}) = 0, \ \forall j \in \mathcal{V}; \\ & \lambda_{jk}, \lambda_{kj} \geq 0, \ \forall jk \in \mathcal{E}; \ \ \lambda_{j0} \geq 0, \ \forall j \in \mathcal{V}. \end{aligned} \tag{23}$$

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden Markov support vector machines. In *International Conference on Machine Learning*. ACM Press, New York, 2003.

Drago Anguelov, Ben Taskar, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremey Heitz, and Andrew Y. Ng. Discriminative learning of Markov random fields for segmentation of 3D scan data. In *International Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, 2005.

Pierre Baldi, Jianlin Cheng, and Alessandro Vullo. Large-scale prediction of disulphide bond connectivity. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2005.

Peter L. Bartlett, Michael Collins, Ben Taskar, and David McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2005.

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 24:509–522, 2002.

Dimitri P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, MA, 1998.

Dimitri P. Bertsekas, Lazaros C. Polymenakos, and Paul Tseng. An ε-relaxation method for separable convex cost network flow problems. *SIAM Journal on Optimization*, 7(3):853–870, 1997.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington, DC, 2001.

Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics*, 18(3):606–635, 2005.

Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, 2002.

Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis*. Cambridge University Press, UK, 1998.

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society B*, 51:271–279, 1989.

Francesca Guerriero and Paul Tseng. Implementation and test of auction methods for solving generalized network flow problems with separable convex cost. *Journal of Optimization Theory and Applications*, 115:113–144, 2002.

Trevor Hastie, Saharon Rosset, Robert Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

Bingsheng He and Li-Zhi Liao. Improvements of some projection methods for monotone nonlinear variational inequalities. *Journal of Optimization Theory and Applications*, 112:111–128, 2002.

Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087–1116, 1993.

Vladimir Kolmogorov and Martin Wainwright. On the optimality of tree-reweighted max-product message passing. In *Uncertainty in Artificial Intelligence: Proceedings of the Twenty-First Conference*. Morgan Kaufmann, San Mateo, CA, 2005.

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized using graph cuts? *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 26:147–159, 2004.

G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747–756, 1976.

Sanjiv Kumar and Martial Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I. Jordan. Word alignment via quadratic assignment. In *Human Language Technology–North American Association for Computational Linguistics*, New York, NY, 2006.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*. Morgan Kaufmann, San Mateo, CA, 2001.

John Lafferty, Xiaojin Zhu, and Yan Liu. Kernel conditional random fields: Representation and clique selection. In *International Conference on Machine Learning*. ACM Press, New York, 2004.

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

Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

Evgeny Matusov, Richard Zens, and Herman Ney. Symmetric word alignments for statistical machine translation. In *Proceedings of the Twentieth International Conference on Computational Linguistics*, Geneva, Switzerland, 2004.

Rada Mihalcea and Ted Pedersen. An evaluation exercise for word alignment. In *Human Language Technology–North American Association for Computational Linguistics*, Edmonton, Canada, 2003.

Yurii Nesterov. Dual extrapolation and its application for solving variational inequalites and related problems. Technical report, CORE, Catholic University of Louvain, 2003.

Franz Och and Herman Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52, 2003.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.

Saharon Rosset. Tracking curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004.

Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin, 2003.

Ben Taskar. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, 2004.

Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative Markov networks. In *International Conference on Machine Learning*. ACM Press, New York, 2004a.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max margin Markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004b.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: a large margin approach. In *International Conference on Machine Learning*. ACM Press, New York, 2005a.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. A discriminative matching approach to word alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Vancouver, CA, 2005b.

Ben Taskar, Simon Lacoste-Julien, and Michael I. Jordan. Structured prediction via the extragradient method. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2006.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*. ACM Press, New York, 2004.

Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8: 189–201, 1979.

Martin Wainwright, Tommi Jaakkola, and Alan Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. In *Proceedings of the Allerton Conference on Communication, Control and Computing*, Allerton, IL, 2002.

Journal of Machine Learning Research 7 (2006) 1655–1686 Submitted 12/05; Revised 4/06; Published 8/06

Active Learning with Feedback on Both Features and Instances

Hema Raghavan[*] HEMA@CS.UMASS.EDU
140 Governor's Drive
University of Massachusetts
Amherst, MA 01003, USA

Omid Madani MADANI@YAHOO-INC.COM
Rosie Jones JONESR@YAHOO-INC.COM
Yahoo! Research
3333 Empire Ave, Burbank
CA 91504, USA

Editor: Isabelle Guyon

Abstract

We extend the traditional active learning framework to include feedback on features in addition to labeling instances, and we execute a careful study of the effects of feature selection and human feedback on features in the setting of text categorization. Our experiments on a variety of categorization tasks indicate that there is significant potential in improving classifier performance by feature re-weighting, beyond that achieved via membership queries alone (traditional active learning) if we have access to an *oracle* that can point to the important (most predictive) features. Our experiments on human subjects indicate that human feedback on feature relevance can identify a sufficient proportion of the most relevant features (over 50% in our experiments). We find that on average, labeling a feature takes much less time than labeling a document. We devise an algorithm that interleaves labeling features and documents which significantly accelerates standard active learning in our simulation experiments. Feature feedback can complement traditional active learning in applications such as news filtering, e-mail classification, and personalization, where the human teacher can have significant knowledge on the relevance of features.

Keywords: active learning, feature selection, relevance feedback, term feedback, text classification

1. Introduction

Automated text categorization has typically been tackled as a supervised machine learning problem (Sebastiani, 2002; Lewis, 1998). The training data should be fairly representative of the test data in order to learn a fairly accurate classifier. In document classification where categories can be as broad as *sports*, this means that a large amount of training data would be needed. The training data is often labeled by editors who are paid to do the job. Now consider a scenario where a user wants to organize documents on their desktop into categories of their choice. The user might be willing to engage in some amount of interaction to train the system, but may be less willing to label as much data as a paid editor. To build a generic text categorization system that could learn almost arbitrary categories based on an end user's changing needs and preferences, for example in applications such as news filtering and e-mail classification, the system should extract a large number of features. In

∗. This work was done in part when the author was at Yahoo! Research.

e-mail classification for example, any subset of the features extracted from the subject, the sender, and the text in the body of the message could be highly relevant. While algorithms such as Winnow (Littlestone, 1988) and Support Vector Machines (SVMs) (Joachims, 1998) are robust in the presence of large numbers of features, these algorithms still require a substantial amount of labeled data to achieve adequate performance.

Techniques such as active learning (Cohn et al., 1994), semi-supervised learning (Zhu, 2005), and transduction (Joachims, 1999) have been pursued with considerable success in reducing labeling requirements. In the standard active learning paradigm, learning proceeds sequentially, with the learning algorithm actively asking for the *labels* (categories) of some instances from a teacher (also referred to as membership queries). The objective is to ask the teacher to label the most informative instances in order to reduce labeling costs and accelerate the learning. Still, in text categorization applications in particular, active learning might be perceived to be too slow, especially since the teacher may have much prior knowledge on relevance of features for the task. Such knowledge may be more effectively communicated to the learner than mere labeling of whole documents. There has been very little work in supervised learning in which the teacher is queried on something other than whole instances.

One possibility is to ask the user questions about features. That users have useful prior knowledge which can be used to access information is evident in information retrieval tasks. In the information retrieval setting, the user issues a query, that is, states a few words (features) indicating her information need. Thereafter, feedback which may be either at a term or at a document level may be incorporated. In fact, even in traditional supervised learning, the editors may use keyword based search to locate the initial training instances [1]. However, traditional supervised learning tends to ignore this knowledge of features that the user has, once a set of training instances have been obtained. In experiments in this paper we study the benefits and costs of feature feedback via humans on active learning.

We try to find a marriage between approaches to incorporating user feedback from machine learning and information retrieval and show that active learning should be a twofold process – at the term-level and at the document-level. We find that people have a good intuition for important features in text classification tasks, since features are typically words, and the categories to learn may often be approximated by some disjunction or conjunction of a subset of the features. We show that human knowledge on features can indeed increase active learning efficiency and accelerate training significantly in the initial stages of learning. This has applications in e-mail classification and news filtering where the user has knowledge of the relevance of features and a willingness to label some (as few as possible) documents in order to build a system that suits her needs.

This paper extends our previous work in employing such a two-tiered approach to active learning (Raghavan et al., 2005). We state the active learning problems that we address and present our approach to use feedback on both features and instances to solve the problems in Section 2. We give the details of the implementations in Section 3. In Section 4 we describe the data and metrics we will use to evaluate the performance of active learning. We obtain a sense of the extent of the improvement possible via feature feedback by defining and using a feature oracle. The oracle and the experiments are described in Section 2, and the results are reported in Section 5. In section 6 we show that humans can indeed identify useful features. Furthermore, we find that labeling a feature

1. See `http://projects.ldc.upenn.edu/TDT4/Annotation/label_instructions.html`. The annotators at the LDC (Linguistic Data Consortium, home-page: `http://ldc.upenn.edu`) use a combination of techniques like nearest neighbors and creative search to annotate corpora for the Topic Detection and Tracking (Allan, 2002) task.

takes one fifth of the time of labeling a document. In Section 6.2 we show that the human-chosen features significantly accelerate learning in experiments that simulate human feedback in an active learning loop. We discuss related work in Section 7 and conclude in Section 8.

Standard Active Learning

Input: T (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances), init_size (number of random feedback iterations)
Output: $\mathcal{M}_T$ (Model)

$t = 1$; $\mathcal{U}_0 = \mathcal{U}$; $\mathcal{M}_0$ =NULL;
1. While $t \leq$ init_size
   a. $\langle X_t, \mathcal{U}_t \rangle$ = Instance_Selection($\mathcal{M}_0$, $\mathcal{U}_{t-1}$, *random*)
   b. Teacher assigns label Y_t to X_t
   d. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
   c. $t++$
2. While $t \leq T$
   a. $\langle X_t, \mathcal{U}_t \rangle$ = Instance_Selection($\mathcal{M}_{t-1}$, $\mathcal{U}_{t-1}$, *uncertain*)
   b. Teacher assigns label Y_t to X_t
   c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
   d. $t++$
Return $\mathcal{M}_T$

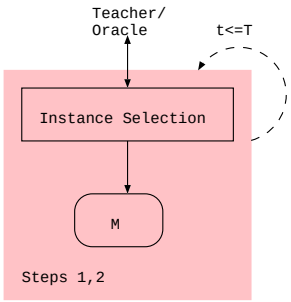


Figure 1: Algorithm and block diagram for traditional active learning where the system asks for feedback on instances only (**System 1**).

2. Active Learning

For background on the use of machine learning in automated text categorization as well as active learning, we refer the reader to the works of Sebastiani (2002) and Lewis and Catlett (1994). Active learning techniques are sequential learning methods that are designed to reduce manual training costs in achieving adequate learning performance. Active learning methods reduce costs by requesting training feedback selectively and intelligently from a *teacher*. The teacher is a human in the text categorization domain. The teacher may also be called the *user*, especially when the teacher training the model is the same as the person using it, for example a user who is training a personalized news filtering system. Traditionally in active learning the teacher is asked *membership queries* which are questions on the class labels or categories of selected instances (documents in our case).

The teacher is sometimes referred to as an oracle in the literature (Baum and Lang, 1992). We will also use the term oracle to refer to a source that gives feedback on instances and/or features, but in this paper we make a distinction between teacher and oracle. We will reserve the term teacher or user to refer to a real human, whose feedback may not be perfect, and we use the term oracle to refer to a source whose feedback is (close to) perfect for speeding active learning. See Section 2.1 for a longer discussion of the distinction between the two.

A typical algorithm for active learning and a block diagram are shown in Figure 1. An *instance* X (which is a document in our case) belongs to a *class* Y. X is represented as a vector $x_1 ... x_N$ of *features*, where N is the total number of features. The features we use for documents are words, bi-grams (adjacent pairs of words) and tri-grams (adjacent triples of words), since these have consistently been found to work well for topic classification. The value of x_j is the number of occurrences of term i in document X. We work on binary *one-versus-rest* classification. Therefore the value of Y for each learning problem of interest is either -1 or 1, signaling whether the instance belongs to the category of interest, or not. An instance in the document collection is *unlabeled* if the algorithm does not know its *label* (Y value). The active learner may have access to all or a subset of the unlabeled instances. This subset is called the *pool* (denoted by $\mathcal{U}$).

Active Learning Augmented with Feature Feedback

Input: T (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances), init_size (number of random feedback iterations)
Output: $\mathcal{M}_T$ (Model)

$t = 1$; $\mathcal{U}_0 = \mathcal{U}$; $\mathcal{M}_0$ =NULL;
1. While $t \leq$ init_size
   a.$\langle X_t, \mathcal{U}_t \rangle$ = Instance_Selection($\mathcal{M}_0$, $\mathcal{U}_{t-1}$, *random*)
   b. Teacher assigns label Y_t to X_t
   c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
   d. $t++$
2. While $t \leq T$
   a. $\langle X_t, \mathcal{U}_t \rangle$=
      Instance_Selection($\mathcal{M}_{t-1}$, $\mathcal{U}_{t-1}$, *uncertain*)
   b. Teacher assigns label Y_t to X_t
   c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
   d. i. $\{F1, ..., F_f\}$ = Feature_Selection($\mathcal{M}_t$, $\mathcal{U}_t$)
      ii. Teacher selects $\{F_1, .., F_k\} \subseteq \{F_1, ..., F_f\}$
   e. Incorporate_Feature_Feedback($\mathcal{M}_t$, $\{F_1, ..., F_k\}$)
   c. $t++$
Return $\mathcal{M}_T$.

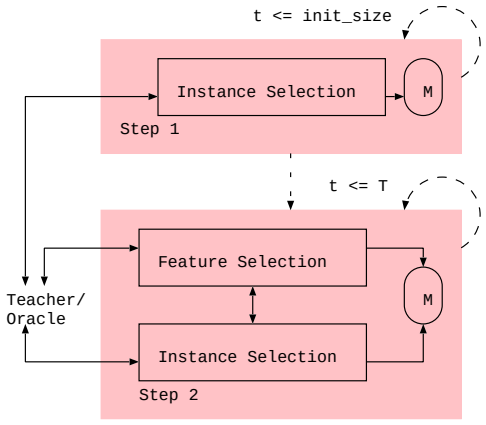


Figure 2: An active learning system where feedback on features is also requested (**System 2**).

The algorithm begins by training the classifier or model $\mathcal{M}$ on some initial set of labeled instances of size *init_size*. The subscript t on $\mathcal{M}$, $\mathcal{U}$, X and Y correspond to the value when t instances have been labeled. The initial set is picked by a random sampling procedure (step 1) from $\mathcal{U}$. The parameter *random* is passed to it. Sometimes one may use keyword based search or some other procedure in place of random sampling. Next, active learning begins. In each iteration of active learning the learner selects an instance from $\mathcal{U}$ using some criterion (*e.g.*, a measure of informativeness) and asks the teacher to label it (step 2.a). In a popular active learning method, called uncertainty sampling, the classifier selects the most *uncertain* instance (Lewis and Catlett, 1994), for a given model (M) and a pool of unlabeled instances ($\mathcal{U}$). The newly labeled instance is added to the set of labeled

instances and the classifier is retrained (step 2.c). The teacher is queried a total of T times. The *train_classifier* subroutine uses the labeled data as training, as well as the model ($\mathcal{M}$) learned in a previous iteration, allowing for the case of incremental training (Domeniconi and Gunopulos, 2001) or the case when the model may be initialized by prior knowledge (Wu and Srihari, 2004).

We will also consider the variant in which instances are picked uniformly at random in all iterations, which we call *random sampling* (it is equivalent to regular supervised learning on a random sample of data). In the pseudo-code in Figure 1, random sampling corresponds to the case when $init\_size > T$.

2.1 Our Proposal: Feature Feedback and Instance Feedback in Tandem

In this paper we propose to extend the traditional active learning framework to engage the teacher in providing feedback on features in addition to instances. A realization of this idea is system 2 shown in Figure 2, where the active learner not only queries the teacher on an informative document, but also presents a list of f features for the teacher to judge (step 2.d) at each iteration. The simplest implementation of such a system can consist of one where $f = |X|$ (the length of the document X), and where the user is simply asked to highlight relevant words or phrases (features) or passages while reading the document in order to label the document (step 2b), akin to the system in the paper by Croft and Das (1990). In our experiments, individual features are presented to the user for selection. Section 6.3 provides the details of our method.

In our proposed system the teacher is asked two types of questions: (1) membership queries and (2) questions about the relevance of features. A relevant feature is highly likely to help discriminate the positive class from the negative class. In this paper we aim to determine whether a human teacher can answer the latter type of question sufficiently effectively so that active learning is accelerated significantly. A human and a classifier probably use very different processes to categorize instances. A human may use her understanding of the sentences within the document, which probably involves some reasoning and use of knowledge, in order to make the categorization decision, while a (statistical) classifier, certainly of the kind that we use in this paper, simply uses patterns of occurrences of the features (phrases). Therefore, it is not clear whether a human teacher can considerably accelerate the training of a statistical classifier, beyond simple active learning, by providing feedback on features.

Before we address that issue, we determine whether feature feedback can accelerate active learning in an idealized setting. We seek to get a sense of the room for improvement. We will then examine how actual human teachers can approximate this ideal. Towards this goal we define a (feature) *oracle*. We use the oracle to obtain an upper bound on the performance of our proposed two-tiered approach. The oracle knows the correct answer needed by the learning algorithm. For example the word *ct* is a highly relevant feature for classifying Reuters news articles on the *earnings* category and our oracle would be able to determine that this feature is relevant when asked. However, a teacher (human) who did not understand that *ct* stood for *cents* may not be able to identify *ct* as relevant (we will see this exact example in Section 6.1). Therefore, the oracle and teacher may differ in their answers to questions about features, that is, questions of type (2) above. We assume that the oracle and the teacher always agree on the labels of documents that is, questions of type (1) above. After showing the usefulness of oracle feature selection, we will then show that humans can emulate the oracle for feature feedback to an extent that results in significant improvements over traditional active learning.

2.2 Extent of Speed Up Possible: Oracle Experiments

We perform two types of experiments with the oracle. In the first kind, the oracle, knowing the allotted time T, picks the best subset of features to improve, as much as possible, the performance of active learning. The procedure is shown in Figure 3. In Figure 3, the *Incorporate_Feature_Feedback* subroutine is called to initialize the model. When System 3 is used with a user instead of the oracle it is equivalent to a scenario where prior knowledge is used to initialize the classifier (Schapire et al., 2002; Wu and Srihari, 2004; Godbole et al., 2004; Jones, 2005). In Section 3.4 we describe how this oracle is *approximated* in our experiments.

Use of Feature Feedback Before Active Learning

Input: T (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances), *init_size* (number of random feedback iterations)
Output: $\mathcal{M}_T$ (Model)

$t = 1$; $\mathcal{U}_0 = \mathcal{U}$; $\mathcal{M}_0$ =NULL;
1.a. $\{F1,...,F_f\}$ = Feature_Selection($\mathcal{U}_0$)
  b. Oracle selects $\{F1,..,F_k\} \subseteq \{F_1,...,F_f\}$
2.Incorporate_Feature_Feedback($\mathcal{M}_0$, $\{F_1,...,F_k\}$)
3. While $t \leq$ init_size
  a. $\langle X_t, \mathcal{U}_t \rangle$=Instance_Selection($\mathcal{M}_{t-1}$, $\mathcal{U}_{t-1}, random$)
  b. Oracle assigns label Y_t to X_t
  c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
  d. $t++$
4. While $t \leq T$
  a. $\langle X_t, \mathcal{U}_t \rangle$=Instance_Selection($\mathcal{M}_{t-1}$, $\mathcal{U}_{t-1}, uncertain$)
  b. Oracle assigns label Y_t to X_t
  c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
  d. $t++$
Return $\mathcal{M}_T$

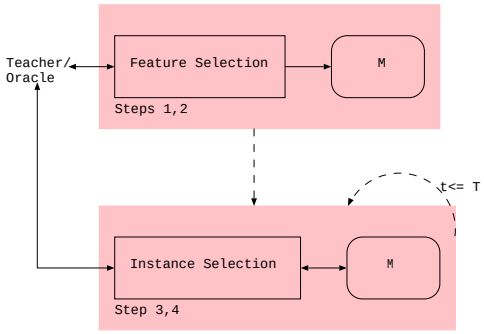


Figure 3: An active learning system where feature selection is done before instance selection (**System 3**). This is one of the two set-ups used in our oracle experiments described in Section 2.2. The second set-up is shown in Figure 4.

The second type of experiment is a slight variation designed to isolate the effect of oracle feature selection on example selection versus model selection during active learning. In these experiments, active learning proceeds normally with all the features available, but after all the instances are picked (after T iterations), the best set of k features that improve the resulting trained classifier the most are picked and the resulting performance is reported. This is shown schematically and with pseudo-code in Figure 4. We note that even when starting with the same initial set of labeled instances, the classifiers learned during active learning, hyperplanes in our case, in these two systems may be different as they are learned in different spaces (using different feature subset sizes). Besides, the set of labeled instances is small, so the learning algorithm may not be able to find the best "unique" hyperplane. In turn, the instances picked subsequently during active learning may differ

substantially as both the spaces the instances reside in and the learned classifiers may be different. The classifier learned in the feature reduced space may have better accuracy or lead to better choice of instances to label during active learning, though this is not guaranteed or the benefits may be negligible. In short, the trajectory of the active learning process, that is, the instances labeled and classifiers learned, can be different in the two regimes, which may lead to substantially different active learning performance. In Section 5 we provide the details of these experiments.

Systems 3 and 4 can also be used with a teacher (a human) instead of an oracle. For an actual use in practice, we prefer an approach that combines feature selection and instance selection (*e.g.*, as proposed in Section 2.1) because it also allows the system to benefit from the increase in the knowledge of the teacher or the process may help remind the teacher about the usefulness of features as she reads the documents. For example, the teacher who did not know that *ct* stood for *cents* may realize that the word is indeed relevant upon reading documents containing the term. We will discuss these related approaches in Section 7.

Use of Feature Feedback After Active Learning

Input: T (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances, init_size (number of random feedback iterations)
Output: $\mathcal{M}_T$ (Model)

$t = 1$; $\mathcal{U}_0 = \mathcal{U}$; $\mathcal{M}_0$ =NULL;
1. While $t \leq$ init_size
   a. X_t = Instance_Selection($\mathcal{M}_0$, $\mathcal{U}_{t-1}$, *random*)
   b. Oracle assigns label Y_t to X_t
   c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i\rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
   c. $t++$
2. While $t \leq T$
   a. $\langle X_t, \mathcal{U}_t\rangle$ = Instance_Selection($\mathcal{M}_{t-1}$, $\mathcal{U}_{t-1}$, *instance*)
   b. Oracle assigns label Y_t to X_t
   c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i\rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
   d. $t++$
3. a. $\{F1, ..., F_f\}$ = Feature_Selection($\mathcal{M}_T$, $\mathcal{U}_T$)
   b. Oracle selects $\{F1, .., F_k\} \subseteq \{F_1, ..., F_f\}$
4. Incorporate_Feature_Feedback($\mathcal{M}_T$, $\{F_1, ..., F_k\}$)

Return $\mathcal{M}_T$

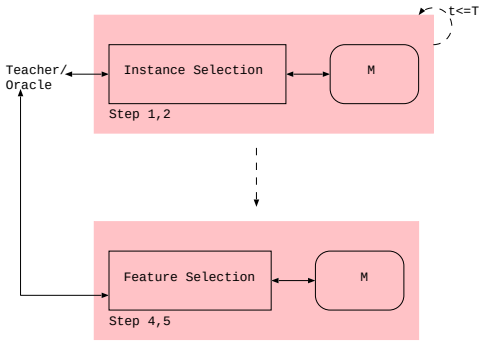


Figure 4: An active learning system where feature selection is done after instance selection (**System 4**). This is one of the two set-ups used in our oracle experiments described in Section 2.2. The first set-up is shown in Figure 3.

3. Implementation

In this section we give implementation details for our experiments. While our approach is applicable to a variety of machine learning algorithms and feature selection approaches, we give the details of our implementation. We use Support Vector Machines (SVMs) as the machine learned classifier, uncertainty sampling as our approach to active learning and information gain as the feature selection technique. We also give details on how we construct the approximate feature oracle.

3.1 Classifier: Support Vector Machines

We use support vector machines (SVMs) in our experiments (the model $\mathcal{M}$ is a Support Vector Machine (SVM)) (Joachims, 1998). An SVM learning algorithm tries to find a hyperplane of maximum margin that separates the data into one of two classes ($Y \in \{-1,+1\}$). A linear SVM is a binary classifier given as

$$f(X) = sign(w \bullet X + b), \tag{1}$$

where w is the vector of weights and b is a threshold, both learned by the SVM learning algorithm.

SVMs are considered to be state-of-the-art classifiers in the domains that we described in Section 4.1 and have been found to be fairly robust even in the presence of many redundant and irrelevant features (Brank et al., 2002; Rose et al., 2002.). Our SVM implementation uses the LibSVM toolkit (Chang and Lin).

3.2 Active Learning: Uncertainty Sampling

Uncertainty sampling (Lewis and Catlett, 1994) is a type of active learning in which the instance that the teacher is queried on is the unlabeled instance that the classifier is most uncertain about. In the case of a naive Bayes classifier, this is the instance which is almost equally likely to be in either of the two classes in a binary classification setting. When the classifier is an SVM, unlabeled instances closest to the margin are chosen as queries (Schohn and Cohn, 2000; Tong and Koller, 2002). This results in the version space being split approximately in half each time an instance is queried. We use a pool size of 500 in our experiments, such that for each instance selection, we look at a new random sample of 500 instances from the unlabeled data. All our methods start out with two randomly picked instances, one in the positive class and one in the negative class. Each subsequent instance is picked through uncertainty sampling.

3.3 Feature Selection: Information Gain

We could have chosen any one of several methods for the ordering of features (Sebastiani, 2002; Brank et al., 2002). Information gain is a common measure for ranking features and has been found to be quite effective (Sebastiani, 2002; Brank et al., 2002), and is easy and quick to compute.

Information gain is given as

$$IG = \sum_{c \in \{-1,+1\}} \sum_{\tau \in \{0,1\}} P(c,\tau) \log \frac{P(c,\tau)}{P(c)P(\tau)} \tag{2}$$

where c denotes the class label (+1 or -1) from section 3.1, and τ is 0 or 1 indicating the presence or absence of a feature respectively. We used information gain wherever we needed to do feature selection.

3.4 Construction of the Approximate Feature Oracle

The (feature) oracle in our experiments has access to the labels of all documents in the data-set (hence the name oracle) and uses this information to return a ranked list of features sorted in decreasing order of importance. We use information gain for feature ranking since it is easy to compute, especially with a large number of training instances. Other feature selection methods (*e.g.*, forward selection) may somewhat increase our upper bound estimates of usefulness of oracle feature feedback. Such improvements will further motivate the idea of using feature feedback, but we don't expect the improvements to be very high. In our oracle experiments, we cut off the ranked list (therefore obtaining a feature subset) at the point that yields the highest average active learning performance. The next section describes our experiments and performance measures.

4. Experimental Set Up

We will now describe our data sets and our data collection methodology for experiments which use teacher feedback on features.[2] We then describe our evaluation framework.

4.1 Data Sets

Our test bed for this paper comes from three domains. The first data set consists of the 10 most frequent classes from the Reuters-21578 corpus (Rose et al., 2002.). The 12,902 documents are Reuters news articles categorized based on topics such as *earnings* and *acquisitions*. The Reuters corpus is a standard benchmark for text categorization. The second corpus is the 20-Newsgroups data set collected by Lang (1995). It has 20,000 documents which are postings on 20 Usenet newsgroups. This is a slightly harder problem because it has a large vocabulary compared to the Reuters corpus (news articles tend to be more formal and terse) and it has many documents in each category which are tangentially related to the topic. The topics reside in a hierarchy with broader topics like *sports* and *computers* at the top level which are further divided into narrower subdivisions. For example, *sports* encompasses more focused groups like *baseball* and *hockey*. There are 20 categories at the lowest level of the hierarchy.

The third corpus is the TDT3 corpus (Allan, 2002) . We used 10 topics from the TDT3 corpus which has 67,111 documents in three languages from both broadcast and news-wire sources. The Linguistic Data Consortium (LDC) provides the output of an automatic speech recognizer (ASR) for the broadcast news sources. Similarly they provide the machine translations of all documents that are not originally in English. We use the ASR and machine translated documents in our experiments. The noise in the ASR and machine translation output makes the TDT corpus particularly difficult to work with. The topics in the TDT corpus are based on news events. Thus, hurricane Mitch and hurricane George would be two different topics and developing a classifier to separate the two classes is seemingly a more difficult problem. The two classes would have a lot of common words especially with regard to lives lost, rescue operations etc. For example, the words *storm* and *damage* each respectively occur in 50% and 27% of the documents on hurricane Mitch and in 75% and 54% of the documents on hurricane George. These common words are probably useful to detect a generic topic like *hurricane* but are not that useful in discriminating hurricane Mitch from hurricane George. However, we think it would be fairly trivial for a human to point out *Mitch* and *George* as two keywords of importance which could then accelerate learning. The word Mitch occurs in

2. The data sets have been made available at `http://ciir.cs.umass.edu/~hema/data/jmlr2006/`.

42% documents on hurricane Mitch and in 0 documents on hurricane George. Similarly, the word George appears in 0.05% documents on the topic of hurricane Mitch and in 88% of the documents on hurricane George.

For all three corpora we consider each topic as a one-versus-rest classification problem, giving us a total of 40 such problems listed in Appendix A. We also pick two pairs of easily confusable classes from the 20-Newsgroups domain to obtain two binary classification problems viz., *baseball vs hockey* and *automobiles vs motorcycles*. In all we have 42 classification problems. As features we use words, bi-grams and trigrams obtained after stopping and stemming with the Porter stemmer (Porter, 1980) in the Rainbow toolkit (McCallum, 1996).

4.2 Data for Whether Humans Can Emulate the Oracle

We picked five classification problems which we thought were perceptible to a non-expert and also represented the broad spectrum of problems from our set of 42 classification problems. We took the two binary classification problems and from the remaining 40 one-versus-rest problems we chose three (*earnings*, *hurricane Mitch* and *talk.politics.mideast*). For a given classification problem we took the top 20 features as ranked by information gain on the entire labeled set. We randomly mixed these with features which are much lower in the ranked list. We showed each user one feature at a time and gave them two options – *relevant* and *not-relevant/don't know*. A feature is relevant if it helps discriminate the positive or the negative class. We measured the time it took the user to label each feature. We did not show the user all the features as a list, though this may be easier, as lists provide some context and serve as a summary. Hence we expect that our method provides an upper bound on the time it takes a user to judge a feature. The instructions given to the annotator are given in Appendix B.

Similarly, we obtain judgments on fifteen documents in each of five categories (see Appendix C). In this case we gave the user three choices – Class 1, Class 2, Don't know. We randomly sampled documents such that at least five documents belonged to each class. We have complete judgments on all the documents for all three data sets. The main purpose of obtaining document judgments was to determine how much time it would take a person to judge documents. We compare the time it takes a user to judge a feature with the time it takes a user to judge a document. We measure the precision and recall of the user's ability to label features. We ask the user to first label the features and then documents, so that the feature labeling process receives no benefit due to the fact that the user has viewed relevant documents. In the learning process we have proposed, though, the user would be labeling documents and features simultaneously, so the user would indeed be influenced by the documents she reads. Hence, we expect that the feature labels we obtained by our experimental method are worse in terms of precision and recall than the real setting. We could in practice ask users to highlight terms as they read documents. Experiments in this direction have been conducted in information retrieval (Croft and Das, 1990).

Our users (participants) were six graduate students and two employees of an Information Technology company, none of whom were authors of this paper. Of the graduate students, five were in computer science and one from public health. All our users were familiar with the use of computers. Five users understood the problem of document classification but none had worked with these corpora. One of our users was not a native speaker of English. The topics were distributed randomly, and without considering user expertise, so that each user got an average of two to three topics. There were overlapping topics between users such that each topic was labeled by two to

three users on average. A feedback form asking the users some questions about the difficulty of the task was handed out at the end (see Appendix D).

4.3 Evaluation

The *deficiency* measure was proposed by Baram et al. (2003) as a measure of the speed of an active learning algorithm, useful for comparing different active learning algorithms. Baram et al. defined deficiency in terms of accuracy. Accuracy is a reasonable measure of performance when the positive class is a sizable portion of the total. Since this is not the case for all the classification problems we have chosen, we modify the definition of deficiency, and define it in terms of the $F1$ score (harmonic mean of precision and recall). For deficiency a lower value is better. As we also report on the $F1$ scores, for which higher values are better, for consistency and easier interpretation of our charts and tables we define *efficiency* $= 1 -$ *deficiency*. Efficiency has a range from 0 to 1, and a larger value indicates a faster rate of learning. Thus, in all our reports higher values are better.

Let $F1_t(\text{RAND})$ be the average $F1$ achieved by an algorithm when it is trained on t randomly picked instances and $F1_t(\text{ACT})$ be the average $F1$ obtained using t actively picked instances.

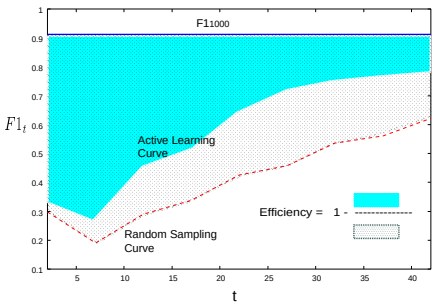
Efficiency, E_T is defined as

$$E_T = 1 - \frac{\sum_{t=2}^{T}(F1_M(\text{RAND}) - F1_t(\text{ACT}))}{\sum_{t=2}^{T}(F1_M(\text{RAND}) - F1_t(\text{RAND}))}. \quad (3)$$

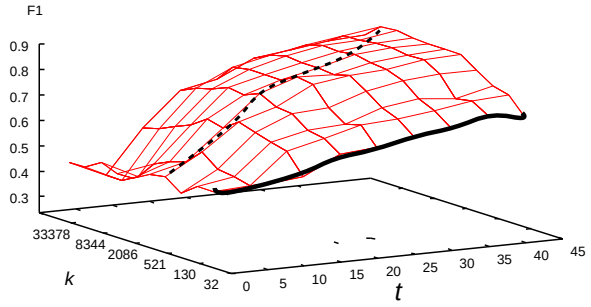
$F1_M(\text{RAND})$ is the $F1$ obtained with a large number (M) of randomly picked instances. The value $F1_M(\text{RAND})$ represents the performance of a classifier with a large amount of training data, and can be considered the optimal performance under supervised learning. With large amounts of training data, we expect the performance of a classifier trained using active learning to be about the same as a classifier trained using random sampling. However, we would like active learning to approach this level as *quickly* as possible. The metric therefore takes into consideration how far the performance is from the optimal performance by computing the difference $F1_M(\text{RAND}) - F1_t(\text{ACT})$ and $F1_M(\text{RAND}) - F1_t(\text{RAND})$. The metric compares this difference when t documents have been actively picked to the difference when t documents have been randomly picked for increasing number of training documents t.

Since we are concerned with the beginning of the learning curve, we stop after $T = 42$ number of documents have been sampled. For expedience, we did not measure performance at every point from 2 to 42 labeled documents, but compute the summation at discrete intervals, measuring $F1$ after each additional five documents have been labeled: $t = 2, 7, 12, 17 \ldots 42$. For this paper we take $M = 1000$, that is, we consider the optimal random-learning performance to be attained after the classifier has seen 1000 labeled instances. In our experiments $F1_t(\bullet)$ is the average $F1$ computed over 10 trials. In addition to efficiency we report $F1_t$ for some values of t.

To understand the intuition behind efficiency, we can draw the active learning curve by plotting $F1_t(\text{ACT})$ for increasing values of t, as shown in Figure 5(a). Similarly we can draw the random learning curve by measuring $F1_t(\text{RAND})$ for increasing values of t. $F1_M$ is a straight line representing the best achievable performance. Then efficiency is one minus the ratio of the solid colored area to the spotted area. The higher the efficiency, the better the active learning algorithm. We aim to maximize both efficiency and $F1$. In some of our experiments we obtain efficiencies exceeding 1. This is due to using a finite M: it is possible that a classifier produced by active learning on 42 or fewer instances may do better than a classifier trained on a random sample of a 1000 instances.



(a) Efficiency



(b) The best feature set size increases as the training data increases.

Figure 5: The figure on the left (a) illustrates *efficiency*, the performance metric which captures rate of learning. The figure on the right illustrates the *learning surface*. The plot is a measure of $F1$ as a function of the number of features and training documents. The dotted line traces the region of maximum $F1$. With few training documents, aggressive feature selection (few features) are needed to maintain high accuracy. The thick dark band illustrates traditional active learning.

5. Results: Experiments with an Oracle

In this section we seek the answer to the following questions:

- Can feature feedback significantly boost active learning performance?
- Should we use feature feedback during the entire active learning process (both instance selection, and model selection) or only for model selection?

To measure how much gain we can get from feature feedback we can measure the impact of the oracle (which has knowledge of the best set of features) on active learning. This gives us an upper bound on how useful feature feedback is for active learning. Then in the next section we go on to measure the extent to which humans can emulate the oracle.

We will use systems 3 and 4 (described in Section 2.2) to help understand the answers to the above questions.

5.1 Improvements to Active Learning with Feature Selection

Following the algorithm for system 3 (see Section 2.2, Figure 3), let $f = N$ (the total number of features) and let us assume that the oracle selects the k most important features (by information gain) in step 1.b, which is used to initialize the model in step 2. Random sampling (step 3.a), in this particular implementation, does not use any of the feature information or the initial model. Then in step 3.c, we prune the data set by retaining only the chosen k features for each instance. We now perform active learning on the instances in this reduced feature space (step 4). We evaluate these experiments at many points in the two-dimensional space of number of features k versus number of labeled documents t by measuring the F1 score: $F1_t(\mathrm{ACT}, k)$. We can similarly measure

Data Set ↓	$E_{42}(k)$ $k=N$	$k=n$	n	$F1_7(\text{ACT},k)$ $k=N$	$k=m$	m	$F1_{22}(\text{ACT},k)$ $k=N$	$k=p$	p	$F1_{1000}$
Reuters	0.59	**0.68**	11179.3	0.36	**0.48**	8481.1	0.580	**0.66**	11851.6	0.73
20 NG	0.40	**0.66**	41.5	0.07	**0.22**	48.3	0.21	**0.29**	487.1	0.45
TDT	0.26	**0.34**	1275.7	0.19	**0.29**	11288	0.28	**0.41**	10416.1	0.75
Bas vs Hock	0.29	**0.55**	25	0.59	**0.70**	25	0.78	**0.83**	200	0.96
Auto vs Mot.	0.68	**0.32**	125	0.43	**0.72**	62	0.76	**0.86**	31	0.90

Table 1: Improvements in efficiency, $F1_7$ and $F1_{22}$ using an oracle to select the most important features (Figure 3). We show results for each metric at N (total number of features for a particular data set) and at feature set sizes for which the scores are maximized (n, m and p for E_{42}, F_7, and F_{22} respectively). For each of the three metrics, figures in bold are statistically significant improvements over uncertainty sampling using all features (the corresponding columns with feature set size of N). We see that with only seven documents labeled ($F1_7$) the optimal number of features is smaller (8481.1 on average), while with more documents labeled, (22 documents labeled for $F1_{22}$) the optimal number of features is larger (11851.6 on average). When 1000 documents are labeled ($F1_{1000}$) using the entire feature set leads to better scores with the $F1$ measure. This suggests that our best active-learning algorithm would adjust the feature set size according to the number of training documents available.

performance in the reduced feature space when instances are picked randomly. Thus we can compute efficiency in the reduced feature space as $E_T(k)$. When $f = k = N$ the algorithm reduces to traditional active learning (Figure 1).

Figure 5(b) shows a plot of $F1_t(\text{ACT},k)$ for different values of the number of features k and number of labeled training instances t, for the *earnings* category in Reuters. The dotted curve traces the maximum F_t for each value of t. The x, y and z axes denote k, t and $F1_t(\text{ACT},k)$ respectively. The number of labeled training instances t ranges from 2 to 42 in increments of 5. The number of features used for classification k has values from $33,378$ (all features), $33378/2$, $33378/4$ to 32. The dark band represents the case when all features are used. This method of learning in one dimension is representative of traditional active learning. Clearly when the number of documents is few, performance is better when there is a smaller number of features. As the number of documents increases the number of features needed to maintain high accuracy increases. From the figure it is obvious that we can get a big boost in accuracy by starting with fewer features and then increasing the complexity of the model as the number of labeled documents increase.

Table 1 captures the behavior of all the problems in the Reuters corpus when there is an oracle to do the feature selection. The second column ($k = N$) in Table 1 shows the efficiency obtained using uncertainty sampling and all (N) features. The third column ($k = n$) indicates the average efficiency obtained using uncertainty sampling and a reduced subset of features. The feature set size n at which this efficiency is attained is shown in column four. For each classification problem, we identify the feature set size which optimizes the efficiency, that is, optimizes the rate at which classification performance under active learning approaches learning with all of the data. This optimal feature set size for active learning n is given by

$$n = \text{argmax}_k E_{42}(k).$$

Figure 6 shows the efficiencies at $E_{42}(N)$ and $E_{42}(n)$ for the individual problems in the three corpora. In many cases, $E_{42}(N)$ is much less than $E_{42}(n)$.

Column 5 ($k = N$) in Table 1 shows the value of $F1_7(\text{ACT}, N)$: the F1 score with seven instances selected using active learning, when all features are used. Column 6 shows the average $F1_7(\text{ACT}, m)$ using a reduced feature subset. As for efficiency the best feature subset size (m) for each classification problem is obtained as the feature subset size at which $F1_7(\text{ACT}, k)$ is maximum. For example in Figure 5(b) at seven instances the best $F1$ is obtained with 512 features. Figure 7 shows the values of $F1_7$ computed using all (N) features and using a reduced subset of (m) features for individual problems.

Columns 7, 8, and 9 in Table 1 show similar results for $F1_{22}(\text{ACT}, k)$ with the best feature subset size at $t = 22$ being denoted by p. The values for individual problems is illustrated in Figure 8. The last column shows $F1_{1000}(\text{RAND})$.

All 42 of our classification problems exhibit behavior as in Figure 5(b). For all classification problems, n, m and p are less than the maximum number of features. Also, for 31 of 42 cases $m \leq p$ (that is, the number of features optimal for seven labeled instances, m is less than the number of features optimal for 22 labeled instances, p) meaning that as the number of labeled instances (t) increases, the complexity of the classifier also needs to increase. For 20-Newsgroups, for all classes we observe that efficiency, $F1_7$ and $F1_{22}$ are best at very small feature subset sizes. For Reuters and TDT there are classes for which a large number of features become important very early (for example: *trade, Bin Laden indictment, NBA labor disputes*).

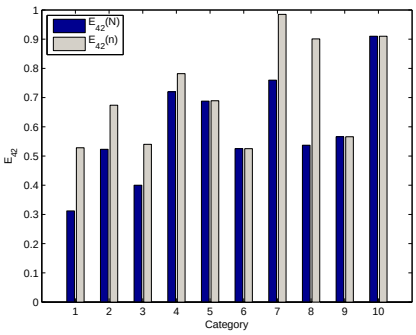
5.2 Feature Selection for Instance Selection or Model Selection

As mentioned in Section 2.2 the difference between systems 3 and 4 is in that feature selection precedes active learning in the former, and the best feature subset is picked in a retrospective manner, while it follows active learning in the latter. The two systems when used with oracle feature selection will help us understand the extent to which oracle feedback aids different aspects of the active learning process. Figure 9 compares the results of using system 4 and system 3 on the Reuters corpus.

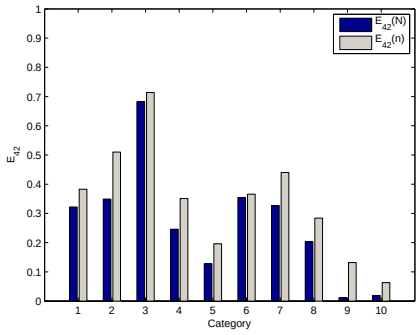
There is hardly any difference between systems 3 and 4, especially on $F1_7$. All other data sets exhibit the same behavior. The $F1_{22}$ and E_{42} values are slightly better for the method that does feature selection before active learning (system 3) but it is not significantly different (determined using a t-test at the 0.05 level of confidence) from the method where feature pruning is done after instance selection (system 4). Thus, our experimental results suggest there is some benefit for instance selection but most of the benefit from oracle feature selection comes from improving the model learned (model selection).

5.3 Discussion: Why Does Feature Selection Help?

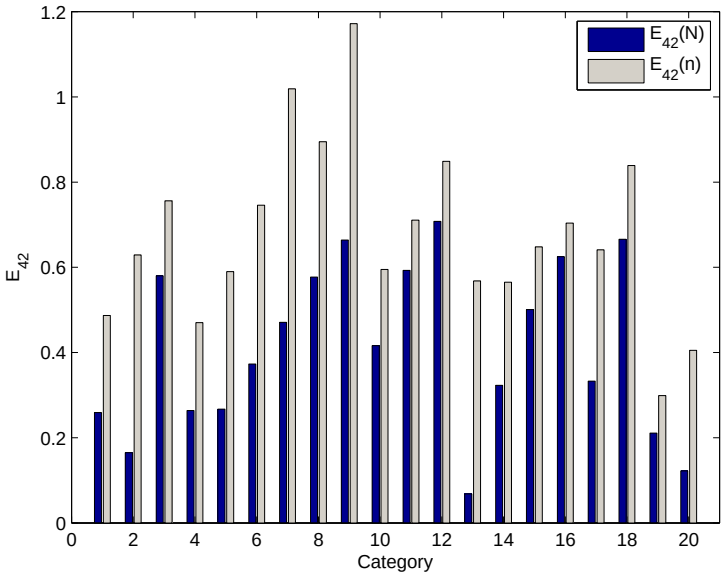
Intuitively, with limited labeled data, there is little evidence to prefer one feature over another, so the learner has to spread the feature weights more or less evenly on many features. In other words, the learner has to remain conservative. Feature/dimension reduction by the oracle allows the learner to "focus" on dimensions that matter, rather than being overwhelmed with numerous dimensions



(a) Reuters

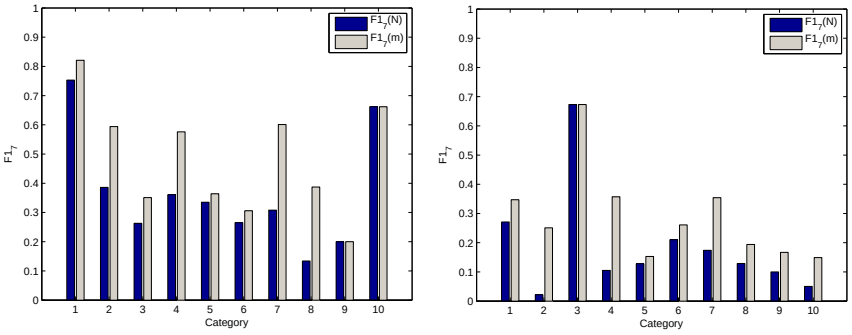


(b) TDT

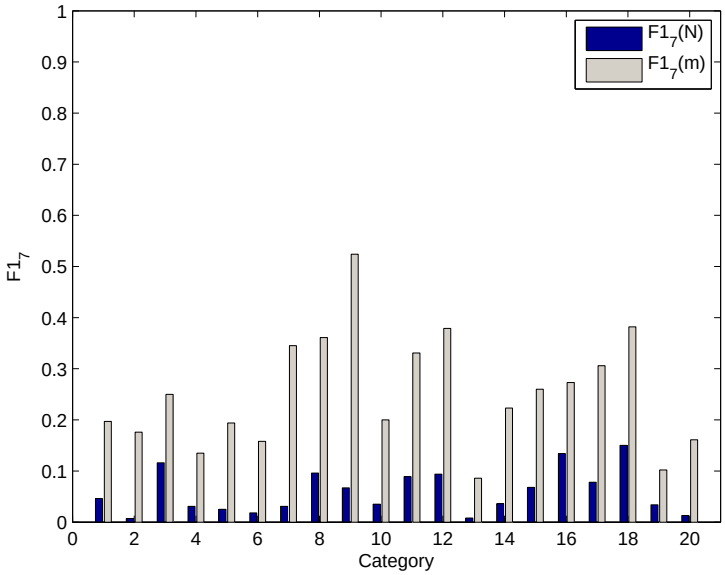


(c) 20 Newsgroups

Figure 6: Improvements in efficiency using an oracle to select the most important features. For each problem we show efficiency at N (total number of features for a particular data set) on the right and efficiency at the feature set sizes for which the efficiency is maximized (n) on the left. The class keys are given in Appendix A.

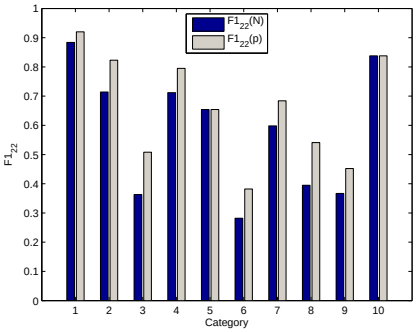


(a) Reuters (b) TDT

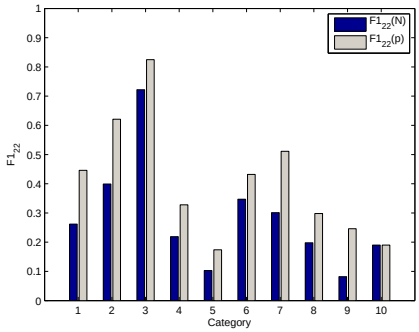


(c) 20 Newsgroups

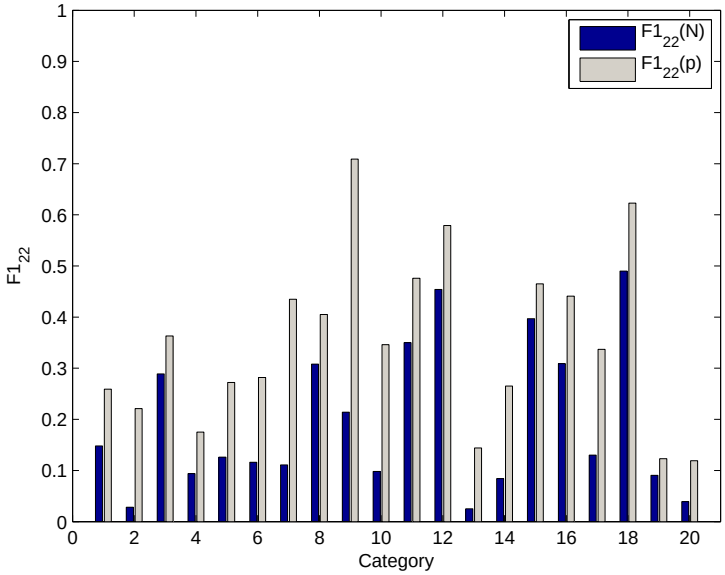
Figure 7: Improvements in $F1_7$ using an oracle to select the most important features. For each problem we show $F1_7$ at N (total number of features for a particular data set) on the left and $F1_7$ at the feature set sizes for which the $F1_7$ is maximized (m) on the right. Remember, the objective is to maximize $F1_7$. The class keys are given in Appendix A.



(a) Reuters

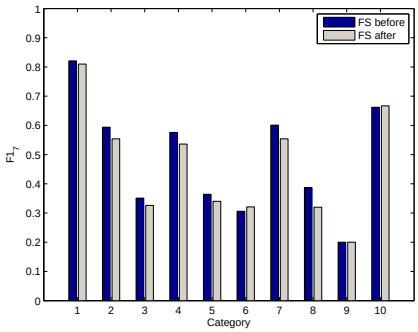


(b) TDT

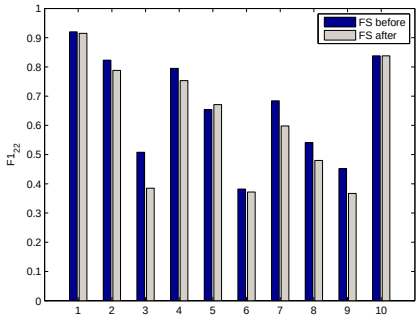


(c) 20 Newsgroups

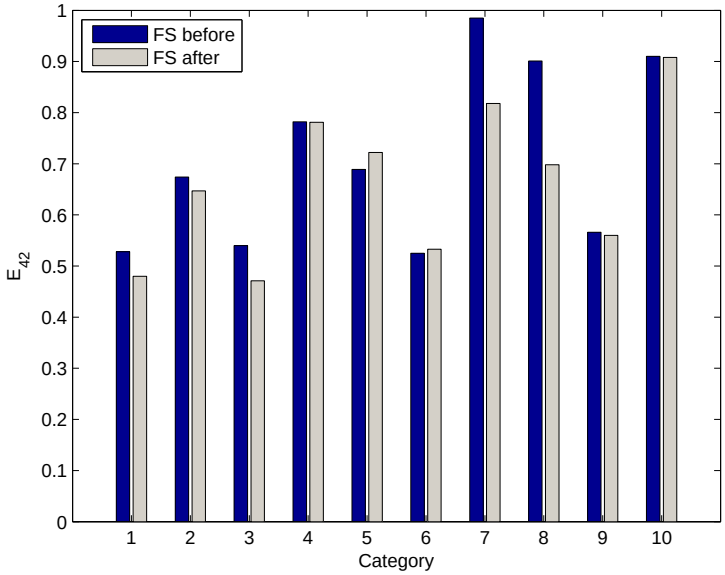
Figure 8: Improvements in $F1_{22}$ using an oracle to select the most important features. For each problem we show $F1_{22}$ at N (total number of features for a particular data set) on the right and $F1_{22}$ at the feature set sizes for which the $F1_{22}$ is maximized (p). Remember that the objective is to maximize $F1_{22}$. The class keys are given in Appendix A.



(a) $F1_7$



(b) $F1_{22}$



(c) E_{42}

Figure 9: $F1_7$, $F1_{22}$ and efficiency E_{42} for the Reuters corpus when feature selection is done before active learning (system 3) and when feature selection is done after active learning (system 4).

right at the outset of learning. Oracle feature reduction allows the learner to assign higher weights to fewer features. This tends to improve accuracy, since the oracle selected features are the actual most predictive features. Oracle feature reduction may also improve instance selection as the learner obtains instances to query that are important for finding better weights on the features that matter. As the number of labeled instances increases, feature selection becomes less important, as the learning algorithm becomes better capable of finding the discriminating hyperplane (feature weights) on its own. We experimented with filter based methods for feature selection, which did not work very well (we got tiny or no improvements). This is expected given such limited training set sizes, and is consistent with most previous findings (Sebastiani, 2002). Next we determine if humans can identify these *important features*.

6. Results: Experiments with a Human (Teacher)

Consider our introductory example of the editor who was looking for training instances for the topic *hurricane Mitch*. From a human perspective the words *hurricane, Mitch* etc may be important features in documents discussing this topic. Given a large number of documents labeled as on-topic and off-topic, and given a classifier trained on these documents, the classifier may also find these features to be most relevant. With little labeled data (say two labeled instances) the classifier may not be able to determine the discriminating features. While in general in machine learning the source of labels is not important to us, in active learning scenarios in which we expect the labels to come from humans we have valid questions to pose:

1. Can humans label features as well as documents? In other words are features that are important to the classifier perceptible to a human?

2. If the feature labels people provide are imperfect, is the feedback still beneficial to active learning?

We address the first question in the following section. Our concern in this paper is asking people to give feedback on features, or word n-grams, as well as entire documents. We may expect this to be more efficient, since documents are often long and may contain redundant or irrelevant content, and results from our oracle experiments indicate great potential in doing feature selection. We then move on to discuss a real system which employs a two-tiered approach of document feedback and feature feedback like the system in Figure 2 which we evaluate using a simulation: we obtain feedback on features and documents apriori, and use the judgments so obtained to measure the effectiveness of our approach. We employed this approach rather than one where an actual user labels features and documents in tandem because our approach allows us to run many repeated trials of our experiments, enabling us to do significance testing. Given that we have demonstrated the effectiveness of our algorithm, we reserve a more realistic evaluation with a true human in the loop for future work.

6.1 Can Humans Emulate the Oracle?

We evaluated user feature labeling by calculating their average precision and recall at identifying the top 20 features as ranked by an oracle using information gain on the entire labeled set. Table 2 shows these results. For comparison we have also provided the precision and recall (against the

Class	Precision		Recall		Avg. Time (secs)		kappa
Problem	Hum.	@50	Hum.	@50	Feat.	Docs	
baseball vs hockey	0.42	0.30	0.70	0.30	2.83	12.60	0.503
auto vs motorcycle	0.54	0.25	0.81	0.25	3.56	19.84	0.741
earnings	0.53	0.20	0.66	0.25	2.97	13.00	0.495
talk.politics.mideast	0.68	0.35	0.55	0.35	2.38	12.93	0.801
hurricane Mitch	0.72	0.65	0.56	0.65	2.38	13.19	0.857
Average	0.580	0.35	0.65	0.38	2.82	14.31	0.68

Table 2: Ability of users to identify important features. Precision and Recall against an oracle, of users (Hum.) and an active learner which has seen 50 documents (@50). Note that precision and recall denote the ability of the user to recognize the oracle features and are not measures of classification accuracy. Average labeling times for features and documents are also shown. All numbers are averaged over users.

same oracle ranking of top 20 features) obtained using 50 labeled instances (picked using uncertainty sampling) denoted by @50. Precision and recall of our participants is high, supporting our hypothesis that features that a classifier finds to be relevant after seeing a large number of labeled instances are obvious to a human after seeing little or no labeled data (the latter case being true of our experiments). Additionally the precision and recall @50 is significantly lower than that of humans, indicating that a classifier like an SVM needs to see much more data before it can find discriminatory features.

Table 2 also shows the times taken for labeling features and documents. On average humans take five times longer to label one document than to label one feature. Note that features may be even easier to label if they are shown in context – as lists, with relevant passages etc. We measured whether document length influences document labeling time. We found the two to be correlated by $r = 0.289$ which indicates a small increase in time for a large increase in length. The standard deviations for precision and recall are 0.14 and 0.15 respectively. Different users vary significantly in precision, recall and the total number of features labeled relevant. From the post-labeling survey we are inclined to believe that this is due to individual caution exercised during the labeling process.

We also measure the extent to which our users tend to agree with each other about the importance of features. For this we use the kappa statistic (Cohen, 1960) which is a measure that quantifies the agreement between annotators that independently classify a set of entities (in our case the features) into classes (relevant versus non-relevant/don't know). Kappa is given by:

$$\text{kappa} = (p_o - p_c)/(1 - p_c) \tag{4}$$

Where p_o is the observed proportion of agreement and p_c is the agreement due to chance (Cohen, 1960; Landis and Koch, 1977). Landis and Koch (1977) provide a table giving guidelines about how to interpret kappa values. We find a value of 0.68 to be the average kappa across the five categories in our user study. According to Landis and Koch (1977) this indicates substantial agreement.

We obtained judgments on a handful of documents for each user. We used those judgments to measure time. Some of our users had difficulty judging documents. For example, for the earnings category, one of our users had very low agreement with the true Reuters categories. This person did

not have a finance background and could not distinguish well between earnings and acquisitions, often confusing the two. But this user did quite a good job of identifying useful features. She missed only six of 20 of the relevant features and had only five false alarms. The features that she marked relevant, when used in the human-in-the-loop algorithm resulted in an efficiency of 0.29. This is still an improvement over traditional uncertainty sampling which has a efficiency of 0.10. These results can be explained by looking at the question posed to the annotator. When it came to features, the question was on the discriminative power of the feature. Hence a user did not have to determine whether the words *shares* was pertinent to *earnings* or not but rather she only needed to indicate whether the word was likely to be discriminatory. Additionally, one of our users suggested that terms shown in context would have carried more meaning. The user said that she did not realize the term *ct* stood for *cents* until she read the documents. But since she was made to judge terms before documents this user's judgment had marked the term *ct* as non-relevant/don't know.

Some of the highlights of the post-labeling survey are as follows. On average users found the ease of labeling features to be 3.8 (where 0 is most difficult and 5 is very easy) and documents 4.2. In general users with poor prior knowledge found the feature labeling process very hard. The average expertise (5=expert) was 2.4, indicating that most users felt they had little domain knowledge for the tasks they were assigned. We now proceed to see how to use features labeled as relevant by our naive users in active learning.

6.2 Using Human Feature Feedback simultaneously with Document Feedback in Active Learning

We saw in Section 5 that feature selection coupled with uncertainty sampling gives us big gains in performance when there are few labeled instances. In Section 6.1 we saw that humans can discern discriminative features with reasonable accuracy. We now describe our approach of applying term and document level feedback simultaneously in active learning. In Section 2.2 we discussed the possible cognitive advantages of an interleaved approach of feature selection and instance selection. Additionally, we found that feature selection does not hurt uncertainty sampling and may aid it. In the following section we describe an implementation for system 2.

6.3 Implementation

Following Figure 2, the features to be displayed to the user (in step 2.d.i) are the top f features obtained by ordering the features by information gain. More specifically, we trained the SVM classifier on these t labeled instances. Then to compute information gain, we used the five top ranked (farthest from the margin on the positive side) documents from the unlabeled set in addition to the t labeled documents. Using the unlabeled data for term level feedback is very common in information retrieval and is called pseudo-relevance feedback (Salton, 1968). The user labels $k \geq 0$ of the f features as relevant or discriminative (step 2.d.ii). If a user has labeled a feature in a previous iteration, we don't show the user that feature again (the top f are picked from the unlabeled features). We set f to 10 in our experiments.

We incorporate feature feedback (step 2.e) as follows. Let $\vec{s} = s_1 \ldots s_N$ be a vector containing weights of relevant features. If a feature number i that is presented to the user is labeled as relevant then we set $s_i = a$, otherwise $s_i = b$, where a and b are parameters of the system. For each X in the

labeled and unlabeled sets we multiply x_i by s_i to get x'_i. In other words, we scale all the features that the user indicated as relevant by a and the rest of the features by b. We set $a = 10$ and $b = 1$. [3]

By scaling the important features by a we are forcing the classifier to assign higher weights to these features. We demonstrate the intuition with the following example. Consider a linear SVM, $N = 2$ and two data points $X_1 = (1,2)$ and $X_2 = (2,1)$ with labels $+1$ and -1 respectively. An SVM trained on this input learns a classifier with $w = (-0.599, +0.599)$. Thus, both features are deemed equally discriminative by the learned classifier. If feature 1 is indicated to be more discriminative by our user, then by our method $X'_1 = (10,2)$ and $X'_2 = (20,1)$ and $w' = (0.043, -0.0043)$, thus f_1 is assigned a much higher weight in the learned classifier. Now, this is a "soft" version of the feature selection mechanism of section 5. But in that case the oracle knew the ideal set of features. Those experiments may be viewed as a special case where $b = 0$. We expect that human feedback is imperfect and we do not want to zero-out potentially relevant features.

6.4 Simulating User Feedback

We use the relevance judgments on features obtained as described in Section 6.1 to simulate the user in each iteration. At each iteration of the algorithm, if a feature that is presented had been marked by the user as relevant, in the relevance judgment experiments of the previous section, we mark the value of that feature as 1 in the vector $\vec{s}$. The vector $\vec{s}$ is noisier (less complete) than the case where we would have obtained relevance judgments on features during the actual execution of the algorithm. This is because in addition to mistakes made by the user, we lose out on those features that the user might have considered relevant, had she been presented that feature when we were collecting relevance judgments for a relatively small subset of features. In a real life scenario this might correspond to the lazy user who labels few features as relevant and leaves some features unlabeled in addition to making mistakes.

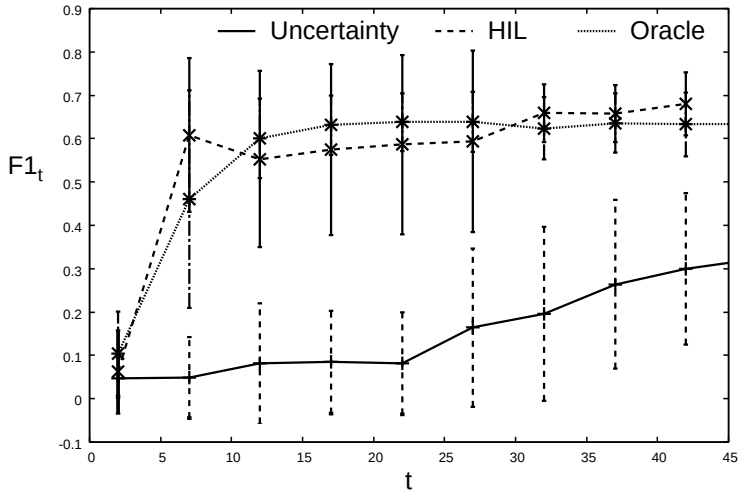
To make our experiments repeatable (to compute average performance and for convenience) we simulate user interaction as follows. For each classification problem we maintain a list of features that a user might have considered relevant had she been presented that feature. For these lists we used the judgments obtained in Section 4.2. Thus for each of the five classification problems we had two or three such lists, one per user who judged that topic. For the 10 TDT topics we have topic descriptions as provided by the LDC. These topic descriptions contain names of people, places and organizations that are key players in this topic in addition to other keywords. We used the words in these topic descriptions to be equal to the list of relevant features. Now, given these lists we can perform the simulated HIL (*human in the Loop*) experiments for 15 classification problems. Figure 10 shows the performance of the HIL experiments. Like before we report efficiency (E_{42}), the $F1$ score with 7 labeled documents ($F1_7$), and the $F1$ score with 22 labeled documents ($F1_{22}$) for each of uncertainty sampling (Unc), oracle feature selection with uncertainty sampling (Ora) and the Human in the Loop (HIL) algorithm. As a baseline we also report results for the case when the top 20 features as obtained by the information gain oracle are input to the simulated HIL experiments (this represents what a user with 100% precision and recall would obtain by our method). The oracle is (as expected) much better than plain uncertainty sampling, on all three measures, validating the effectiveness of our proposed system Section 2.1. The performance of the HIL experiments is almost as good as the oracle, indicating that user input (although imperfect)

3. We picked our algorithm's parameters based on a quick test on three topics (baseball, earnings, and acquisitions) using the oracle features of Section 5.

can help improve performance significantly. The plot on the right is of $F1_t(\text{HIL})$ for *hurricane Mitch*. As a comparison $F1_t(\text{ACT})$ is shown. The HIL values are much higher than for uncertainty sampling.

Dataset	E_{42}			$F1_7$			$F1_{22}$		
	Unc	Ora	HIL	Unc	Ora	HIL	Unc	Ora	HIL
Baseball	0.29	0.59	0.54	0.49	0.63	0.60	0.63	0.79	0.70
Earnings	0.10	0.36	0.36	0.61	0.79	0.73	0.80	0.85	0.86
Auto vs Motor	0.18	0.66	0.40	0.35	0.62	0.60	0.71	0.83	0.73
Hurr. Mitch	0.11	0.62	0.62	0.04	0.46	0.60	0.08	0.63	0.58
mideast	0.51	0.72	0.72	0.14	0.28	0.29	0.32	0.49	0.49
TDT (avg)	0.14	0.23	0.11	0.09	0.21	0.24	0.18	0.32	0.22

(a)



(b) The graph shows the learning curves for *Hurricane Mitch* (6th row of the above table) with the x-axis being the number of labeled documents and y-axis $F1(\text{HIL})$.

Figure 10: Improvements due to human feature selection. The $F1_7$ and $F1_{22}$ scores in the table show the points on the curves where 7 and 22 documents have been labeled. The difference between no feature feedback (Unc) and human-labeled features (HIL) is greatest with few documents labeled, but persists up to 42 documents labeled.

When to stop asking for labels on both features and documents and switch entirely to documents remains an area for future work. We provide some initial results in this regard. Consider that we ask for both document and feature feedback up to j iterations and after that we only ask for document feedback. Figure 11 shows the active learning curves for different values of j for the *hurricane Mitch* problem in the TDT corpus. The case when $j = 0$ represents traditional uncertainty sampling. When $j = 5$ there is improvement over the case when $j = 0$, and when $j = 10$ there is even more improvement. Beyond $j = 10$ there is little gain in obtaining feature feedback. It seems that relevant features are usually spotted in very early iterations. We see similar behavior for other problems in our domains. For the *auto vs motorcycles* problem, the user has been asked to label 75% of the

oracle features (averaged over multiple iterations and multiple users) at some point or the other. The most informative words (as determined by the oracle) – *car* and *bike* are asked of the user in very early iterations. The label for *car* is always (100% of the times) asked, and 70% of the time the label for this word is asked to the user in the first iteration itself. This is closely followed by the word *bike* which the user is queried about within the first five iterations 80% of the time. Most relevant features are asked within 10 iterations which makes us believe that we can often stop feature level feedback in around 10 iterations.

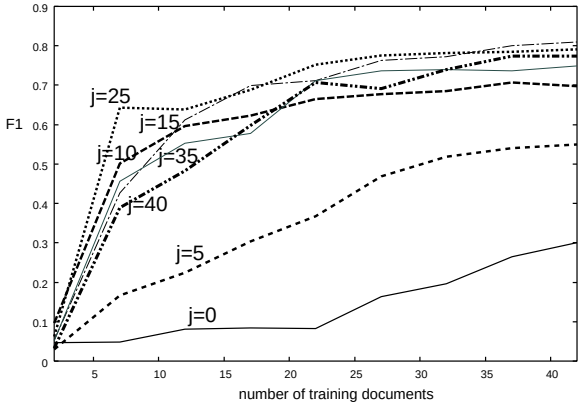


Figure 11: Human Feature Selection for *Hurricane Mitch* for different amounts of feature feedback. The legend indicates the number of iterations (j) for which there was both feature and document feedback, after which only document feedback was asked for. The line at the bottom, labeled $j = 0$ corresponds to regular uncertainty sampling or the case when feature feedback was asked for 0 iterations. The line corresponding to $j = 5$ iterations is significantly better than when $j = 0$. All other cases, $j = 10$ ... $j = 40$ are clumped at the top.

7. Related Work

Our work is related to a number of areas including query learning, active learning, use of (prior) knowledge and feature selection in machine learning, term-relevance feedback in information retrieval, and human-computer interaction.

Term level feedback has been studied in information retrieval (Anick, 2003; Croft and Das, 1990; Belkin et al., 2001). Many participants in the TREC HARD track (Voorhees and Buckland, 2005) generate clarification forms for users to refine or disambiguate their query. Many of the effective forms are composed of lists of terms and the user is asked to mark terms as relevant or not, and some have found that term level feedback is more effective than document level feedback (Diaz and Allan, 2005). The TREC interactive task has focused on issues regarding the kinds of questions that can be asked of the user. They find that users are happy to use interfaces which ask the user to reformulate their queries through a list of suggested terms. They also find that users are willing to mark both positive and negative terms (Belkin et al., 2001).

Our proposed method is an instance of query-based learning and an extension of standard ("pool-based") active learning which focuses on selective sampling of instances from a pool of unlabeled data alone (Cohn et al., 1994). Although query-based learning can be very powerful in theory (Angluin, 1992), arbitrary queries may be difficult to answer in practice (Baum and Lang, 1992). Hence the popularity of pool-based methods, and the motivation for studying the effectiveness and ease of predictive feature identification by humans in our application area. To best of our knowledge, all prior work on query learning and active learning focused on variants of membership queries, that is, requesting the label of a possibly synthesized instance. Our work is unique in the field of active learning as we extend the query model to include feature as well as document level feedback.

Feature feedback may be viewed as the teacher providing evidence or an explanation for the learner on the reasoning behind the labeling. The field of explanation-based learning, however, concerns itself on a deductive rather than an inductive learning task, using one instance and a given domain theory to generalize (Mitchell et al., 1986; DeJong and Mooney, 1986).

Feature selection can lead to improvements in the performance (accuracy) or in the space or time efficiency of the classifier. When there are sufficient labeled instances, most state of the art learning algorithms are able to distinguish the relevant features from the irrelevant ones (Brank et al., 2002). Hence there is little improvement in performance with an additional feature selection component. When there are few labeled instances, working with a small set of relevant features tends to be more useful. This phenomenon has been referred to in statistics as the Hughes phenomenon (Hughes, 1968). Weight regularization may be viewed as a soft version of feature selection: for best performance, in general the smaller the training set, the smaller the total weight that is allowed to be spread over the features. Unfortunately, to do automatic feature selection well, we need sufficient training data, leading to a chicken-and-egg problem. Fortunately, in document classification users have the intuition to point out a small subset of useful features which would be beneficial when there are few labeled instances.

Budgeted learning also works on identifying the predictive features during an active learning setting, but in this case the feature values are unknown and there is a cost to finding each feature's value for each instance of interest (such as the outcome of blood test on an individual) (Lizotte et al., 2003). That human prior knowledge can accelerate learning has been investigated by Pazzani and Kibler (1992), but our work differs in techniques (they use prior knowledge to generate horn-clause rules) and application domains. Beineke et al. (2004) use human prior knowledge of co-occurrence of words, at feature generation time, to improve classification of product reviews. None of this work, however, considers the use of prior knowledge in the active (sequential) learning setting.

Our study of the human factors (such as quality of feedback and costs) is also a major differentiating theme between our work from previous work in incorporating prior knowledge for training. Past work has not addressed this issue, or might have assumed experts in machine learning taking a role in training the system (Schapire et al., 2002; Wu and Srihari, 2004; Godbole et al., 2004; Jones, 2005). We only assume knowledge about the topic of interest. Our algorithmic techniques and the studied modes of interaction also differ somewhat and are worth further comparison. Jones (2005) also used single feature-set labeling in the context of active learning: the user was queried on a feature rather than the whole instance. The labeled feature was taken as a proxy for the label of any instance containing that feature, so a single feature labeling potentially labeled many documents (similar to the *soft* labeling technique discussed next). This was found to be more economical than whole-instance labeling for some tasks. The instances in this work consisted of only two features (a

noun-phrase and a context), so labeling one feature is equivalent to labeling half an instance. Our work differs in that our instances (documents) contain many features (words) and we combine both feature labeling and document labeling. Our work also differs in that we use the labeled features for feature selection and feature re-weighting, rather than as proxies for document labels.

Both Wu and Srihari (2004) and Schapire et al. (2002) assume that prior knowledge is given at the outset which leads to a "soft" labeling of the unlabeled data. This extra labeling is incorporated into training via modified boosting or SVM training. By soft labeling, we mean the extra labels, generated via prior knowledge, are not certain and a method that uses such information may for example assign low confidences to such labellings or lower the misclassification costs compared to misclassification costs for instances labeled directly by a human. However, in our scheme the user is labeling documents and features in an interactive and interleaved fashion. We expect that our proposed interactive mode has an advantage over requesting prior knowledge from the outset, as it may be easier for the user to identify or recall relevant features while labeling documents in the collection and being presented with candidate features. Our method of scaling the dimensions and training (without using the unlabeled data) has an advantage over soft labeling in situations where one may not have access to much unlabeled data, for example in online tasks such as filtering news streams and categorizing personal emails. Furthermore, we simplify the user's task in that our technique does not require the user to specify whether the feature is positively or negatively correlated with the category, just whether the user thinks the feature is relevant or predictive. On the other hand, in the presence of ample unlabeled data, soft labeling methods might more effectively incorporate the information available in the unlabeled data. Both approaches require extra parameters specifying how much to scale the dimensions or the confidence or misclassification costs to assign to the generated labellings, though some fixed parameter settings may work for most cases, or automated methods could be designed.

The work of Godbole et al. (2004) emphasizes system issues and focuses on multi-class training rather than a careful analysis of effects of feature selection and human efficacy. Their proposed method is attractive in that it treats features as single term documents that can be labeled by humans, but they also study labeling features before documents (and only in an "oracle" setting, without using actual human annotators). They do not observe much improvements using their particular method over standard active learning in the single domain (Reuters) they test on. Finally, we mention another method of incorporating prior knowledge that has much similarity to our method of differential scaling of dimensions: differential weightings of features in feature weight initializations when using online methods such as Winnow. A better understanding of effective ways of incorporating (prior) knowledge in various learning scenarios is a promising research direction.

8. Conclusions and Future Work

We have demonstrated experimentally that for learning with few labeled examples good (oracle-based) feature selection is extremely useful. As the number of examples increases, the "vocabulary" of the system, in other words, the effective feature set size for best performance, also needs to increase. A teacher, who may not necessarily be knowledgeable in machine learning, but has prior knowledge on the relevance of the features, can help accelerate training the system by pointing out the potentially important features for the system to focus on. We conducted a user study to see how well naive users performed as compared to a feature oracle in the domain of text categorization. Our technique weighted the features marked relevant by the users more than the other features. We

used our users' outputs in realistically simulated *human in the loop* experiments and observed a significant increase in learning performance with our techniques over plain active learning.

In summary, our contributions are:

1. We demonstrated that access to a feature importance oracle can improve performance (the $F1$ score) significantly over uncertainty sampling, even with as few as 7 examples labeled.

2. We found that even naive users can provide effective feedback on the most relevant features (about 60% accuracy of the oracle in our experiments).

3. We measured the manual costs of relevance feedback on features versus labeling documents: we found that feature feedback takes about one fifth of the time taken by document labeling on average.

4. We devised a method of simultaneously soliciting class labels and feature feedback that improves classifier performance significantly over soliciting class labels alone.

Consider a user who is interested in training a personalized news filter that delivers news stories on topics of their interest as and when they appear in the news. The user is probably willing to engage in some form of interaction in order to train the system to better suit their need. Similarly a user wanting to organize their e-mail into folders may be willing to train the e-mail filter as long as training is not too time consuming. Both the news filter and the e-mail filter are document classification systems. The idea of using as few documents as possible for training classifiers has been studied in semi-supervised learning and active learning. In this paper we extended the traditional active learning setting which concerns the issue of minimal feedback and proposed an approach where the user provides feedback on features as well as documents. We showed that such an approach has good potential in significantly decreasing the overall amount of interaction required for training the system.

This paper points to three promising inter-related questions for further exploration. The first question concerns what to ask from the user. In general, the active learner has to make decisions at various time points during active learning regarding the choice of feedback. For example, whether to ask for feedback on a document or on a feature, or even whether to stop asking questions all together (ask nothing), appropriate for a scenario where no additional feedback is likely to improve performance significantly. This involves some implicit or explicit assessment of the expected benefits and costs of different kinds of feedback. Furthermore, there are alternate kinds of feedback that are potentially useful – feedback on clusters of features for example. The second question involves human computer interaction issues and seeks to explore how to translate what the learner needs to know, into a question, or a user interface, that the human teacher can easily understand. In our case, the learner asked the teacher labels on word features and documents, both of which required little effort on the part of the teacher to understand what was being asked of him. Our subjects did indeed find labeling words without context a little hard, and suggested that context might have helped. An attractive alternative or complementary method of soliciting feature feedback is asking users to highlight some relevant or predictive terms as they read a document. Experiments in this direction have been conducted in information retrieval (Croft and Das, 1990). The third question is about the choice of learning algorithms for effectively incorporating these alternate forms of feedback. We explored one method in this paper and discussed alternatives in Section 7. Related to the above is better understanding and quantifying the potential of active learning enhanced with feature feedback

as a function of various aspects of the learning problem, such as measures of the difficulty of the category that one seeks to learn.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor. We would also like to thank our users who willingly labeled data. We are also grateful to the action editor and the anonymous reviewers who helped enhance this paper with their useful comments.

Appendix A. Class Key

The class key for the Reuters corpus is given below:

1. earnings	2. acquisitions	3. money-fx	4. crude	5. trade	6. interest	7. wheat
8. corn	9. money supply	10. gold				

The class key for the 20 Newsgroups corpus is given below:

1. alt.atheism	2. comp.graphics	3. comp.os.wind.misc	4. comp.sys.ibm.pc.hw
5. comp.sys.mac.hw	6. comp.windows.x	7. misc.forsale	8. rec.autos
9. rec.motorcycles	10. rec.sport.baseball	11. rec.sport.hockey	12. sci.crypt
13. sci.electronics	14. sci.med	15. sci.space	16. soc.rel.christian
17. talk.politics.guns	18. talk.politics.mideast	19. talk.politics.misc	20. talk.religion.misc

Similarly the class key for the TDT corpus is:

1. Cambodian government coalition	2. Hurricane Mitch	3. Pinochet Trial
4. Chukwu Octuplets	5. Bin Laden Indictment	6. NBA Labor Disputes
7. Congolese Rebels	8. APEC Summit Meeting	9. Anti-Doping Proposals
10. Car Bomb in Jerusalem		

Appendix B. Instructions for Annotating Features

Class 1: Documents from the Usenet newsgroups that discuss baseball
Class 2: Documents from the Usenet newsgroups that discuss hockey

Instructions: You will be shown a list of features one at a time. For each feature you will be asked to determine whether it is relevant or not for the given classification problem. If it is relevant to Class 1 or to Class 2, mark the radio button which says “Relevant”. If it is not relevant or you don’t know whether the feature is relevant mark DONT KNOW correspondingly

A feature is relevant if it helps discriminate between documents in Class 1 versus documents in Class 2. Features are words, pairs of words (bi grams) and so on. Think of a bi gram as a pair of words that may occur in close proximity to each other For every feature ask yourself the following question: “Is this more likely to occur in a document in Class 1 as opposed to Class 2?”. If that is the

case mark the feature as relevant. If the reverse is true then again mark the feature as relevant. If the feature is not really relevant, for example "banana" may make no sense in trying to find documents in either class mark the "Not relevant/Don't know" option. DO NOT use any resources(the web, encyclopedias etc) to determine your answer. If you are not sure simply click the "Don't Know" option

The time between which you are shown a feature and you hit the submit button is timed. So do not do anything else in this time. After you submit, A THANK YOU page is displayed. You may take a break here before you proceed to the next feature.

To modify the last annotation use the browsers BACK button.

To begin annotating click here.

Appendix C. Instructions for Annotating Documents

Class 1: Documents from the Usenet newsgroups that discuss baseball
Class 2: Documents from the Usenet newsgroups that discuss hockey

Instructions: You will be shown a list of documents one at a time. For each documents you will be asked to determine whether it belongs to class 1 or class 2. You also have the option to mark a document as DONT KNOW. Read as much of the document as is needed to make an informed judgment. The time between which you are shown a document and you hit the submit button is timed. So do not do anything else in this time. After you submit, A THANK YOU page is displayed. You may take a break here before you proceed to the next document.

To modify the last annotation use the browsers BACK button

To begin annotating click here

Appendix D. End of Labeling Survey

Please take 2 minutes to fill out the following:

1. How easy was it to mark features?
   (a) On an integer scale of 1-5 (1=very difficult, 5=very easy) (b) Remarks:

2. How easy was it to mark documents?
   (a) On an integer scale of 1-5 (1=very difficult, 5=very easy) (b) Remarks:

3. For each of the following tasks please state your domain knowledge (only if you did relevance assessments for them) on a scale of 1-5 (1=very little, 5=expert):
   (a) Baseball versus Hockey. (b) Earnings versus All.
   (c) Automobiles versus Motorcycles. (d) Hurricane Mitch versus all.
   (e) Middle eastern crisis versus all.

4. Your Internet connection
   (a) DSL/Cable (b) T1 LAN (c) Dial-up

References

J. Allan. *Topic detection and tracking*. Kluwer Academic Publishers, 2002.

D. Angluin. Computational learning theory: survey and selected bibliography. In *Proceedings of the 24th Annual ACM Symposium on the Theory Computation*, pages 351–369, 1992.

P. Anick. Using terminological feedback for web search refinement: a log-based study. In *Proceedings of SIGIR '03: The 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 88–95, 2003.

Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. In *Proceedings of ICML 03: The 20th International Conference on Machine Learning*, pages 19–26, 2003.

E. B. Baum and K. Lang. Query learning can work poorly when human oracle is used. In *International Joint Conference in Neural Netwroks*, 1992.

P. Beineke, T. Hastie, and S. Vaithyanathan. The sentimental factor: Improving review classification via human-provided information. In *Proceedings of ACL 04: The 42nd Meeting of the Association for Computational Linguistics, Main Volume*, pages 263–270, 2004.

N. J. Belkin, C. Cool, D. Kelly, S. J. Lin, S. Y. Park, J. Perez-Carballo, and C. Sikora. Iterative exploration, design and evaluation of support for query reformulation in interactive information retrieval. *Information Processing and Management*, 37(3):403–434, 2001.

J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Feature selection using linear support vector machines. Technical report, Microsoft Research, 2002.

C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines. Available electronically at `http://www.csie.ntu.edu.tw/cjlin/libsvm`.

J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:27–46, 1960.

D. A. Cohn, L. Atlas, and R. E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

W. B. Croft and R. Das. Experiments with query acquisition and use in document retrieval systems. In *Proceedings of SIGIR '90: The 13th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 349–368, 1990.

G. DeJong and R. Mooney. Explanation-based generalization: an alternative view. *Machine Learning*, 1(2):145–176, 1986.

F. Diaz and J. Allan. When less is more: Relevance feedback falls short and term expansion succeeds at HARD 2005. In *Text REtrieval Conference (TREC 2005) Notebook*. Dept. of Commerce, NIST, 2005.

C. Domeniconi and D. Gunopulos. Incremental support vector machine construction. In *Proceedings of ICDM 01:2001 IEEE International Conference on Data Mining*, pages 589–592, 2001.

S. Godbole, A. Harpale, S. Sarawagi, and S. Chakrabarti. Document classification through interactive supervision of document and term labels. In *Proceedings of PKDD 04: The 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 185–196, 2004.

G. F. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14:55–63, 1968.

T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *ECML 98: The 10th European Conference on Machine Learning*, pages 137–142, 1998.

T. Joachims. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-612-2.

R. Jones. *Learning to extract entities from labeled and unlabeled text*. PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 2005.

G. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174, 1977.

K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of ICML 95: The 12th International Conference on Machine Learning*, pages 331–339, 1995.

D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML 98: 10th European Conference on Machine Learning*, pages 4–15, 1998.

D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of ICML 94: The 11th International Conference on Machine Learning*, pages 148–156, 1994.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

D. J. Lizotte, O. Madani, and R. Greiner. Budgeted learning of naive-bayes classifiers. In *Proceedings of UIA 03: The 19th Conference on Uncertainty in AI (UAI)*, 2003.

A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Available electronically at `http://www.cs.cmu.edu/~mccallum/bow`, 1996.

T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

M. J. Pazzani and D. Kibler. The role of prior knowledge in inductive learning. *Machine Learning, 9 , 54-97.*, 9, 1992.

M. Porter. An algorithm for suffix stripping. *Automated Library and Information Systems*, 14(3): 130–137, 1980.

H. Raghavan, O. Madani, and R. Jones. Interactive feature selection. In *Proceedings of IJCAI 05: The 19th International Joint Conference on Artificial Intelligence*, pages 841–846, 2005.

T. G. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Vol. 1 - from yesterday's news to tomorrow's language resources. In *Proceedings of International Conference on Language Resources and Evaluation*, 2002.

G. Salton. *Automatic information organization and retrieval*. McGraw Hill, 1968.

R. Schapire, M. Rochery, M. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *Proceedings of ICML 02: The 19th International Conference on Machine Learning*, 2002.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of ICML 00: The 17th International Conference on Machine Learning*, pages 839–846, 2000.

F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1): 1–47, 2002.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2002. ISSN 1533-7928.

E. M. Voorhees and L. P. Buckland, editors. *Text REtrieval Conference (TREC 2005) Notebook*, 2005. Dept of Commerce, NIST.

X. Wu and R. Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *Proceedings of KDD 04: Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 326–333, 2004.

X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Journal of Machine Learning Research 7 (2006) 1687–1712 Submitted 12/05; Revised 4/06; Published 8/06

Large Scale Transductive SVMs

Ronan Collobert RONAN@COLLOBERT.COM
NEC Laboratories America
4 Independence Way
Princeton, NJ 08540, USA

Fabian Sinz FABEE@TUEBINGEN.MPG.DE
NEC Laboratories America
4 Independence Way
Princeton, NJ 08540, USA, and
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tuebingen, Germany

Jason Weston JASEWESTON@GMAIL.COM
Léon Bottou LEON@BOTTOU.ORG
NEC Laboratories America
4 Independence Way
Princeton, NJ 08540, USA

Editor: Thorsten Joachims

Abstract

We show how the concave-convex procedure can be applied to transductive SVMs, which traditionally require solving a combinatorial search problem. This provides for the first time a highly scalable algorithm in the nonlinear case. Detailed experiments verify the utility of our approach. Software is available at `http://www.kyb.tuebingen.mpg.de/bs/people/fabee/transduction.html`.

Keywords: transduction, transductive SVMs, semi-supervised learning, CCCP

1. Introduction

Transductive support vector machines (TSVMs) (Vapnik, 1995) are a method of improving the generalization accuracy of SVMs (Boser et al., 1992) by using unlabeled data. TSVMs, like SVMs, learn a large margin hyperplane classifier using labeled training data, but simultaneously force this hyperplane to be far away from the unlabeled data.

One way of justifying this algorithm, in the context of *semi-supervised learning* is that one is finding a decision boundary that lies in a region of low density, implementing the so-called cluster assumption (see e.g. Chapelle and Zien, 2005). In this framework, if you believe the underlying distribution of the two classes is such that there is a "gap" or low density region between them, then TSVMs can help because it selects a rule with exactly those properties. Vapnik (1995) has a different interpretation for the success of TSVMs, rooted in the idea that transduction (labeling a test set) is inherently easier than induction (learning a general rule). In either case, experimentally

it seems clear that algorithms such as TSVMs can give considerable improvement in generalization over SVMs, if the number of labeled points is small and the number of unlabeled points is large.

Unfortunately, TSVM algorithms (like other semi-supervised approaches) are often unable to deal with a large number of unlabeled examples. The first implementation of TSVM appeared in (Bennett and Demiriz, 1998), using an integer programming method, intractable for large problems. Joachims (1999b) then proposed a combinatorial approach, known as SVMLight-TSVM, that is practical for a few thousand examples. Fung and Mangasarian (2001) introduced a sequential optimization procedure that could potentially scale well, although their largest experiment used only 1000 examples. However, their method was for the linear case only, and used a special kind of SVM with a 1-norm regularizer, to retain linearity. Finally, Chapelle and Zien (2005) proposed a primal method, which turned out to show improved generalization performance over the previous approaches, but still scales as $(L+U)^3$, where L and U are the numbers of labeled and unlabeled examples. This method also stores the entire $(L+U)\times(L+U)$ kernel matrix in memory. Other methods (Bie and Cristianini, 2004; Xu et al., 2005) transform the non-convex transductive problem into a convex semi-definite programming problem that scales as $(L+U)^4$ or worse.

In this article we introduce a large scale training method for TSVMs using the concave-convex procedure (CCCP) (Yuille and Rangarajan, 2002; Le Thi, 1994), expanding on the conference proceedings paper (Collobert et al., 2006). CCCP iteratively optimizes non-convex cost functions that can be expressed as the sum of a convex function and a concave function. The optimization is carried out iteratively by solving a sequence of convex problems obtained by linearly approximating the concave function in the vicinity of the solution of the previous convex problem. This method is guaranteed to find a local minimum and has no difficult parameters to tune. This provides what we believe is the best known method for implementing transductive SVMs with an empirical scaling of $(L+U)^2$, which involves training a sequence of typically 1-10 conventional convex SVM optimization problems. As each of these problems is trained in the dual we retain the SVM's linear scaling with problem dimensionality, in contrast to the techniques of Fung and Mangasarian (2001).

2. The Concave-Convex Procedure for TSVMs

Notation We consider a set of L training pairs $\mathcal{L} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_L, y_L)\}$, $\boldsymbol{x} \in \mathbb{R}^n$, $y \in \{1, -1\}$ and an (unlabeled) set of U test vectors $\mathcal{U} = \{x_{L+1}, \ldots, x_{L+U}\}$. SVMs have a decision function $f_\theta(.)$ of the form

$$f_\theta(x) = w \cdot \Phi(\boldsymbol{x}) + b\,,$$

where $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ are the parameters of the model, and $\Phi(\cdot)$ is the chosen feature map, often implemented implicitly using the kernel trick (Vapnik, 1995).

TSVM Formulation The original TSVM optimization problem is the following (Vapnik, 1995; Joachims, 1999b; Bennett and Demiriz, 1998). Given a training set $\mathcal{L}$ and a test set $\mathcal{U}$, find among the possible binary vectors

$$\{\mathcal{Y} = (y_{L+1}, \ldots, y_{L+U})\}$$

the one such that an SVM trained on $\mathcal{L} \cup (\mathcal{U} \times \mathcal{Y})$ yields the largest margin.

This is a combinatorial problem, but one can approximate it (see Vapnik, 1995) as finding an SVM separating the training set under constraints which force the unlabeled examples to be as far

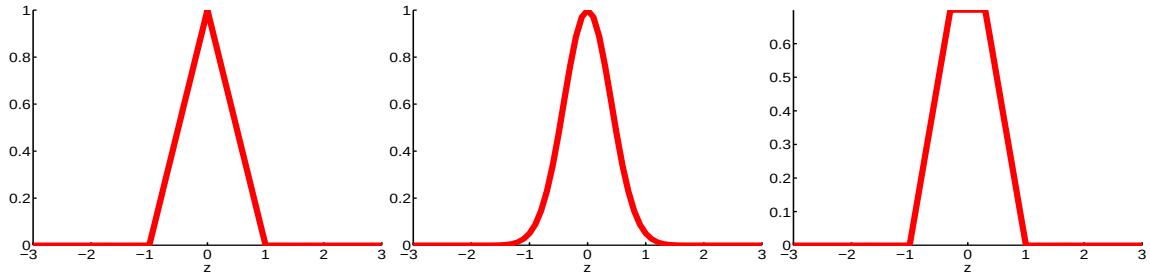


Figure 1: Three loss functions for unlabeled examples, from left to right (i) the Symmetric Hinge $H_1(|t|) = \max(0, 1-|t|)$, (ii) Symmetric Sigmoid $S(t) = \exp(-3t^2)$; and (iii) Symmetric Ramp loss, $R_s(|t|) = \min(1+s, \max(0, 1-|t|))$. The last loss function has a plateau of width $2|s|$ where $s \in (-1, 0]$ is a tunable parameter, in this case $s = -0.3$.

as possible from the margin. This can be written as minimizing

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+U}\xi_i$$

subject to

$$y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1-\xi_i, \;\; i = 1,\ldots,L$$
$$|f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)| \geq 1-\xi_i, \;\; i = L+1,\ldots,L+U$$

This minimization problem is equivalent to minimizing

$$J(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^*\sum_{i=L+1}^{L+U} H_1(|f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)|), \tag{1}$$

where the function $H_1(\cdot) = \max(0, 1-\cdot)$ is the classical Hinge Loss (Figure 2, center). The loss function $H_1(|\cdot|)$ for the unlabeled examples can be seen in Figure 1, left. For $C^* = 0$ in (1) we obtain the standard SVM optimization problem. For $C^* > 0$ we penalize unlabeled data that is inside the margin. This is equivalent to using the hinge loss on the unlabeled data as well, but where we assume the label for the unlabeled example is $y_i = \text{sign}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$.

Losses for transduction TSVMs implementing formulation (1) were first introduced in SVM-Light (Joachims, 1999b). As shown above, it assigns a Hinge Loss $H_1(\cdot)$ on the labeled examples (Figure 2, center) and a "Symmetric Hinge Loss" $H_1(|\cdot|)$ on the unlabeled examples (Figure 1, left). More recently, Chapelle and Zien (2005) proposed to handle unlabeled examples with a smooth version of this loss (Figure 1, center). While we also use the Hinge Loss for labeled examples, we use for unlabeled examples a slightly more general form of the Symmetric Hinge Loss, that we allow to be "non-peaky" (Figure 1, right). Given an unlabeled example $\boldsymbol{x}$ and using the notation $z = f_{\boldsymbol{\theta}}(\boldsymbol{x})$, this loss can be written as

$$z \mapsto R_s(z) + R_s(-z) + const.^1, \tag{2}$$

where $-1 < s \leq 0$ is a hyper-parameter to be chosen and $R_s = \min(1-s, \max(0, 1-t))$ is what we call the "Ramp Loss", a "clipped" version of the Hinge Loss (Figure 2, left).

1. The constant does not affect the optimization problem we will later describe.

Losses similar to the Ramp Loss have been already used for different purposes, like in the Doom II algorithm (Mason et al., 2000) or in the context of "Ψ-learning" (Shen et al., 2003). The s parameter controls where we clip the Ramp Loss, and as a consequence it also controls the wideness of the flat part of the loss (2) we use for transduction: when $s = 0$, this reverts to the Symmetric Hinge $H_1(|\cdot|)$. When $s \neq 0$, we obtain a non-peaked loss function (Figure 1, right) which can be viewed as a simplification of Chapelle's loss function. We call this loss function (2) the "Symmetric Ramp Loss".

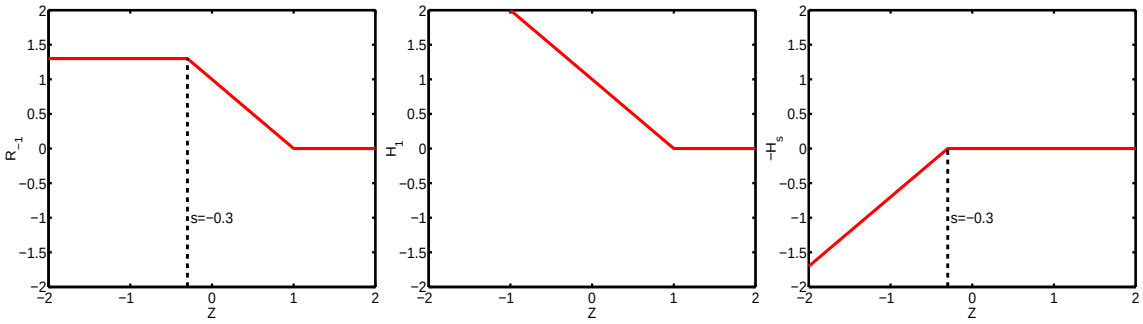


Figure 2: The Ramp Loss function $R_s(t) = \min(1-s, \max(0, 1-t)) = H_1(t) - H_s(t)$ (left) can be decomposed into the sum of the convex Hinge Loss (center) and a concave loss (right), where $H_s(t) = \max(0, s-t)$. The parameter s controls the cutoff point of the usual Hinge loss.

Training a TSVM using the loss function (2) is equivalent to training an SVM using the Hinge loss $H_1(\cdot)$ for labeled examples, and using the Ramp loss $R_s(\cdot)$ for unlabeled examples, where each unlabeled example appears as two examples labeled with both possible classes. More formally, after introducing

$$\begin{array}{rcll} y_i & = & 1 & i \in [L+1 \ldots L+U] \\ y_i & = & -1 & i \in [L+U+1 \ldots L+2U] \\ \boldsymbol{x}_i & = & \boldsymbol{x}_{i-U} & i \in [L+U+1 \ldots L+2U]\,, \end{array}$$

we can rewrite (1) as

$$J^s(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} R_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\,. \tag{3}$$

This is the minimization problem we now consider in the rest of the paper.

Balancing constraint One problem with TSVM as stated above is that in high dimensions with few training examples, it is possible to classify all the unlabeled examples as belonging to only one of the classes with a very large margin, which leads to poor performance. To cure this problem, one further constrains the solution by introducing a balancing constraint that ensures the unlabeled data are assigned to both classes. Joachims (1999b) directly enforces that the fraction of positive and negatives assigned to the unlabeled data should be the same fraction as found in the labeled data. Chapelle and Zien (2005) use a similar but slightly relaxed constraint, which we also use in this work:

$$\frac{1}{U}\sum_{i=L+1}^{L+U} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \frac{1}{L}\sum_{i=1}^{L} y_i\,. \tag{4}$$

Concave-Convex Procedure (CCCP) Unfortunately, the TSVM optimization problem as given above is not convex, and minimizing a non-convex cost function is often considered difficult. Gradient descent techniques, such as conjugate gradient descent or stochastic gradient descent, often involve delicate hyper-parameters (LeCun et al., 1998). In contrast, convex optimization seems much more straight-forward. For instance, the SMO algorithm (Platt, 1999) locates the SVM solution efficiently and reliably.

We propose to solve this non-convex problem using the "Concave-Convex Procedure" (CCCP) (Yuille and Rangarajan, 2002). The CCCP procedure is closely related to the "Difference of Convex" (DC) methods that have been developed by the optimization community during the last two decades (Le Thi, 1994). Such techniques have already been applied for dealing with missing values in SVMs (Smola et al., 2005), for improving boosting algorithms (Krause and Singer, 2004), and in the "Ψ-learning" framework (Shen et al., 2003).

Assume that a cost function $J(\theta)$ can be rewritten as the sum of a convex part $J_{vex}(\theta)$ and a concave part $J_{cav}(\theta)$. Each iteration of the CCCP procedure (Algorithm 1) approximates the concave part by its tangent and minimizes the resulting convex function.

Algorithm 1 : The concave-convex procedure (CCCP)

Initialize θ^0 with a best guess.
repeat

$$\theta^{t+1} = \arg\min_{\theta} \left(J_{vex}(\theta) + J'_{cav}(\theta^t) \cdot \theta \right) \tag{5}$$

until convergence of θ^t

One can easily see that the cost $J(\theta^t)$ decreases after each iteration by summing two inequalities resulting from (5) and from the concavity of $J_{cav}(\theta)$.

$$\begin{aligned} J_{vex}(\theta^{t+1}) + J'_{cav}(\theta^t) \cdot \theta^{t+1} &\leq J_{vex}(\theta^t) + J'_{cav}(\theta^t) \cdot \theta^t & (6) \\ J_{cav}(\theta^{t+1}) &\leq J_{cav}(\theta^t) + J'_{cav}(\theta^t) \cdot \left(\theta^{t+1} - \theta^t\right) & (7) \end{aligned}$$

The convergence of CCCP has been shown by Yuille and Rangarajan (2002) by refining this argument. The authors also showed that the CCCP procedure remains valid if θ is required to satisfy some linear constraints. Note that no additional hyper-parameters are needed by CCCP. Furthermore, each update (5) is a convex minimization problem and can be solved using classical and efficient convex algorithms.

CCCP for TSVMs Interestingly, the Ramp Loss can be rewritten as the difference between two Hinge losses (see Figure 2):

$$R_s(z) = H_1(z) - H_s(z)\,. \tag{8}$$

Because of this decomposition, the TSVM minimization problem as stated in (3) is amenable to CCCP optimization. The cost $J^s(\theta)$ can indeed be decomposed into a convex $J^s_{vex}(\theta)$ and concave

$J^s_{cav}(\boldsymbol{\theta})$ part as follows:

$$\begin{aligned} J^s(\boldsymbol{\theta}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_\theta(\boldsymbol{x}_i)) + C^*\sum_{i=L+1}^{L+2U} R_s(y_i\, f_\theta(\boldsymbol{x}_i)) \\ &= \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_\theta(\boldsymbol{x}_i)) + C^*\sum_{i=L+1}^{L+2U} H_1(y_i\, f_\theta(\boldsymbol{x}_i))}_{J^s_{vex}(\boldsymbol{\theta})} \\ &\quad \underbrace{-C^*\sum_{i=L+1}^{L+2U} H_s(y_i\, f_\theta(\boldsymbol{x}_i))}_{J^s_{cav}(\boldsymbol{\theta})} . \end{aligned} \tag{9}$$

This decomposition allows us to apply the CCCP procedure as stated in Algorithm 1. The convex optimization problem (5) that constitutes the core of the CCCP algorithm is easily reformulated into dual variables $\boldsymbol{\alpha}$ using the standard SVM technique.

After some algebra, we show in Appendix A that enforcing the balancing constraint (4) can be achieved by introducing an extra Lagrangian variable α_0 and an example $\boldsymbol{x}_0$ implicitely defined by

$$\Phi(\boldsymbol{x}_0) = \frac{1}{U}\sum_{i=L+1}^{L+U} \Phi(\boldsymbol{x}_i)\,,$$

with label $y_0 = 1$. Thus, if we note K the kernel matrix such that

$$K_{ij} = \Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j)\,,$$

the column corresponding to the example $\boldsymbol{x}_0$ is computed as follow:

$$K_{i0} = K_{0i} = \frac{1}{U}\sum_{j=L+1}^{L+U} \Phi(\boldsymbol{x}_j)\cdot\Phi(\boldsymbol{x}_i) \quad \forall i\,. \tag{10}$$

The computation of this special column can be achieved very efficiently by computing it only once, or by approximating the sum (10) using an appropriate sampling method.

Given the decomposition of the cost (9) and the trick of the special extra example (10) to enforce the balancing constraint, we can easily apply Algorithm 1 to TSVMs. To simplifiy the first order approximation of the concave part in the CCCP procedure (5), we denote

$$\beta_i \;=\; y_i\frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_\theta(\boldsymbol{x}_i)} \;=\; \begin{cases} C^* & \text{if } y_i\, f_\theta(\boldsymbol{x}_i) < s \\ 0 & \text{otherwise} \end{cases}\,, \tag{11}$$

for unlabeled examples (that is $i \geq L+1$).[2] The concave part J^s_{cav} does not depend on labeled examples ($i \leq L$) so we obviously have $\beta_i = 0$ for all $i \leq L$. This yields Algorithm 2, after some standard derivations detailed in Appendix A.

Convergence of Algorithm 2 in finite time t^* is guaranteed because variable $\boldsymbol{\beta}$ can only take a finite number of distinct values, because $J(\boldsymbol{\theta}^t)$ is decreasing, and because inequality (7) is strict unless $\boldsymbol{\beta}$ remains unchanged.

2. Note that $J^s_{cav}(\cdot)$ is non-differentiable at $z = s$, because $H_s(\cdot)$ is not. It can be shown that the CCCP remains valid when using any super-derivative of the concave function. Alternatively, the function $H_s(\cdot)$ could be made smooth in a small interval $[s-\varepsilon, s+\varepsilon]$.

Algorithm 2 : CCCP for TSVMs

Initialize $\boldsymbol{\theta}^0 = (\boldsymbol{w}^0, b^0)$ with a standard SVM solution on the labeled points.

Compute $\beta_i^0 = \begin{cases} C^* & \text{if } y_i f_{\boldsymbol{\theta}^0}(\boldsymbol{x}_i) < s \text{ and } i \geq L+1 \\ 0 & \text{otherwise} \end{cases}$

Set $\zeta_i = y_i$ for $1 \leq i \leq L+2U$ and $\zeta_0 = \frac{1}{L}\sum_{i=1}^{L} y_i$

repeat

- **Solve** the following convex problem (with $K_{ij} = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$)

$$\max_{\boldsymbol{\alpha}} \left(\boldsymbol{\alpha} \cdot \boldsymbol{\zeta} - \frac{1}{2} \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{\alpha} \right) \text{ subject to } \begin{cases} \boldsymbol{\alpha} \cdot \mathbf{1} = 0 \\ 0 \leq y_i \alpha_i \leq C \ \ \forall 1 \leq i \leq L \\ -\beta_i \leq y_i \alpha_i \leq C^* - \beta_i \ \ \forall i \geq L+1 \end{cases}$$

- **Compute** b^{t+1} using $f_{\theta^{t+1}}(x_i) = \sum_{j=0}^{L+2U} \alpha_j^{t+1} K_{ij} + b^{t+1}$ and

$$\begin{array}{rrcl} \forall i \leq L: & 0 < y_i \alpha_i < C & \Longrightarrow & y_i f_{\theta^{t+1}}(x_i) = 1 \\ \forall i > L: & -\beta_i < y_i \alpha_i < C^* - \beta_i & \Longrightarrow & y_i f_{\theta^{t+1}}(x_i) = 1 \end{array}$$

- **Compute** $\beta_i^{t+1} = \begin{cases} C^* & \text{if } y_i f_{\boldsymbol{\theta}^{t+1}}(\boldsymbol{x}_i) < s \text{ and } i \geq L+1 \\ 0 & \text{otherwise} \end{cases}$

until $\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t$

Complexity The main point we want to emphasize in this paper is the advantage in terms of training time of our method compared to existing approaches. Training a CCCP-TSVM amounts to solving a series of SVM optimization problems with $L+2U$ variables. Although SVM training has a worst case complexity of $O((L+2U)^3)$ it typically scales quadratically (see Joachims, 1999a; Platt, 1999), and we find this is the case for our TSVM subproblems as well. Assuming a constant number of iteration steps, the whole optimization of TSVMs with CCCP should scale quadratically in most practical cases (see Figure 3, Figure 8 and Figure 9). From our experience, around five iteration steps are usually sufficient to reach the minimum, as shown in the experimental section of this paper, Figure 4.

3. Previous Work

SVMLight-TSVM Like our work, the heuristic optimization algorithm implemented in SVM-Light (Joachims, 1999b) solves successive SVM optimization problems, but on $L+U$ instead of $L+2U$ data points. It improves the objective function by iteratively switching the labels of two unlabeled points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with $\xi_i + \xi_j > 2$. It uses two nested loops to optimize a TSVM which solves a quadratic program in each step. The convergence proof of the inner loop relies on the fact that there is only a finite number 2^U of labelings of U unlabeled points, even though it is unlikely that all of them are examined. However, since the heuristic only swaps the labels of two unlabeled examples at each step in order to enforce the balancing constraint, it might need many iterations to reach a minimum, which makes it intractable for big data set sizes in practice (cf. Figure 3).

SVMLight uses annealing heuristics for the selection of C^*. It begins with a small value of C^* ($C^* = 1e-5$), and multiplies C^* by 1.5 on each iteration until it reaches C. The numbers $1e-5$ and 1.5 are hard coded into the implementation. On each iteration the tolerance on the gradients is also changed so as to give more approximate (but faster) solutions on earlier iterations. Again, several heuristics parameters are hard coded into the implementation.

∇TSVM The ∇TSVM of Chapelle and Zien (2005) is optimized by performing gradient descent in the primal space: minimize

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H^2(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^*\sum_{i=L+1}^{L+U} H^*(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)),$$

where $H^2(t) = \max(0, 1-t)^2$ and $H^*(t) = \exp(-3t^2)$ (cf. Figure 1, center). This optimization problem can be considered a smoothed version of (1). ∇TSVM also has similar heuristics for C^* as SVMLight-TSVM. It begins with a small value of C^* ($C^* = bC$), and iteratively increases C^* over l iterations until it finally reaches C. The values $b = 0.01$ and $l = 10$ are default parameters in the code available at: `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds`.

Since the gradient descent is carried out in the primal, to learn nonlinear functions it is necessary to perform kernel PCA (Schölkopf et al., 1997). The overall algorithm has a time complexity equal to the square of the number of variables times the complexity of evaluating the cost function. In this case, evaluating the objective scales linearly in the number of examples, so the overall worst case complexity of solving the optimization problem for ∇TSVM is $o((U+L)^3)$. The KPCA calculation alone also has a time complexity of $o((U+L)^3)$. This method also requires one to store the entire kernel matrix in memory, which clearly becomes infeasible for large data sets.

CS^{3}VM The work of Fung and Mangasarian (2001) is algorithmically the closest TSVM approach to our proposal. Following the formulation of transductive SVMs found in Bennett and Demiriz (1998), the authors consider transductive linear SVMs with a 1-norm regularizer, which allow them to decompose the corresponding loss function as a sum of a linear function and a concave function. Bennett proposed the following formulation which is similar to (1): minimize

$$||\boldsymbol{w}||_1 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{U}\min(\xi_i, \xi_i^*)$$

subject to

$$y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1-\xi_i,\ \ i = 1,\ldots,L$$

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1-\xi_i,\ \ i = L+1,\ldots,L+U$$

$$-(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1-\xi_i^*,\ \ i = L+1,\ldots,L+U$$

$$\xi_i \geq 0, \xi_i^* \geq 0.$$

The last term of the objective function is nonlinear and corresponds to the loss function given in Figure 1, left. To deal with this, the authors suggest to iteratively approximate the concave part as a linear function, leading to a series of linear programming problems. This can be viewed as a simplified subcase of CCCP (a linear function being convex) applied to a special kind of SVM.

Note also that the algorithm presented in their paper did not implement a balancing constraint for the labeling of the unlabeled examples as in (4). Our transduction algorithm is nonlinear and the use of kernels, solving the optimization in the dual, allows for large scale training with high dimensionality and number of examples.

4. Small Scale Experiments

This section presents small scale experiments appropriate for comparing our algorithm with existing TSVM approaches. In order to provide a direct comparison with published results, these experiments use the same setup as (Chapelle and Zien, 2005). All methods use the standard RBF kernel, $\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}') = \exp(-\gamma||\boldsymbol{x} - \boldsymbol{x}'||^2)$.

data set	classes	dims	points	labeled
g50c	2	50	500	50
Coil20	20	1024	1440	40
Text	2	7511	1946	50
Uspst	10	256	2007	50

Table 1: Small-Scale Data Sets. We used the same data sets and experimental setup in these experiments as found in Chapelle and Zien (2005).

	Coil20	g50c	Text	Uspst	(number of hyperparameters)
SVM	24.64	8.32	18.86	23.18	2
SVMLight-TSVM	26.26	6.87	7.44	26.46	2
∇TSVM	17.56	5.80	5.71	17.61	2
CCCP-TSVM$|^{s=0}_{UC^*=LC}$	16.69	5.62	7.97	16.57	2
CCCP-TSVM$|_{UC^*=LC}$	16.06	5.04	5.59	16.47	3
CCCP-TSVM	15.92	3.92	4.92	16.45	4

Table 2: Results on Small-Scale Data Sets. We report the best test error over the hyperparameters of the algorithms, as in the methodology of Chapelle and Zien (2005). SVMLight-TSVM is the implementation in SVMLight. ∇TSVM is the primal gradient descent method of Chapelle and Zien (2005). CCCP-TSVM$|^{s=0}_{UC^*=LC}$ reports the results of our method using the heuristic $UC^* = LC$ with the Symmetric Hinge Loss, that is with $s = 0$. We also report CCCP-TSVM$|_{UC^*=LC}$ where we allow the optimization of s, and CCCP-TSVM where we allow the optimization of both C^* and s.

Table 1 lists the data sets we have used. The `g50c` data set is an artificial data set where the labels correspond to two Gaussians in a 50-dimensional space. The means of those Gaussians are

placed in such a way that the Bayes error is 5%. The `coil20` data is a set of gray-scale images of 20 different objects taken from different angles, in steps of 5 degrees (S.A.Nene et al., 1996). The `text` data set consists of the classes `mswindows` and `mac` of the `Newsgroup20` data set preprocessed as in Szummer and Jaakkola (2001a). The `uspst` data set is the test part of the `USPS` hand written digit data. All data sets are split into ten parts with each part having a small amount of labeled examples and using the remaining part as unlabeled data.

4.1 Accuracies

Consistent with (Chapelle and Zien, 2005), all hyperparameters are tuned on the test set. Chapelle and Zien (2005) argue that, in order to perform algorithm comparisons, it is sufficient to be "*interested in the best performance and simply select the parameter values minimizing the test error*". However we should be more cautious when comparing algorithms that have different sets of hyperparameters. For CCCP-TSVMs we have two additional parameters, C^* and s. Therefore we report the CCCP-TSVM error rates for three different scenarios:

- CCCP-TSVM, where all four parameters are tuned on the test set.
- CCCP-TSVM$|_{UC^*=LC}$ where we choose C^* using a heuristic method. We use heuristic $UC^* = LC$ because it decreases C^* when the number of unlabeled data increases. Otherwise, for large enough U no attention will be paid to minimizing the training error. Further details on this choice are given in Section 4.3.
- CCCP-TSVM$|^{s=0}_{UC^*=LC}$ where we choose $s = 0$ and C^* using heuristic $UC^* = LC$. This setup has the same free parameters (C and γ) as the competing TSVM implementations, and therefore provides the most fair comparison.

The results are reported in Table 2. CCCP-TSVM in all three scenarios achieves approximately the same error rates as ∇TSVM and appears to be superior to SVMLight-TSVM. Section 4.3 provides additional results using different hyperparameter selection strategies and discusses more precisely the impact of each hyperparameter.

4.2 Training Times

At this point we ask the reader to simply assume that all authors have chosen their hyperparameter selection method as well as they could. We now compare the computation times of these three algorithms.

The CCCP-TSVM algorithm was implemented in C++.[3] The successive convex optimizations are performed using a state-of-the-art SMO implementation. Without further optimization, CCCP-TSVMs run orders of magnitude faster than SVMLight-TSVMs and ∇TSVM.[4] Figure 3 shows the training time on `g50c` and `text` for the three methods as we vary the number of unlabeled examples. For each method we report the training times for the hyperparameters that give optimal performance as measured on the test set on the first split of the data (we use CCCP-TSVM$|^{s=0}_{UC^*=LC}$ in these experiments). Using all 2000 unlabeled data on Text, CCCP-TSVMs are approximately *133 times faster* than SVMLight-TSVM and *50 times faster* than ∇TSVM.

3. Source code available at `http://www.kyb.tuebingen.mpg.de/bs/people/fabee/transduction.html`.
4. ∇TSVM was implemented by adapting the Matlab LDS code of Chapelle and Zien (2005) available at `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds`.

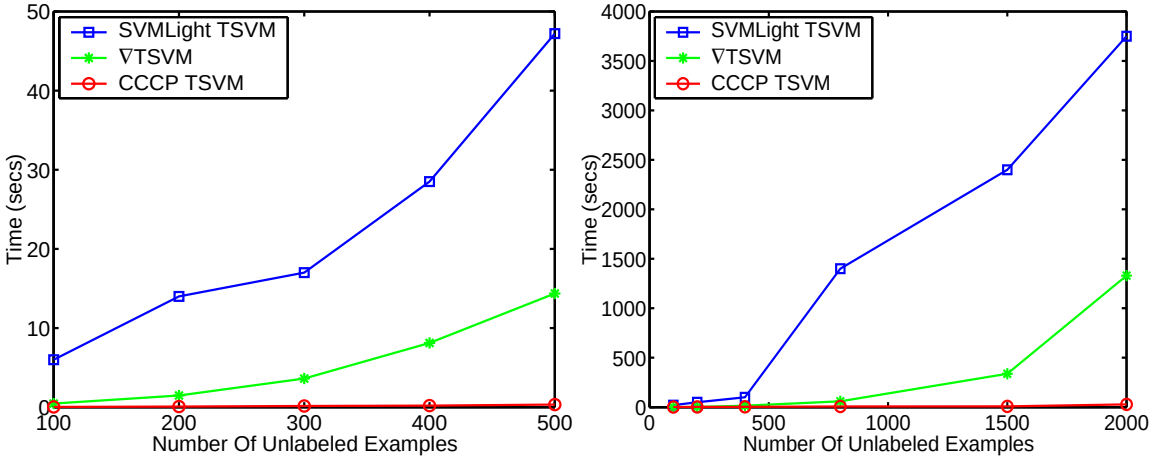


Figure 3: Training times for `g50c` (left) and `text` (right) for SVMLight-TSVMs, ∇TSVMs and CCCP-TSVMs using the best parameters for each algorithm as measured on the test set in a single trial. For the Text data set, using 2000 unlabeled examples CCCP-TSVMs are **133x** faster than SVMLight-TSVMs, and **50x** faster than ∇TSVMs.

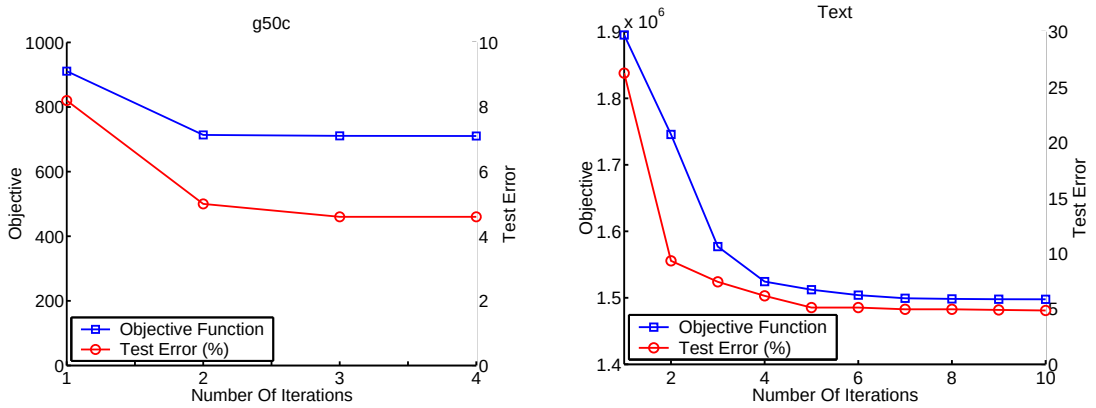


Figure 4: Value of the objective function and test error during the CCCP iterations of training TSVM on two data sets (single trial), `g50c` (left) and `text` (right). CCCP-TSVM tends to converge after only a few iterations.

We expect these differences to increase as the number of unlabeled examples increases further. In particular, ∇TSVM requires the storage of the entire kernel matrix in memory, and is therefore clearly infeasible for some of the large scale experiments we attempt in Section 5.

Finally, Figure 4 shows the value of the objective function and test error during the CCCP iterations of training TSVM on two data sets. The CCCP-TSVM objective function converges after five to ten iterations.

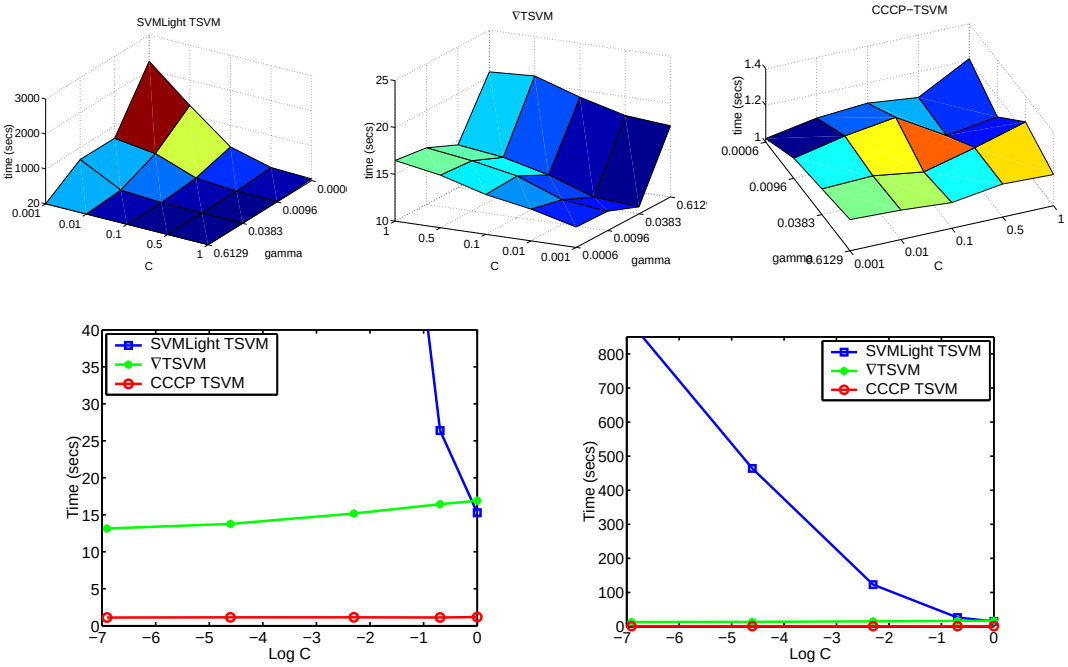


Figure 5: Computation times for different choices of hyperparameters on data set `g50c` (first split only) for the three TSVM implementations tested (top three figures). The bottom two figures show the computation time for all three algorithms with respect to the parameter C only, where the time is the mean training time taken over the possible values of γ. The bottom right figure is a scale up of the bottom left figure, as SVMLight-TSVM is so slow it hardly appears on the left figure. In general, SVMLight-TSVM computation time appears very sensitive to parameter choices, with small values of C and γ resulting in computation times around 2250 seconds, whereas large values of C and γ are much faster. ∇ TSVM has almost the opposite trend on this data set: it is slower for large values of C or γ, although even the slowest time is still only around 20 seconds. Our CCCP-TSVM takes only around 1 second for all parameter choices.

4.3 Hyperparameters

We now discuss in detail how the hyperparameters γ, C, C^* and s affect the performance of the TSVM algorithms.

Effect of the parameters γ and C. The parameters γ and C have similar effects on generalization as in the purely supervised SVM approach (see Keerthi and Lin. (2003) for an empirical study). However, during model selection, one has to try many choices of parameters. Some algorithms have different computational behaviour across different parameter choices. Therefore we have studied how different choices of C and γ affect the computation times of all three TSVM algorithms. Figure 5 compares these computation times for the `g50c` data set. SVMLight-TSVM is particularly slow for small γ and C, taking up to 2250 seconds, whereas the other two algorithms are relatively more stable. In particular, CCCP-TSVM takes only around 1 second for every possible parameter

choice. This means that during cross validation the CCCP-TSVM speedup over SVMLight-TSVM is even larger than the 133x speedup observed for the relatively benign choice of hyperparameters in Figure 3.

Effect of the parameter C^* As mentioned before, both SVMLight-TSVM and ∇TSVM use an annealing heuristic for hyperparameter C^*. They start their optimization using a small value of C^* and slowly increase C^* until it reaches the final desired value $C^* = C$. CCCP-TSVM solves the optimization problem for the desired value of C^* without an annealing heuristic. When one wishes to avoid optimizing C^*, we suggest the heuristic $UC^* = LC$.

Comparing the heuristics $C^* = C$ *and* $UC^* = LC$ Table 3 compares the $C^* = C$ and $UC^* = LC$ heuristics on the small scale data sets. Results are provided for the case $s = 0$ and the case where we allow the optimization of s. Although the heuristic $C^* = C$ gives reasonable results for small amounts of unlabeled data, we prefer the heuristic $UC^* = LC$. When the number of unlabeled examples U becomes large, setting $C^* = C$ will mean the third term in the objective function (1) will dominate, resulting in almost no attention being paid to minimizing the training error. In these experiments the heuristic $UC^* = LC$ is close to the best possible choice of C^*, whereas $C^* = C$ is a little worse.

We also conducted an experiment to compare these two heuristics for larger unlabeled data sizes U. We took the same `uspst` data set (that is, the test part of the USPS data set) and we increased the number of unlabeled examples by adding up to 6000 additional unlabeled examples taken from the original USPS training set. Figure 6 reports the best test error for both heuristics over possible choices of γ and C, taking the mean of the same 10 training splits with 50 labeled examples as before. The results indicate that $C^* = C$ works poorly for large U.

	Coil20	g50c	Text	Uspst	(number of hyperparameters)
CCCP-TSVM$\|^{s=0}_{C^*=C}$	22.33	4.68	7.76	20.09	2
CCCP-TSVM$\|^{s=0}_{UC^*=LC}$	16.69	5.62	7.97	16.57	2
CCCP-TSVM$\|^{s=0}$	16.67	4.56	7.76	16.55	3
CCCP-TSVM$\|_{C^*=C}$	19.02	4.28	5.22	18.33	3
CCCP-TSVM$\|_{UC^*=LC}$	16.06	5.04	5.59	16.47	3
CCCP-TSVM	15.92	3.92	4.92	16.45	4

Table 3: Comparison of $C^* = C$ and $C^* = \frac{L}{U}C$ heuristics on on Small-Scale Data Sets with the best optimized value of C^* (CCCP-TSVM$|^{s=0}$ or CCCP-TSVM, depending on whether s is fixed). The heuristic $C^* = \frac{L}{U}C$ maintains the balance between unlabeled points U and labeled points L as U and L change, and is close to the best possible choice of C^*. The $C^* = C$ heuristic also works for relatively small values of U as in this case. We report all methods with and without the optimization of s.

Iteratively increasing C^* — Iteratively increasing C^* during the optimization can be interpreted as starting from a convex problem ($C^* = 0$) and gradually making it more non-convex, which may be a good strategy to solve such non-convex problems. However, we believe that the annealing procedure

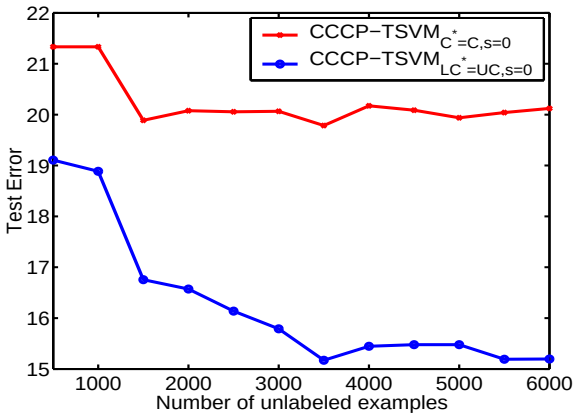


Figure 6: Comparison of the $C^* = C$ and $UC^* = LC$ heuristics on the `uspst` data set as we increase the number of unlabeled examples by adding extra unlabeled data from the original usps training set. We report the best test error for both heuristics over possible of choices of γ and C, taking the mean of the same 10 training splits with 50 labeled examples as before. As the number of unlabeled examples increases, the $C^* = C$ heuristic gives too much weight to the unlabeled data, resulting in no improvement in performance. Intuitively, the $UC^* = LC$ heuristic balances the weight of the unlabeled and labeled data and empirically performs better.

also has a regularizing effect. The optimization is more likely to get stuck in a local minimum that appears when C^* has a value much smaller than C. This may be why the $C^* = C$ heuristic works well for algorithms that also use the annealing trick.

We conducted an experiment to see the performance of SVMLight-TSVM with and without the annealing heuristic. On `g50c`, we chose a linear kernel and computed the optimal value of C on the test set using $C^* = C$. With the annealing heuristic, we obtain a test error of 7.6%. For the same parameters without the annealing procedure, we obtain 12.4%. Clearly the annealing heuristic has a strong effect on the results of SVMLight-TSVM. CCCP-TSVM has no such heuristic.

Effect of the parameter s The parameter s in CCCP-TSVM controls the choice of loss function to minimize over. It controls the size of the plateau of the Symmetric Ramp function (Figure 1, right). Training our algorithm with a tuned value of s appears to give slightly improved results over using the Symmetric Hinge loss ($s = 0$, see Figure 1, left), especially on the `text` data set, as can be seen in Tables 2 and 3. Furthermore, Figure 7 highlights the importance of the parameter s of the loss function (2) by showing the best test error over different choices of s for two data sets, `text` and `g50c`.

We conjecture that the peaked loss of the Symmetric Hinge function forces early decisions for the β variables and might lead to a poor local optimum. This effect then disappears as soon as we clip the loss. That is, the flat part of the loss far inside the margin prevents our algorithm from making erroneous early decisions regarding the labels of the unlabeled data that may be hard to undo later in the optimization.

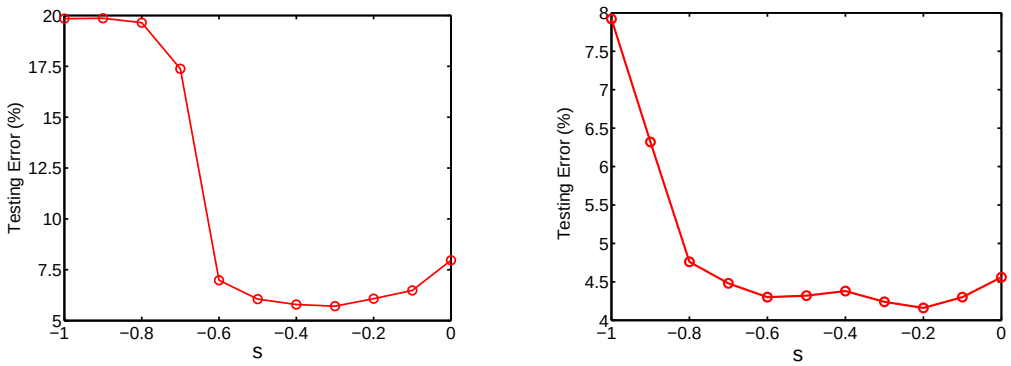


Figure 7: Effect of the parameter s of the Symmetric Ramp loss (see Figure 1 and equation (2)) on the `text` data set (left) and the `g50c` data set (right). The peaked loss of the Symmetric Hinge function ($s = 0$) forces early decisions for the β variables and might lead to a poor local optimum. This effect then disappears as soon as we clip the loss.

In fact, the $\nabla TSVM$ authors make a similar argument to explain why they prefer their algorithm over SVMLight: *"(SVMLight) TSVM might suffer from the combinatorial nature of its approach. By deciding, from the very first step, the putative label of every point (even with low confidence), it may lose important degrees of freedom at an early stage and get trapped in a bad local minimum*".

Here, the authors are refering to the way SVMLight TSVM has a discrete rather than continuous approach of assigning labels to unlabeled data. However, we think that the smoothed loss function of ∇TSVM may help it to outperform the Symmetric Hinge loss of SVMLight TSVM, making it similar to the clipped loss when we use $s < 0$. Indeed, the ∇TSVM smoothed loss, $\exp(-3t^2)$, has small gradients when t is close to 0.

A potential issue of the Symmetric Ramp loss is the fact that the gradient is exactly 0 for points lying on the plateau. Points are not updated at all in this region. This may be sub-optimal: if we are unlucky enough that all unlabeled points lie in this region, we perform no updates at all. Performing model selection on parameter s eliminates this problem. Alternatively, we could use a piece-wise linear loss with two different slopes for $|f(x)| > s$ and for $|f(x)| < s$. Although it is possible to optimize such a loss function using CCCP, we have not evaluated this approach.

5. Large Scale Experiments

In this section, we provide experimental results on large scale experiments. Since other methods are intractable on such data sets, we only compare CCCP-TSVM against SVMs.

5.1 RCV1 Experiments

The first large scale experiment that we conducted was to separate the two largest top-level categories CCAT (CORPORATE/INDUSTRIAL) and GCAT (GOVERNMENT/SOCIAL) of the training part of the Reuters data set as prepared by Lewis et al. (2004). The set of these two categories consists of 17754 documents. The features are constructed using the bag of words technique, weighted with a TF.IDF scheme and normalized to length one. We performed experiments using 100 and 1000 la-

Method	Train size	Unlabeled size	Parameters	Test Error
SVM	100	0	$C = 252.97, \sigma = 15.81$	16.61%
TSVM	100	500	$C = 2.597, C^* = 10, s = -0.2, \sigma = 3.95$	11.99%
TSVM	100	1000	$C = 2.597, C^* = 10, s = -0.2, \sigma = 3.95$	11.67%
TSVM	100	2000	$C = 2.597, C^* = 10, s = -0.2, \sigma = 3.95$	11.47%
TSVM	100	5000	$C = 2.597, C^* = 2.5297, s = -0.2, \sigma = 3.95$	10.65%
TSVM	100	10000	$C = 2.597, C^* = 2.5297, s = -0.4, \sigma = 3.95$	10.64%
SVM	1000	0	$C = 25.297, \sigma = 7.91$	11.04%
TSVM	1000	500	$C = 2.597, C^* = 10, s = -0.4, \sigma = 3.95$	11.09%
TSVM	1000	1000	$C = 2.597, C^* = 2.5297, s = -0.4, \sigma = 3.95$	11.06%
TSVM	1000	2000	$C = 2.597, C^* = 10, s - 0.4 =, \sigma = 3.95$	10.77%
TSVM	1000	5000	$C = 2.597, C^* = 2.5297, s = -0.2, \sigma = 3.95$	10.81%
TSVM	1000	10000	$C = 2.597, C^* = 25.2970, s = -0.4, \sigma = 3.95$	10.72%

Table 4: Comparing CCCP-TSVMs with SVMs on the RCV1 problem for different number of labeled and unlabeled examples. See text for details.

beled examples. For model selection we use a validation set with 2000 and 4000 labeled examples for the two experiments. The remaining 12754 examples were used as a test set.

We chose the parameter C and the kernel parameter γ (using an RBF kernel) that gave the best performance on the validation set. This was done by training a TSVM using the validation set as the unlabeled data. These values were then fixed for every experiment.

We then varied the number of unlabeled examples U, and reported the test error for each choice of U. In each case we performed model selection to find the parameters C^* and s. A selection of the results can be seen in Table 4.

The best result we obtained for 1000 training points was 10.58% test error, when using 10500 unlabeled points, and for 100 training points was 10.42% when using 9500 unlabeled points. Compared to the best performance of an SVM of 11.04% for the former and 16.61% for the latter, this shows that unlabeled data can improve the results on this problem. This is especially true in the case of few training examples, where the improvement in test error is around 5.5%. However, when enough training data is available to the algorithm, the improvement is only in the order of one percent.

Figure 8 shows the training time of CCCP optimization as a function of the number of unlabeled examples. On a 64 bit Opteron processor the optimization time for 12500 unlabeled examples was approximately 18 minutes using the 1000 training examples and 69 minutes using 100 training examples. Although the worst case complexity of SVMs is cubic and the optimization time seems to be dependent on the ratio of the number of labeled to unlabeled examples, the training times show a quadratic trend.

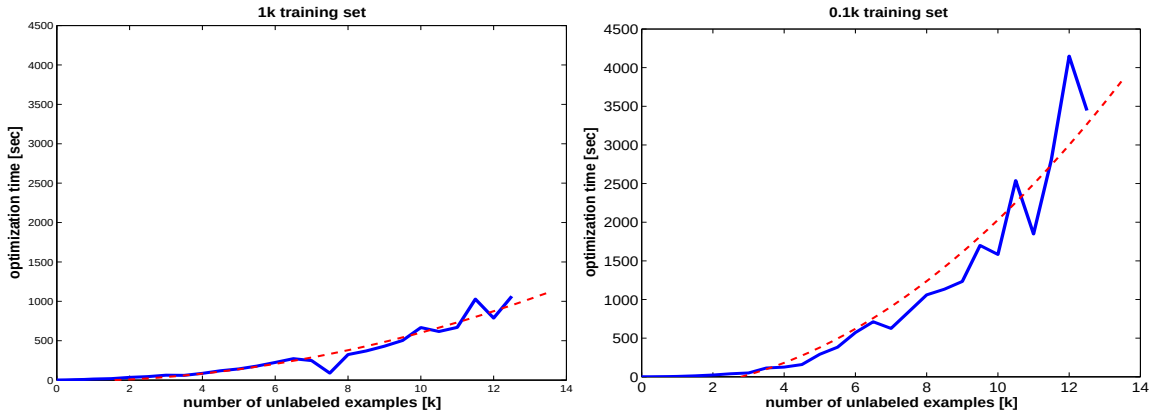


Figure 8: Optimization time for the Reuters data set as a function of the number of unlabeled data. The algorithm was trained on 1,000 points (left) and on 100 points (right). The dashed lines represent a parabola fitted at the time measurements.

Method	Training size	Unlabeled size	Parameters	Test Error
SVM	100	0	$C = 10, \gamma = 0.0128$	23.44%
TSVM	100	2000	$C^* = 1, s = -0.1$	16.81%
SVM	1000	0	$C = 10, \gamma = 0.0128$	7.77%
TSVM	1000	2000	$C^* = 5, s = -0.1$	7.13%
TSVM	1000	5000	$C^* = 1, s = -0.1$	6.28%
TSVM	1000	10000	$C^* = 0.5, s = -0.1$	5.65%
TSVM	1000	20000	$C^* = 0.3, s = -0.1$	5.43%
TSVM	1000	40000	$C^* = 0.2, s = -0.1$	5.31%
TSVM	1000	60000	$C^* = 0.1, s = -0.1$	5.38%

Table 5: Comparing CCCP-TSVMs with SVMs on the MNIST problem for different number of labeled and unlabeled examples. See text for details.

5.2 MNIST Experiments

In the second large scale experiment, we conducted experiments on the MNIST handwritten digit database, as a 10-class problem. The original data has 60,000 training examples and 10,000 testing examples. We subsampled the training set for labeled points, and used the test set for unlabeled examples (or the test set plus remainder of the training set when using more than 10,000 unlabeled examples). We performed experiments using 100 and 1000 labeled examples. We performed model selection for 1-vs-the-rest SVMs by trying a grid of values for σ and C, and selecting the best ones by using a separate validation set of size 1000. For TSVMs, for efficiency reasons we fixed the values of σ and C to be the same ones as chosen for SVMs. We then performed model selection using 2000 unlabeled examples to find the best choices of C^* and s using the validation set. When using

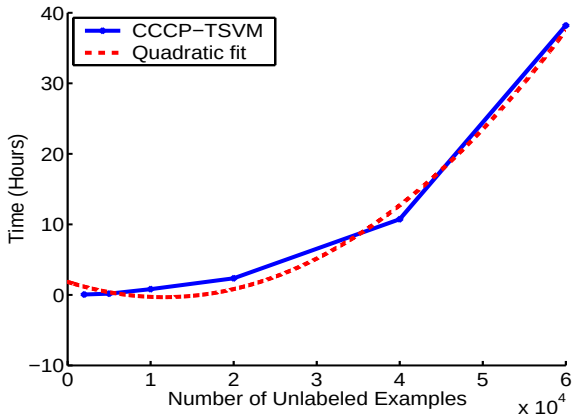


Figure 9: Optimization time for the MNIST data set as a function of the number of unlabeled data. The algorithm was trained on 1,000 labeled examples and up to 60,000 unlabeled examples. The dashed lines represent a polynomial of degree two with a least square fit on the algorithm's time measurements.

more unlabeled data, we only reperformed model selection on C^* as it appeared that this parameter was the most sensitive to changes in the unlabeled set, and kept the other parameters fixed. For the larger labeled set we took 2000, 5000, 10000, 20000, 40000 and 60000 unlabeled examples. We always measure the error rate on the complete test set. The test error rate and parameter choices for each experiment are given in the Table 5, and the training times are given in Figure 9.

The results show an improvement over SVM for CCCP-TSVMs which increases steadily as the number of unlabeled examples increases. Most experiments in semi-supervised learning only use a few labeled examples and do not use as many unlabeled examples as described here. It is thus reassuring to know that these methods do not apply just to toy examples with around 50 training points, and that gains are still possible with more realistic data set sizes.

6. Discussion and Conclusions

TSVMs are not the only means of using unlabeled data to improve generalization performance on classification tasks. In the following we discuss some competing algorithms for utilizing unlabeled data, and also discuss the differences between the transductive and semi-supervised learning frameworks. Finally, we conclude with some closing remarks.

6.1 Cluster Kernels and Manifold-Learning

Transductive SVMs are not the only method of leveraging unlabeled data in a supervised learning task. In recent years this has become a popular research topic, and a battery of techniques have been proposed. One popular class of methods, which we refer to as cluster kernels, do not change the learning algorithm at all, but merely the representation of the data as a pre-processing step. In a purely unsupervised fashion, these methods learn cluster or manifold structure from the data, and produce a new representation of it such that distances between points in the new space are small if they are in the same cluster or on the same manifold. Some of the main methods include (Chapelle

et al., 2002; Chapelle and Zien, 2005; Sindhwani et al., 2005; Szummer and Jaakkola, 2001b); and (Weston et al., 2003).

Other notable methods include generalizations of nearest-neighbor or Parzen window type approaches to learning manifolds given labeled data (Zhu et al., 2003; Belkin and Niyogi, 2002; Zhou et al., 2004). Finally, Bayesian approaches have also been pursued (Graepel et al., 2000; Lawrence and Jordan, 2005).

We note that some cluster kernel methods (Chapelle and Zien, 2005) can perform significantly better than TSVM on some data sets. In fact, Chapelle and Zien (2005) show that, as these methods provide a new representation, one can just as easily run TSVM on the new representation. The combination of TSVM and cluster kernels then provides state-of-the-art results.

6.2 Semi-Supervised Versus Transductive Learning

From a theoretical point of view, there is much ideological debate over which underlying theory that explains TSVM is correct. The argument here is largely about which framework, semi-supervised learning or transductive, is interesting to study theoretically or to apply practically.

Semi-supervised school The majority of researchers appear to be in the *semi-supervised* school of thought, which claims that TSVMs help simply because of a regularizer that reflects prior knowledge, see e.g. (Chapelle and Zien, 2005). That is, one is given a set of unlabeled data, and one uses it to improve an inductive classifier to improve its generalization on an unseen test set.

Transductive school Vapnik (1982) describes *transduction* as a mathematical setup for describing learning algorithms that benefit from the prior knowledge of the unlabeled test patterns. Vapnik claims that transduction is an essentially easier task than first learning a general inductive rule and then applying it to the test examples. Transductive bounds address the performance of the trained system on these test patterns only. They do not apply to test patterns that were not given to the algorithm in the first place. As a consequence, transductive bounds are purely derived from combinatorial arguments (Vapnik, 1982) and are more easily made data-dependent (Bottou et al., 1994; Derbeko et al., 2004). Whether this is a fundamental property or a technical issue is a matter of debate.

Experiments The following experiments attempt to determine whether the benefits of TSVMs are *solely* caused by the prior knowledge represented by the distribution of the unlabeled data. If this is the case, the accuracy should not depend on the presence of the actual test patterns in the unlabeled data.

The following experiments consider three distinct subsets: a small labeled training set and two equally sized sets of unlabeled examples. Generalization accuracies are always measured on the third set. On the other hand, we run CCCP-TSVM using either the second or the third set as unlabeled data. We respectively name these results "Semi-Supervised TSVM" and "Transductive TSVM". Experiments were carried out on both the Text and MNIST data set (class 8 vs rest) using ten splits. For Text, we fixed to a linear kernel, $C = 1000$, and $s = -0.3$. For MNIST-8 we fixed $\gamma = 0.0128$ and $C = 10$. We report the best test error over possible values of C^*. Table 6 shows that transductive TSVMs perform slightly better than semi-supervised TSVMs on these data sets.

Transductive TSVMs are only feasible when the test patterns are known before training. In that sense, its applicability is more limited than that of Semi-Supervised TSVMs. On the other hand, when the test and training data are not identically distributed, we believe the concept of transduction could be particularly worthwhile.

	Text	MNIST-8
SVM	18.86%	6.68%
semi-supervised TSVM	6.60%	5.27%
transductive TSVM	**6.12%**	**4.87%**

Table 6: Transductive TSVM versus Semi-Supervised TSVM.

6.3 Conclusion and Future Directions

In this article we have described an algorithm for TSVMs using CCCP that brings scalability improvements over existing implementation approaches. It involves the iterative solving of standard dual SVM QP problems, and usually requires just a few iterations. One nice thing about being an extension of standard SVM training is that any improvements in SVM scalability can immediately also be applied to TSVMs. For example in the linear case, one could easily apply fast linear SVM training such as in (Keerthi and DeCoste, 2005) to produce very fast linear TSVMs. For the non-linear case, one could apply the online SVM training scheme of Bordes et al. (2005) to give a fast online transductive learning procedure.

Acknowledgments

We thank Hans Peter Graf, Eric Cosatto, and Vladimir Vapnik for their advice and support. Part of this work was funded by NSF grant CCR-0325463.

Appendix A. Derivation of the Optimization Problem

We consider a set of L training pairs $\mathcal{L} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_L, y_L)\}$, $\boldsymbol{x} \in \mathbb{R}^n$, $y \in \{1, -1\}$ and a (unlabeled) set of U test vectors $\mathcal{U} = \{x_{L+1}, \ldots, x_{L+U}\}$. SVMs have a decision function $f_\theta(.)$ of the form

$$f_\theta(x) = w \cdot \Phi(\boldsymbol{x}) + b\,,$$

where $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ are the parameters of the model, and $\Phi(\cdot)$ is the chosen feature map.

We are interested in minimizing the TSVM cost function (3), under the constraint (4). We rewrite the problem here for convenience: minimizing

$$J^s(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_\theta(\boldsymbol{x}_i)) + C^* \sum_{i=L+1}^{L+2U} R_s(y_i\, f_\theta(\boldsymbol{x}_i))\,, \tag{12}$$

under the constraint

$$\frac{1}{U}\sum_{i=L+1}^{L+U} f_\theta(\boldsymbol{x}_i) = \frac{1}{L}\sum_{i=1}^{L} y_i\,. \tag{13}$$

Assume that a cost function $J(\boldsymbol{\theta})$ can be rewritten as the sum of a convex part $J_{vex}(\boldsymbol{\theta})$ and a concave part $J_{cav}(\boldsymbol{\theta})$. As mentioned above in Algorithm 1, the minimization of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ ($\boldsymbol{\theta}$ being possibly restricted to a space $\mathcal{A}$ defined by some linear constraints) can be achieved by iteratively updating the parameters $\boldsymbol{\theta}$ using the following update

$$\boldsymbol{\theta}^{t+1} = \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\mathcal{A}} \left(J_{vex}(\boldsymbol{\theta}) + J'_{cav}(\boldsymbol{\theta}^t)\cdot\boldsymbol{\theta}\right)\,. \tag{14}$$

In the case of our cost (12), we showed (see (9)) that $J^s(\boldsymbol{\theta})$ can be decomposed into the sum of $J^s_{vex}(\boldsymbol{\theta})$ and $J^s_{cav}(\boldsymbol{\theta})$ where

$$J^s_{vex}(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + C^*\sum_{i=L+1}^{L+2U} H_1(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \tag{15}$$

and

$$J^s_{cav}(\boldsymbol{\theta}) = -C^*\sum_{i=L+1}^{L+2U} H_s(y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\,. \tag{16}$$

In order to apply the CCCP update (14) we first have to calculate the derivative of the concave part (16) with respect to $\boldsymbol{\theta}$:

$$\frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -C^*\sum_{i=L+1}^{L+2U} \frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}\frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}$$

We introduce the notation

$$\begin{aligned}
\beta_i &= y_i \frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)} \\
&= \begin{cases} C^* H_s'[y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)] & \text{if } i \geq L+1 \\ 0 & \text{otherwise} \end{cases} \\
&= \begin{cases} C^* & \text{if } y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) < s \text{ and } i \geq L+1 \\ 0 & \text{otherwise} \end{cases} .
\end{aligned}$$

Since $f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i) + b$ with $\theta = (w, b)$, and $\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)/\partial\boldsymbol{\theta} = (\Phi(\boldsymbol{x}_i), 1)$, each update (14) of the CCCP procedure applied to the our minimization problem (12) consists in minimizing the following cost

$$\begin{aligned}
J^s_{vex}(\boldsymbol{\theta}) + \frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\cdot\boldsymbol{\theta} &= J^s_{vex}(\boldsymbol{\theta}) + \left(\sum_{i=L+1}^{L+2U} y_i\beta_i \frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}\right)\cdot\boldsymbol{\theta} \\
&= J^s_{vex}(\boldsymbol{\theta}) + \sum_{i=L+1}^{L+2U} \beta_i y_i\,[\boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i) + b]\,,
\end{aligned} \tag{17}$$

under the linear constraint (13).

The convex part (16) contains Hinge Losses which can be rewritten as

$$H_1(z) = \max(0, 1-z) = \min\xi \;\text{ s.t }\; \xi \geq 0, \xi \geq 1 - z\,.$$

It is thus easy to see that the minimization of (17) under the constraint (13) is equivalent to the following quadratic minimization problem under constraints:

$$\begin{aligned}
\underset{\boldsymbol{\theta},\boldsymbol{\xi}}{\arg\min} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i + \sum_{i=L+1}^{L+2U}\beta_i y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \\
\text{s.t.} \quad & \frac{1}{U}\sum_{i=L+1}^{L+U} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \frac{1}{L}\sum_{i=1}^{L} y_i && (18) \\
& y_i\, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1 - \xi_i \quad \forall 1 \leq i \leq L+2U && (19) \\
& \xi_i \geq 0 \quad \forall 1 \leq i \leq L+2U && (20)
\end{aligned}$$

Introducing Lagrangian variables α_0, $\boldsymbol{\alpha}$ and $\boldsymbol{\nu}$ corresponding respectively to constraints (18), (19) and (20), we can write the Lagrangian of this problem as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\nu}) =& \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{L}\xi_i + C^*\sum_{i=L+1}^{L+2U}\xi_i + \sum_{i=L+1}^{L+2U}\beta_i y_i f_\theta(\boldsymbol{x}_i) \\ & - \alpha_0\left(\frac{1}{U}\sum_{i=L+1}^{L+U} f_\theta(\boldsymbol{x}_i) - \frac{1}{L}\sum_{i=1}^{L} y_i\right) \\ & - \sum_{i=1}^{L+2U}\alpha_i\left(y_i f_\theta(\boldsymbol{x}_i) - 1 + \xi_i\right) \\ & - \sum_{i=1}^{L+2U}\nu_i\xi_i\,,\end{aligned} \tag{21}$$

where α_0 can be positive or negative (equality constraint) and $\alpha_i,\ i \geq 1$ are non-negative (inequality constraints).

Taking into account that $\beta_i = 0$ for $i \leq L$, calculating the derivatives with respect to the primal variables yields

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} &= \boldsymbol{w} - \sum_{i=1}^{L+2U} y_i(\alpha_i-\beta_i)\Phi(\boldsymbol{x}_i) - \frac{\alpha_0}{U}\sum_{i=L+1}^{L+U}\Phi(\boldsymbol{x}_i) \\ \frac{\partial\mathcal{L}}{\partial b} &= -\sum_{i=1}^{L+2U} y_i(\alpha_i-\beta_i) - \alpha_0 \\ \frac{\partial\mathcal{L}}{\partial\xi_i} &= C - \alpha_i - \nu_i \quad \forall 1 \leq i \leq L \\ \frac{\partial\mathcal{L}}{\partial\xi_i} &= C^* - \alpha_i - \nu_i \quad \forall L+1 \leq i \leq L+2U\,.\end{aligned}$$

For simplifying the notation, we now define an extra special example $\boldsymbol{x}_0$ in an implicit manner:

$$\Phi(\boldsymbol{x}_0) = \frac{1}{U}\sum_{i=L+1}^{L+U}\Phi(\boldsymbol{x}_i)\,,$$

and we set $y_0 = 1$ and $\beta_0 = 0$. Setting the derivatives to zero gives us

$$\boldsymbol{w} = \sum_{i=0}^{L+2U} y_i(\alpha_i-\beta_i)\Phi(\boldsymbol{x}_i) \tag{22}$$

and

$$\sum_{i=0}^{L+2U} y_i(\alpha_i-\beta_i) = 0 \tag{23}$$

and also

$$C - \alpha_i - \nu_i = 0 \quad \forall 1 \leq i \leq L, \quad C^* - \alpha_i - \nu_i \quad \forall L+1 \leq i \leq L+2U\,. \tag{24}$$

In order to find the minimum of the minimization problem (12) we want to find a saddle point of the Lagrangian (21), as in classical SVMs methods. Substituting (22), (23) and (24) into the

Lagrangian (21) yields the following maximization problem

$$\begin{aligned}\arg\max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{i,j=0}^{L+2U} y_i y_j (\alpha_i - \beta_i)(\alpha_j - \beta_j)\Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j) \\ & + \sum_{i=1}^{L+2U} \alpha_i + \alpha_0 \left(\frac{1}{L}\sum_{i=1}^{L} y_i\right)\end{aligned} \tag{25}$$

under the constraints

$$\begin{aligned} 0 \le \alpha_i \le C \quad & \forall 1 \le i \le L \\ 0 \le \alpha_i \le C^* \quad & \forall L+1 \le i \le L+2U \\ \textstyle\sum_{i=0}^{L+2U} y_i(\alpha_i - \beta_i) = 0 & . \end{aligned} \tag{26}$$

The parameter $\boldsymbol{w}$ is then given by (22) and b is obtained using one of the following Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned} \alpha_0 \neq 0 &\implies \frac{1}{U}\sum_{i=L+1}^{L+U} [\boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i) + b] = \frac{1}{L}\sum_{i=1}^{L} y_i \\ \forall 1 \le i \le L,\ 0 < \alpha_i < C &\implies y_i[\boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i) + b] = 1 \\ \forall L+1 \le i \le L+2U,\ 0 < \alpha_i < C^* &\implies y_i[\boldsymbol{w}\cdot\Phi(\boldsymbol{x}_i) + b] = 1 \end{aligned}$$

If we define $\zeta_i = y_i$ for $1 \le i \le L+2U$ and $\zeta_0 = \frac{1}{L}\sum_{i=1}^{L} y_i$, and consider the kernel matrix K such that

$$K_{ij} = \Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j),$$

and we perform the substitution

$$\tilde{\alpha}_i = y_i(\alpha_i - \beta_i),$$

then we can rewrite the maximization problem (25) under the constraints (26) as the following

$$\arg\max_{\tilde{\boldsymbol{\alpha}}} \quad \boldsymbol{\zeta}\cdot\tilde{\boldsymbol{\alpha}} - \frac{1}{2}\tilde{\boldsymbol{\alpha}}^{\mathrm{T}} K \tilde{\boldsymbol{\alpha}}$$

under the constraints

$$\begin{aligned} 0 \le y_i\tilde{\alpha}_i \le C \quad & \forall 1 \le i \le L \\ -\beta_i \le y_i\tilde{\alpha}_i \le C^* - \beta_i \quad & \forall L+1 \le i \le L+2U \\ \textstyle\sum_{i=0}^{L+2U} \tilde{\alpha}_i = 0 & . \end{aligned} \tag{27}$$

Obviously this optimization problem is very close to an SVM optimization problem. It is thus possible to optimize it with a standard optimizer for SVMs. Note that only the bounds in (27) on the $\tilde{\alpha}_i$ have to be adjusted after each update of $\boldsymbol{\beta}$.

References

M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. In *Advances in Neural Information Processing Systems*. MIT Press, 2002.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 12*, pages 368–374. MIT Press, Cambridge, MA, 1998.

T. De Bie and N. Cristianini. Convex methods for transduction. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, May 2005. http://jmlr.csail.mit.edu/papers/v6/bordes05a.html.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

L. Bottou, C. Cortes, and V. Vapnik. On the effective VC dimension. Technical Report bottou-effvc.ps.Z, Neuroprose (ftp://archive.cse.ohio-state.edu/pub/neuroprose), 1994.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. *Neural Information Processing Systems 15*, 2002.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 201–208, New York, NY, USA, 2006. ACM Press.

P. Derbeko, R. El-Yaniv, and R. Meir. Explicit learning curves for transduction and application to clustering and compression algorithms. *Journal of Artificial Intelligence Research*, 22:117–142, 2004.

G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. In *Optimisation Methods and Software*, pages 1–14. Kluwer Academic Publishers, Boston, 2001.

T. Graepel, R. Herbrich, and K. Obermayer. Bayesian transduction. In *Advances in Neural Information Processing Systems 12, NIPS*, pages 456–462, 2000.

T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*. The MIT Press, 1999a.

T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning, ICML*, 1999b.

S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6:341–361, 2005.

S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Comp.*, 15:2667–1689, 2003.

N. Krause and Y. Singer. Leveraging the margin more carefully. In *International Conference on Machine Learning, ICML*, 2004.

N. D. Lawrence and M. I. Jordan. Semi-supervised learning via gaussian processes. In *Advances in Neural Information Processing Systems, NIPS*. MIT Press, 2005.

H. A. Le Thi. *Analyse numérique des algorithmes de l'optimisation D.C. Approches locales et globale. Codes et simulations numériques en grande dimension. Applications*. PhD thesis, INSA, Rouen, 1994.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In G.B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 9–50. Springer, 1998.

D. 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. URL `http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf`.

L. Mason, P. L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 38(3):243–255, 2000.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*. The MIT Press, 1999.

S.A.Nene, S.K.Nayar, and H.Murase. Columbia object image libary (coil-20). Technical Report CUS-005-96, Columbia Univ. USA, Febuary 1996.

B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *Proceedings ICANN97*, Springer Lecture Notes in Computer Science, page 583, 1997.

X. Shen, G. C. Tseng, X. Zhang, and W. H. Wong. On (psi)-learning. *Journal of the American Statistical Association*, 98(463):724–734, 2003.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *International Conference on Machine Learning, ICML*, 2005.

A. J. Smola, S. V. N. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. *NIPS*, 14, 2001a.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. *Neural Information Processing Systems 14*, 2001b.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, second edition, 1995.

V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Verlag Series in Statistics. Springer Verlag, 1982.

J. Weston, C. Leslie, D. Zhou, A. Elisseeff, and W. S. Noble. Cluster kernels for semi-supervised protein classification. *Advances in Neural Information Processing Systems 17*, 2003.

L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1537–1544. MIT Press, Cambridge, MA, 2005.

A. L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems, NIPS*, 2004.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *The Twentieth International Conference on Machine Learning*, pages 912–919, 2003.

Journal of Machine Learning Research 7 (2006) 1713–1741 Submitted 10/05; Revised 7/06; Published 8/06

Considering Cost Asymmetry in Learning Classifiers

Francis R. Bach FRANCIS.BACH@MINES.ORG
Centre de Morphologie Mathématique
Ecole des Mines de Paris
35, rue Saint Honoré
77300 Fontainebleau, France

David Heckerman HECKERMAN@MICROSOFT.COM
Eric Horvitz HORVITZ@MICROSOFT.COM
Microsoft Research
Redmond, WA 98052, USA

Editor: John Shawe-Taylor

Abstract

Receiver Operating Characteristic (ROC) curves are a standard way to display the performance of a set of binary classifiers for all feasible ratios of the costs associated with false positives and false negatives. For linear classifiers, the set of classifiers is typically obtained by training once, holding constant the estimated slope and then varying the intercept to obtain a parameterized set of classifiers whose performances can be plotted in the ROC plane. We consider the alternative of varying the asymmetry of the cost function used for training. We show that the ROC curve obtained by varying both the intercept and the asymmetry, and hence the slope, always outperforms the ROC curve obtained by varying only the intercept. In addition, we present a path-following algorithm for the support vector machine (SVM) that can compute efficiently the entire ROC curve, and that has the same computational complexity as training a single classifier. Finally, we provide a theoretical analysis of the relationship between the asymmetric cost model assumed when training a classifier and the cost model assumed in applying the classifier. In particular, we show that the mismatch between the step function used for testing and its convex upper bounds, usually used for training, leads to a provable and quantifiable difference around extreme asymmetries.

Keywords: support vector machines, receiver operating characteristic (ROC) analysis, linear classification

1. Introduction

Receiver Operating Characteristic (ROC) analysis has seen increasing attention in the recent statistics and machine-learning literature (Pepe, 2000; Provost and Fawcett, 2001; Flach, 2003). The ROC is a representation of choice for displaying the performance of a classifier when the costs assigned by end users to false positives and false negatives are not known at the time of training. For example, when training a classifier for identifying cases of undesirable unsolicited email, end users may have different preferences about the likelihood of a false negative and false positive. The ROC curve for such a classifier reveals the ratio of false negatives and positives at different probability thresholds for classifying an email message as unsolicited or normal email.

In this paper, we consider linear binary classification of points in an Euclidean space—noting that it can be extended in a straightforward manner to non-linear classification problems by using

Mercer kernels (Schölkopf and Smola, 2002). That is, given data $x \in \mathbb{R}^d$, $d \geqslant 1$, we consider classifiers of the form $f(x) = \text{sign}(w^\top x + b)$, where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are referred to as the *slope* and the *intercept*. To date, ROC curves have been usually constructed by training once, holding constant the estimated slope and varying the intercept to obtain the curve. In this paper, we show that, while that procedure appears to be the most practical thing to do, it may lead to classifiers with poor performance in some parts of the ROC curve.

The crux of our approach is that we allow the asymmetry of the cost function to vary, that is, we vary the ratio of the cost of a false positive and the cost of a false negative. For each value of the ratio, we obtain a different slope and intercept, each optimized for this ratio. In a naive implementation, varying the asymmetry would require a retraining of the classifier for each point of the ROC curve, which would be computationally expensive. In Section 3.1, we present an algorithm that can compute the solution of a support vector machine (SVM) (see, for example, Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) for all possible costs of false positives and false negatives, with the same computational complexity as obtaining the solution for only one cost function. The algorithm extends to asymmetric costs the algorithm of Hastie et al. (2005) and is based on path-following techniques that take advantage of the piecewise linearity of the path of optimal solutions. In Section 3.2, we show how the path-following algorithm can be used to obtain ROC curves. In particular, by allowing both the asymmetry and the intercept to vary, we can obtain better ROC curves than by methods that simply vary the intercept.

In Section 4, we provide a theoretical analysis of the relationship between the asymmetry of costs assumed in training a classifier and the asymmetry desired in its application. In particular, we show that, even in the population (*i.e.*, infinite sample) case, the use of a training loss function which is a convex upper bound on the true or testing loss function (a step function) creates classifiers with sub-optimal accuracy. We quantify this problem around extreme asymmetries for several classical convex-upper-bound loss functions, including the square loss and the *erf loss*, an approximation of the logistic loss based on normal cumulative distribution functions (also referred to as the "error function," and usually abbreviated as erf). The analysis is carried through for Gaussian and mixture of Gaussian class-conditional distributions (see Section 4 for more details). The main result of this analysis is that given an extreme user-defined testing asymmetry, the training asymmetry should almost always be chosen to be less extreme.

As we shall see, the consequences of the potential mismatch between the cost functions assumed in testing versus training underscore the value of using the algorithm that we introduce in Section 4.3. Even when costs are known (*i.e.*, when only one point on the ROC curve is needed), the classifier resulting from our approach, which builds the entire ROC curve, is never less accurate and can be more accurate than one trained with the known costs using a convex-upper-bound loss function. Indeed, we show in Section 4.3 that computing the entire ROC curve using our algorithm can lead to substantial gains over simply training once.

The paper is organized as follows: In Section 2, we give an introduction to the linear classification problem and review the ROC framework. Section 3 contains the algorithmic part of the paper, while Section 4 provide a theoretical analysis of the discrepancy between testing and training asymmetries, together with empirical results. This paper is an extended version of previously published work (Bach et al., 2005a).

2. Problem Overview

Given data $x \in \mathbb{R}^d$ and labels $y \in \{-1,1\}$, we consider linear classifiers of the form $f(x) = \text{sign}(w^\top x + b)$, where w is the *slope* of the classifier and b the *intercept*. A classifier is determined by the parameters $(w,b) \in \mathbb{R}^{d+1}$. In Section 2.1, we introduce notation and definitions; in Section 2.2, we lay out the necessary concepts of ROC analysis, while in Section 2.3, we describe how these classifiers and ROC curves are typically obtained from data.

2.1 Asymmetric Cost and Loss Functions

Positive (resp. negative) examples are those for which $y = 1$ (resp. $y = -1$). The two types of misclassification, false positives and false negatives, are assigned two different costs. We let C_+ denote the cost of a false negative and C_- the cost of a false positive. The total *expected* cost is equal to

$$R(C_+, C_-, w, b) = C_+ P\{w^\top x + b < 0,\ y = 1\} + C_- P\{w^\top x + b \geqslant 0,\ y = -1\}.$$

In the context of large margin classifiers (see, for example, Bartlett et al., 2004), the expected cost is usually expressed in terms of the *0–1 loss function*; indeed, if we let $\phi_{0-1}(u) = 1_{u<0}$ be the 0–1 loss, we can write the expected cost as

$$R(C_+, C_-, w, b) = C_+ E\{1_{y=1}\phi_{0-1}(w^\top x + b)\} + C_- E\{1_{y=-1}\phi_{0-1}(-w^\top x - b)\},$$

where E denotes the expectation with respect to the joint distribution of (x,y).

The expected cost defined using the 0–1 loss is the cost that end users are usually interested in during the use of the classifier, while the other cost functions that we define below are used solely for training purposes. The convexity of these cost functions makes learning algorithms convergent without local minima, and leads to attractive asymptotic properties (Bartlett et al., 2004).

A traditional set-up for learning linear classifiers from labeled data is to consider a convex upper bound ϕ on the 0–1 loss ϕ_{0-1}, and to use the expected ϕ-cost:

$$R_\phi(C_+, C_-, w, b) = C_+ E\{1_{y=1}\phi(w^\top x + b)\} + C_- E\{1_{y=-1}\phi(-w^\top x - b)\}.$$

We refer to the ratio $C_+/(C_- + C_+)$ as the *asymmetry*. We shall use *training asymmetry* to refer to the asymmetry used for training a classifier using a ϕ-cost, and the *testing asymmetry* to refer to the asymmetric cost characterizing the testing situation (reflecting end user preferences) with the actual cost based on the 0–1 loss. In Section 4, we will show that these may be different in the general case.

We shall consider several common loss functions. Some of the loss functions (square loss, hinge loss) lead to attractive computational properties, while others (square loss, erf loss) are more amenable to theoretical manipulations (see Figure 1 for the plot of the loss functions, as they are commonly used and defined below[1]):

- **square loss** : $\phi_{sq}(u) = \frac{1}{2}(u-1)^2$; the classifier is equivalent to linear regression on y,
- **hinge loss** : $\phi_{hi}(u) = \max\{1-u, 0\}$; the classifier is the support vector machine (Shawe-Taylor and Cristianini, 2004),

1. Note that by rescaling, each of these loss functions can be made to be an upper bound on the 0–1 loss which is tight at zero.

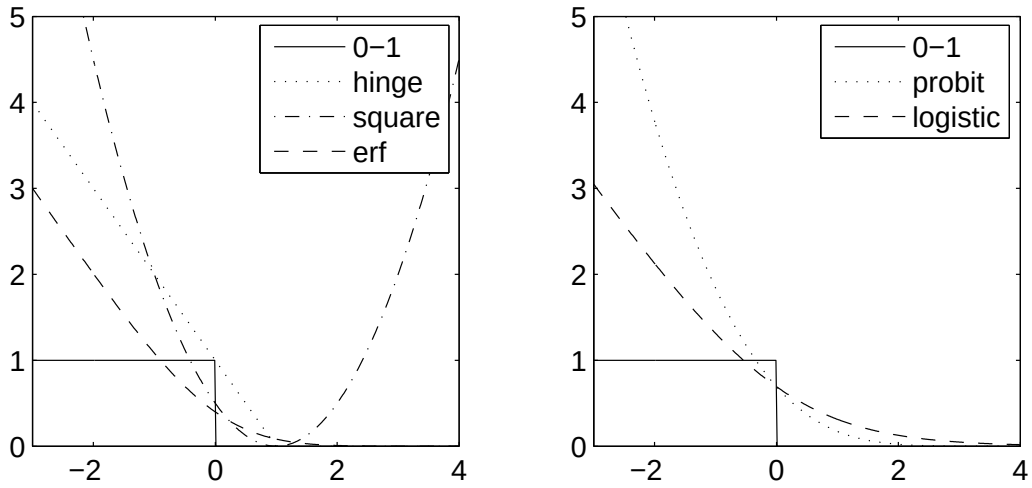


Figure 1: Loss functions. Left: plain: 0–1 loss; dotted: hinge loss, dashed: erf loss, dash-dotted: square loss. Right: plain: 0–1 loss, dotted: probit loss, dashed: logistic loss.

- **erf loss** : $\phi_{erf}(u) = [u\psi(u) - u + \psi'(u)]$, where ψ is the cumulative distribution of the standard normal distribution, that is : $\psi(v) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{v} e^{-t^2/2} dt$, and $\psi'(v) = \frac{1}{\sqrt{2\pi}} e^{-v^2/2}$. The erf loss can be used to provide a good approximation of the *logistic loss* $\phi_{log}(u) = \log(1+e^{-u})$ as well as its derivative, and is amenable to closed-form computations for Gaussians and mixture of Gaussians (see Section 4 for more details). Note that the erf loss is different from the *probit loss* $-\log\psi(u)$, which leads to probit regression (Hastie et al., 2001).

2.2 ROC Analysis

The aim of ROC analysis is to display in a single graph the performance of classifiers for all possible costs of misclassification. In this paper, we consider sets of classifiers $f_\gamma(x)$, parameterized by a variable $\gamma \in \mathbb{R}$ (γ can either be the intercept or the training asymmetry).

For a classifier $f(x)$, we can define a point (u,v) in the "ROC plane," where u is the proportion of false positives $u = P(f(x) = 1 | y = -1)$, and v is the proportion of true positives $v = P(f(x) = 1 | y = 1)$. When γ is varied, we obtain a curve in the ROC plane, the ROC curve (see Figure 2 for an example). Whether γ is the intercept or the training asymmetry, the ROC curve always passes through the point $(0,0)$ and $(1,1)$, which corresponds to classifying all points as negative (resp. positive).

The *upper convex envelope* of the curve is the set of optimal ROC points that can be achieved by the set of classifiers; indeed, if a point in the envelope is not one of the original points, it must lie in a segment between two points $(u(\gamma_0), v(\gamma_0))$ and $(u(\gamma_1), v(\gamma_1))$, and all points in a segment between two classifiers can always be attained by choosing randomly between the two classifiers. Note that this classifier itself is not a linear classifier; its performance, defined by given true positive and false positive rates, can only be achieved by a mixture of two linear classifiers. However, if the user is

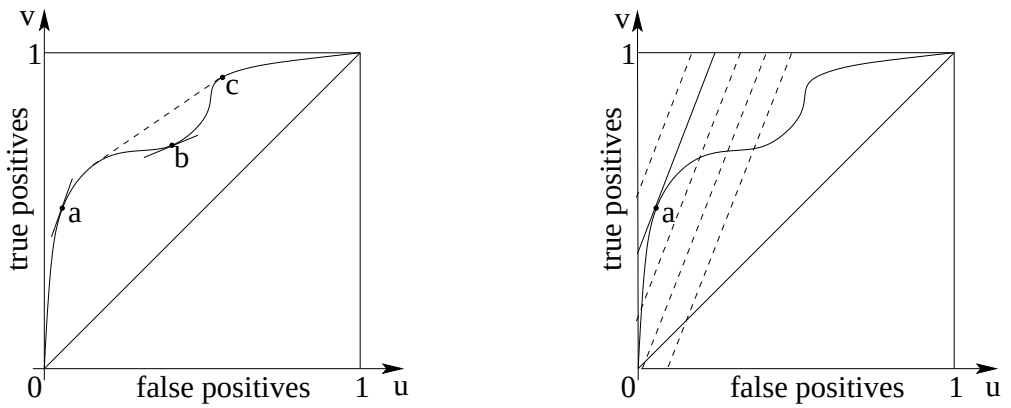


Figure 2: Left: ROC curve: (plain) regular ROC curve; (dashed) convex envelope. The points *a* and *c* are ROC-consistent and the point *b* is ROC-inconsistent. Right: ROC curve and dashed equi-cost lines: All lines have direction (p_+C+, p_-C_-), the plain line is optimal and the point "a" is the optimal classifier.

only interested in the testing cost, which is a weighted linear combination of the true positive and false positive rates, the lowest testing cost is always achieved by both of these two classifiers (see Section 4.3 for an example of such a situation).

Denoting $p_+ = P(y=1)$ and $p_- = P(y=-1)$, the expected (C_+, C_-)-cost for a classifier (u,v) in the ROC space, is simply $p_+C_+(1-v) + p_-C_-u$, and thus optimal classifiers for the (C_+, C_-)-cost can be found by looking at lines of slope that are normal to the direction $(p_-C_-, -p_+C_+)$, which intersects the ROC curve and are as close as the point $(0,1)$ as possible (see Figure 2).

A point $(u(\gamma), v(\gamma))$ is said to be *ROC-consistent* if it lies on the upper convex envelope; In this case, the tangent direction $(du/d\gamma, dv/d\gamma)$ defines a cost $(C_+(\gamma), C_-(\gamma))$ for which the classifier is optimal (for the testing cost, which is defined using the 0–1 loss). The condition introduced earlier, namely that $(du/d\gamma, dv/d\gamma)$ is normal to the direction $(p_-C_-, -p_+C_+)$, leads to:

$$p_-C_-\frac{du}{d\gamma}(\gamma) - p_+C_+\frac{dv}{d\gamma}(\gamma) = 0.$$

The *optimal testing asymmetry* $\beta(\gamma)$ defined as the ratio $\frac{C_+(\gamma)}{C_+(\gamma)+C_-(\gamma)}$, is thus equal to

$$\beta(\gamma) \triangleq \frac{C_+(\gamma)}{C_+(\gamma)+C_-(\gamma)} = \frac{1}{1+\frac{p_+}{p_-}\frac{dv}{d\gamma}(\gamma)/\frac{du}{d\gamma}(\gamma)}. \quad (1)$$

If a point $(u(\gamma), v(\gamma))$ is ROC-inconsistent, then the quantity $\beta(\gamma)$ has no meaning, and such a classifier is generally useless, because, for all settings of the misclassification cost, that classifier can be outperformed by the other classifiers or a combination of classifiers. See Figure 2 for examples of ROC-consistent and ROC-inconsistent points.

In Section 4, we relate the optimal asymmetry of cost in the testing or eventual use of a classifier in the real world, to the asymmetry of cost used to train that classifier; in particular, we show that

they differ and quantify this difference for extreme asymmetries (*i.e.*, close to the corner points $(0,0)$ and $(1,1)$). This analysis highlights the value of generating the entire ROC curve, even when only one point is needed, as we will present in Section 4.3.

Handling ROC surfaces In this paper, we will also consider varying both the asymmetry of the cost function and the intercept, leading to a set of points in the ROC plane parameterized by two real values. Although the concept of ROC-consistency could easily be extended to ROC surfaces, for simplicity we do not consider it here. In all our experiments, those ROC surfaces are reduced to curves by computing their convex upper envelopes.

2.3 Learning From Data

Given n labeled data points (x_i,y_i), $i=1,\ldots,n$, the *empirical cost* is equal to

$$\widehat{R}(C_+,C_-,w,b)=\frac{C_+}{n}\#\{y_i(w^\top x_i+b)<0,y_i=1\}+\frac{C_-}{n}\#\{y_i(w^\top x_i+b)<0,y_i=-1\},$$

where $\#A$ denotes the cardinality of the set A. The *empirical* ϕ*-cost* is equal to

$$\widehat{R}_\phi(C_+,C_-,w,b)=\frac{C_+}{n}\sum_{i\in I_+}\phi(y_i(w^\top x_i+b))+\frac{C_-}{n}\sum_{i\in I_-}\phi(y_i(w^\top x_i+b)),$$

where $I_+=\{i,y_i=1\}$ and $I_-=\{i,y_i=-1\}$. When learning a classifier from data, a classical setup is to minimize the sum of the *empirical* ϕ*-cost* and a regularization term $\frac{1}{2n}||w||^2$, that is, to minimize $\widehat{J}_\phi(C_+,C_-,w,b)=\widehat{R}_\phi(C_+,C_-,w,b)+\frac{1}{2n}||w||^2$.

Note that the objective function is no longer homogeneous in (C_+,C_-); the sum C_++C_- is referred to as the total amount of regularization. Thus, two end-user-defined parameters are needed to train a linear classifier: the *total amount of regularization* $C_++C_-\in\mathbb{R}^+$, and the *asymmetry* $\frac{C_+}{C_++C_-}\in[0,1]$. In Section 3.1, we show how the minimum of $\widehat{J}_\phi(C_+,C_-,w,b)$, with respect to w and b, can be computed efficiently for the hinge loss, for many values of (C_+,C_-), with a computational cost that is within a constant factor of the computational cost of obtaining a solution for one value of (C_+,C_-).

Building an ROC curve from data If a sufficiently large validation set is available, we can train on the training set and use the empirical distribution of the validation data to plot the ROC curve. If sufficient validation data is not available, then we can use several (typically 10 or 25) random splits of the data and average scores over those splits to obtain the ROC scores. We can also use this approach to obtain confidence intervals (Flach, 2003).

3. Building ROC Curves for the SVM

In this section, we present an algorithm to compute ROC curves for the SVM that explores the two-dimensional space of cost parameters (C_+,C_-) efficiently. We first show how to obtain optimal solutions of the SVM without solving the optimization problems many times for each value of (C_+,C_-). This method generalizes the results of Hastie et al. (2005) to the case of asymmetric cost functions. We then describe how the space (C_+,C_-) can be appropriately explored and how ROC curves can be constructed.

3.1 Building Paths of Classifiers

Given n data points x_i, $i = 1, \ldots, n$ which belong to $\mathbb{R}^d$, and n labels $y_i \in \{-1, 1\}$, minimizing the regularized empirical hinge loss is equivalent to solving the following convex optimization problem (Schölkopf and Smola, 2002):

$$\min_{w,b,\xi} \sum_i C_i \xi_i + \frac{1}{2}||w||^2 \quad \text{such that} \quad \forall i,\ \xi_i \geqslant 0,\ \xi_i \geqslant 1 - y_i(w^\top x_i + b),$$

where $C_i = C_+$ if $y_i = 1$ and $C_i = C_-$ if $y_i = -1$.

Optimality conditions and dual problems We know derive the usual Karush-Kuhn-Tucker (KKT) optimality conditions (Boyd and Vandenberghe, 2003). The Lagrangian of the problem is (with dual variables $\alpha, \beta \in \mathbb{R}^n_+$):

$$L(w, b, \xi, \alpha, \beta) = \sum_i C_i \xi_i + \frac{1}{2}||w||^2 + \sum_i \alpha_i (1 - y_i(w^\top x_i + b)) - \xi_i) - \sum_i \beta_i \xi_i.$$

The derivatives with respect to the primal variables are

$$\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i\ , \quad \frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i\ , \quad \frac{\partial L}{\partial \xi_i} = C_i - \alpha_i - \beta_i\ ,$$

The first set of KKT conditions corresponds to the nullity of the Lagrangian derivatives with respect to the primal variables, that is:

$$w = \sum_i \alpha_i y_i x_i\ , \quad \sum_i \alpha_i y_i = \alpha^\top y = 0\ , \quad \forall i,\ C_i = \alpha_i + \beta_i. \tag{2}$$

The slackness conditions are

$$\forall i,\ \alpha_i(1 - \xi_i + y_i(w^\top x_i + b)) = 0 \ \text{ and } \ \beta_i \xi_i = 0. \tag{3}$$

Finally the dual problem can be obtained by computing the minimum of the Lagrangian with respect to the primal variables. If we let denote K the $n \times n$ Gram matrix of inner products, that is, defined by $K_{ij} = x_i^\top x_j$, and $\widetilde{K} = \mathrm{Diag}(y) K \mathrm{Diag}(y)$, the dual problem is:

$$\max_{\alpha \in \mathbb{R}^n} -\frac{1}{2}\alpha^\top \widetilde{K} \alpha + 1^\top \alpha \quad \text{such that} \quad \alpha^\top y = 0,\ \ \forall i,\ 0 \leqslant \alpha_i \leqslant C_i.$$

In the following, we only consider primal variables (w, b, ξ) and dual variables (α, β) that verify the first set of KKT conditions (2), which implies that w and β are directly determined by α.

Piecewise affine solutions Following Hastie et al. (2005), for an optimal set of primal-dual variables (w, b, α), we can separate data points in three disjoint sets, depending on the values of $y_i(w^\top x_i + b) = (\widetilde{K}\alpha)_i + b y_i$.

$$\begin{array}{lllll}
\text{Margin :} & \mathcal{M} & = \{i, & y_i(w^\top x_i + b) = 1 & \}, \\
\text{Left of margin :} & \mathcal{L} & = \{i, & y_i(w^\top x_i + b) < 1 & \}, \\
\text{Right of margin :} & \mathcal{R} & = \{i, & y_i(w^\top x_i + b) > 1 & \}.
\end{array}$$

Because of the slackness conditions (3), the sign of $y_i(w^\top x_i + b) - 1$ is linked with the location of α_i in the interval $[0, C_i]$. Indeed we have:

$$\begin{array}{lrcl} \text{Left of margin :} & i \in \mathcal{L} & \Rightarrow & \alpha_i = C_i, \\ \text{Right of margin :} & i \in \mathcal{R} & \Rightarrow & \alpha_i = 0. \end{array}$$

In the optimization literature, the sets $\mathcal{L}$, $\mathcal{R}$ and $\mathcal{M}$ are usually referred to as *active sets* (Boyd and Vandenberghe, 2003; Scheinberg, 2006). If the sets $\mathcal{M}$, $\mathcal{L}$ and $\mathcal{R}$ are known, then α, b are optimal if and only if:

$$\begin{array}{l} \forall i \in \mathcal{L}, \alpha_i = C_i \\ \forall i \in \mathcal{R}, \alpha_i = 0 \\ \forall i \in \mathcal{M}, (\widetilde{K}\alpha)_i + b y_i = 1 \\ \alpha^\top y = 0. \end{array}$$

This is a linear system with as many equations as unknowns (*i.e.*, $n+1$). The real unknowns (not clamped to C_i or 0) are $\alpha_{\mathcal{M}}$ and b, and the smaller system is (z_A denotes the vector z reduced to its components in the set A and $K_{A,B}$ denotes the submatrix of K with indices in A and B):

$$\begin{array}{rcl} \widetilde{K}_{\mathcal{M},\mathcal{M}} \alpha_{\mathcal{M}} + b y_{\mathcal{M}} & = & 1_{\mathcal{M}} - \widetilde{K}_{\mathcal{M},\mathcal{L}} C_{\mathcal{L}} \\ y_{\mathcal{M}}^\top \alpha_{\mathcal{M}} & = & -y_{\mathcal{L}}^\top C_{\mathcal{L}}, \end{array}$$

whose solution is affine in $C_{\mathcal{L}}$ and thus in C.

Consequently, for known active sets, the solution is affine in C, which implies that the optimal variables (w, α, b) are piecewise affine continuous functions of the vector C. In our situation, C depends linearly on C_+ and C_-, and thus the path is piecewise affine in (C_+, C_-).

Following a path The active sets (and thus the linear system) remain the same as long as all constraints defining the active sets are satisfied, that is, (a) $y_i(w^\top x_i + b) - 1$ is positive for all $i \in \mathcal{R}$ and negative for all $i \in \mathcal{L}$, and (b) for each $i \in \mathcal{M}$, α_i remains between 0 and C_i. This defines a set of linear inequalities in (C_+, C_-). The facets of the polytope defined by these inequalities can always be found in linear time in n, if efficient convex hull algorithms are used (Avis et al., 1997). However, when we only follow a straight line in the (C_+, C_-)-space, the polytope is then a segment and its extremities are trivial to find (also in linear time $O(n)$).

Following Hastie et al. (2005), if a solution is known for one value of (C_+, C_-), we can follow the path along a line, by monitoring which constraints are violated at the boundary of the polytope that defines the allowed domain of (C_+, C_-) for the given active sets.

Numerical issues Several numerical issues have to be solved before the previous approach can be made efficient and stable. Some of the issues directly follow the known issues of the simplex method for linear programming (which is itself an active set method) (Maros, 2002).

- *Path initialization*: In order to easily find a point of entry into the path, we look at situations when all points are in $\mathcal{L}$, that is, $\forall i$, $\alpha_i = C_i$. In order to verify $\alpha^\top y = 0$, this implies that $\sum_{i \in I_+} C_i = \sum_{i \in I_-} C_i$, that is, this means $C_+ n_+ = C_- n_-$ (where $n_+ = |I_+|$ and $n_- = |I_-|$ are the

number of positive and negative training examples), which we now assume. The active sets remain unchanged as long as $\forall i, y_i(w^\top x_i + b) \leqslant 1$, that is:

$$\forall i \in I_+ \quad , \quad b \leqslant 1 - C_+ \left((\widetilde{K}\delta_+)_i + \frac{n_+}{n_-}(\widetilde{K}\delta_-)_i \right)$$
$$\forall i \in I_- \quad , \quad b \geqslant -1 + C_+ \left((\widetilde{K}\delta_+)_i + \frac{n_+}{n_-}(\widetilde{K}\delta_-)_i \right),$$

where δ_+ (resp. δ_-) is the indicator vector of the positive (resp. negative) examples.

Let us define the following two maxima: $m_+ = \max_{i \in I_+} \left((\widetilde{K}\delta_+)_i + \frac{n_+}{n_-}(\widetilde{K}\delta_-)_i \right)$ (attained at i_+) and $m_- = \max_{i \in I_-} \left((\widetilde{K}\delta_+)_i + \frac{n_+}{n_-}(\widetilde{K}\delta_-)_i \right)$ (attained at i_-). The previous conditions are equivalent to

$$-1 + C_+ m_- \leqslant b \leqslant 1 - C_+ m_+.$$

Thus, all points are in $\mathcal{L}$ as long as $C_+ \leqslant 2/(m_- + m_+)$, and when this is verified, b is undetermined, between $-1 + C_+ m_-$ and $1 - C_+ m_+$. At the boundary point $C_+ = 2/(m_- + m_+)$; then both i_+ and i_- are going from $\mathcal{L}$ to $\mathcal{M}$.

Since we vary both C_+ and C_- we can start by following the line $C_+ n_+ = C_- n_-$ and we are thus able to avoid to solve a quadratic program to enter the path, as is done by Hastie et al. (2005) when the data sets are not perfectly balanced.

- *Switching between active sets*: Indices can go from $\mathcal{L}$ to $\mathcal{M}$, $\mathcal{R}$ to $\mathcal{M}$, or $\mathcal{M}$ to $\mathcal{R}$ or $\mathcal{L}$. Empirically, when we follow a line in the plane (C_+, C_-), most points go from $\mathcal{L}$ to $\mathcal{R}$ through $\mathcal{M}$ (or from $\mathcal{R}$ to $\mathcal{L}$ through $\mathcal{M}$), with a few points going back and forth; this implies that empirically the number of kinks when following a line in the (C_+, C_-) plane is $O(n)$.

- *Efficient implementation of linear system*: The use of Cholesky updating and downdating is necessary for stability and speed (Golub and Van Loan, 1996).

- *Computational complexity*: Following the analysis of Hastie et al. (2005), if m is the maximum number of points in $\mathcal{M}$ along the path and p is the number of path following steps, then the algorithm has complexity $O(m^2 p + pmn)$. Empirically, the number of steps is $O(n)$ for one following one line in the (C_+, C_-) plane, so the empirical complexity is $O(m^2 n + mn^2)$.

  It is worth noting that the complexity of obtaining one path of classifiers across one line is roughly the same as obtaining the solution for only one SVM using classical techniques such as sequential minimal optimization (Platt, 1998).

Classification with kernels The path following algorithm developed in this section immediately applies to non-linear classification, by replacing the Gram matrix defined as $K_{ij} = x_i^\top x_j$, by any *kernel matrix* K defined as $K_{ij} = k(x_i, x_j)$, where k is a positive semi-definite kernel function (Shawe-Taylor and Cristianini, 2004).

3.2 Constructing the ROC Curve

Given the tools of Section 3.1, we can learn paths of linear classifiers from data. In this section, we present an algorithm to build ROC curves from the paths. We do this by exploring relevant parts of the (C_+, C_-) space, selecting the best classifiers among the ones that are visited.

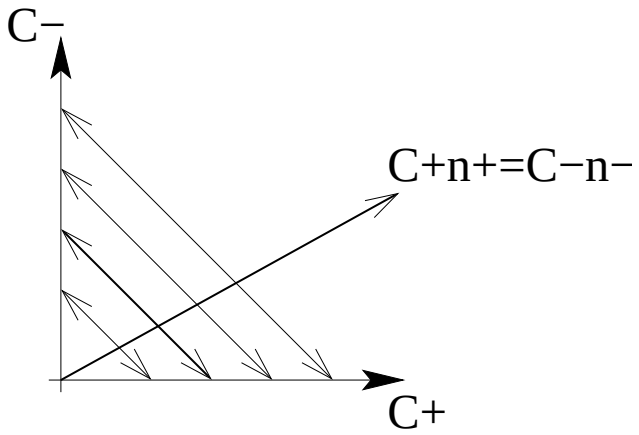


Figure 3: Lines in the (C_+, C_-)-space. The line $C_+ n_+ = C_- n_-$ is always followed first; then several lines with constant $C_+ + C_-$ are followed in parallel, around the optimal line for the validation data (bold curve).

We assume that we have two separate data sets, one for training and one for testing. This approach generalizes to cross validation settings in a straightforward manner.

Exploration phase In order to start the path-following algorithm, we need to start at $C_+ = C_- = 0$ and follow the line $C_+ n_+ = C_- n_-$. We follow this line up to a large upper bound on $C_+ + C_-$. For all classifiers along that line, we compute a misclassification cost on the testing set, with given asymmetry (C_+^0, C_-^0) (as given by the user, and usually, but not necessarily, close to a point of interest in the ROC space). We then compute the best performing pair (C_+^1, C_-^1) and we select pairs of the form (rC_+^1, rC_-^1), where r belongs to a set R of the type $R = \{1, 10, 1/10, 100, 1/100, \dots\}$. The set R provides further explorations for the total amount of regularization $C_+ + C_-$.

Then, for each r, we follow the paths of direction $(1,-1)$ and $(-1,1)$ starting from the point (rC_+^1, rC_-^1). Those paths have a fixed total amount of regularization but vary in asymmetry. In Figure 3, we show all of lines that are followed in the (C_+, C_-) space.

Selection phase After the exploration phase, we have $|R|+1$ different lines in the (C_+, C_-) space: the line $C_- n_- = C_+ n_+$, and the $|R|$ lines $C_+ + C_- = r(C_+^1 + C_-^1), r \in R$. For each of these lines, we know the optimal solution (w, b) for any cost settings on that line. The line $C_- n_- = C_+ n_+$ is used for computational purposes (*i.e.*, to enter the path), so we do not use it for testing purposes.

For each of the R lines in the (C_+, C_-)-plane, we can build the three following ROC curves, as shown in the top of Figure 4 for a simple classification problem involving mixtures of Gaussians:

- *Varying intercept*: We extract the slope w corresponding to the best setting $(C_+^1 + C_-^1)$, and vary the intercept b from $-\infty$ to ∞. This is the traditional method for building an ROC curve for an SVM.

- *Varying asymmetry*: We only consider the line $C_+ + C_- = C_+^1 + C_-^1$ in the (C_+, C_-)-plane; the classifiers that are used are the optimal solutions of the convex optimization problem. Note that for each value of the asymmetry, we obtain a different value of the slope and the intercept.

- *Varying intercept and asymmetry*: For each of the points on the R lines in the (C_+, C_-)-plane, we discard the intercept b and keep the slope w obtained from the optimization problem; we then use all possible intercept values; this leads to R two-dimensional surfaces in the ROC plane. We then compute the convex envelope of these, to obtain a single curve.

Since all classifiers obtained by varying only the intercept (resp. the asymmetry) are included in the set used for varying both the intercept and the asymmetry, the third ROC curve always outperforms the first two curves (*i.e.*, it is always closer to the top left corner). This is illustrated in Figure 4. Once ROC curves are obtained for each of the R lines, we can combine them by taking their upper convex envelope to obtain ROC curves obtained from even larger sets of classifiers, which span several total amounts of regularization. Note that the ROC scores that we consider are obtained by using held out testing data or cross-validation scores; thus by considering larger sets of classifiers, taking upper convex envelopes will always lead to better performance for these scores, which are only approximations of the expected scores on unseen data, and there is a potential risk of overfitting the cross-validation scores (Ng, 1997). In Section 4.3, we present empirical experiments showing that by carefully selecting classifiers based on held out data or cross-validation scores, we are not overfitting.

Intuitively, the ROC curve obtained by varying the asymmetry should be better than the ROC generated by varying the intercept because, for each point, the slope of the classifier is optimized. Empirically, this is generally true, but is not always the case, as displayed in the top of Figure 4. In the bottom of Figure 4, we show the same ROC curve, but in the infinite sample case, where the solution of the SVM was obtained by working directly with densities, separately for each training asymmetry. The ROC curve in the infinite sample case exhibit the same behavior than in the finite sample case, hinting that this behavior is not a small sample effect.

Another troubling fact is that the ROC curve obtained by varying asymmetry, exhibits strong concavities, that is, there are many ROC-inconsistent points: for those points, the solution of the SVM with the corresponding asymmetry is far from being the best linear classifier when performance is measured with the same asymmetry but with the exact 0–1 loss. In addition, even for ROC-consistent points, the training asymmetry and the testing asymmetry differ. In the next section, we analyze why they may differ and characterize their relationships in some situations.

4. Training Versus Testing Asymmetry

We observed in Section 3.2 that the training cost asymmetry can differ from the testing asymmetry. In this section, we analyze their relationships more closely for the population (*i.e.*, infinite sample) case. Although a small sample effect might alter some of the results presented in this section, we argue that most of the discrepancies come from using a convex surrogate to the $0-1$ loss.

Recent results (Bartlett et al., 2004; Zhang, 2004) have shown that using a convex surrogate may lead, under certain conditions, to the *Bayes optimal*, that is, the (usually non-linear) classifier with minimal expected cost. However, those conditions are usually not met in practice, since they essentially implies that the class of functions over which the expected surrogate cost is minimized contains the Bayes optimal classifier. In many cases, the Bayes optimal classifier does not belong

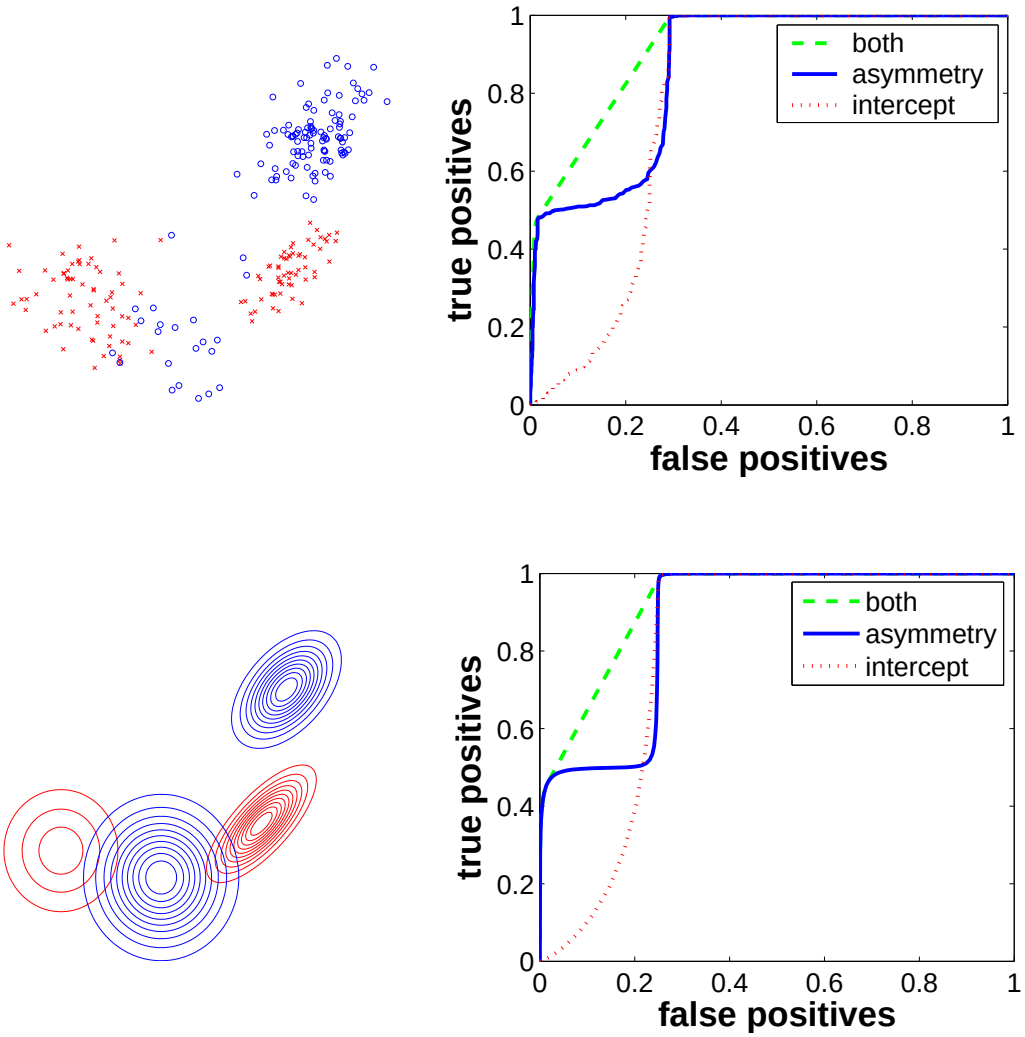


Figure 4: Two examples of ROC curves for bimodal class conditional densities, varying intercept (dotted), varying asymmetry (plain) and varying both (dashed). Top: ROC curves obtained from 10 random splits, using the data shown on the left side (one class is plotted as circles, the other one as crosses); Bottom: ROC curves obtained from corresponding population densities.

to the class of functions, and the best that can be hoped for is to obtain the classifier with minimal expected cost within the class of functions which is considered. Using a surrogate does not always lead to this classifier and the results of this section illustrate and quantify this fact in the context of varying cost asymmetries.

Since we are using population densities, we can get rid of the regularization term and thus only the asymmetry will have an influence on the results, that is, we can restrict ourselves to $C_+ + C_- = 1$. We let $\gamma = C_+/(C_+ + C_-) = C_+$ denote the training asymmetry. For a given training asymmetry γ and a loss function ϕ, in Section 2.2, we defined the *optimal testing asymmetry* $\beta(\gamma)$ for the training asymmetry γ as the testing asymmetry for which the classifier obtained by training with asymmetry γ is optimal. In this section, we will refer to the $\beta(\gamma)$ simply as the *testing asymmetry*.

Although a difference might be seen empirically for all possible asymmetries, we analyze the relationship between the testing cost asymmetry and training asymmetry in cases of extreme asymmetries, that is, in the ROC framework, close to the corner points $(0,0)$ and $(1,1)$. We prove that, depending on the class conditional densities, there are three possible different regimes for extreme asymmetries and that under two of these regimes, the training asymmetry should be chosen less extreme. We also provide, under certain conditions, a simple test that can determine the regime given class conditional densities.

In this section, we choose class conditional densities that are either Gaussian or a mixture of Gaussians, because (a) any density can be approximated as well as desired by mixtures of Gaussians (Hastie et al., 2001), and (b) for the square loss and the erf loss, they enable closed-form calculations that lead to Taylor expansions.

4.1 Optimal Solutions for Extreme Cost Asymmetries

We assume that the class conditional densities are mixtures of Gaussians, that is, the density of positive (resp. negative) examples is a mixture of k_+ Gaussians, with means μ_+^i and covariance matrix Σ_+^i, and mixing weights π_+^i, $i \in \{1,\ldots,k_+\}$ (resp. k_- Gaussians, with means μ_-^i and covariance matrix Σ_-^i, and mixing weights π_-^i, $i \in \{1,\ldots,k_-\}$). We denote M_+ (resp. M_-) the $d \times k_+$ (resp. $d \times k_-$) the matrix of means.

We denote p_+ and p_- as the marginal class densities, $p_+ = P(y=1)$, $p_- = P(y=-1)$. We assume that all mixing weights $\pi_\pm^i$ are strictly positive, that all covariance matrices $\Sigma_\pm^i$ have full rank, and that $p_+, p_- \in (0,1)$.

In the following sections, we provide Taylor expansions of various quantities around the null training asymmetry $\gamma = 0$. The quantities trivially extend around the reverse asymmetry $\gamma = 1$. We focus on two losses, the square loss and the erf loss. The square loss, which leads to a classifier obtained by usual least-square linear regression on the class labels, leads to closed form computations and simple analysis. However, losses $\phi(u)$ which remain bounded when u tends to $+\infty$ are viewed as preferable and usually lead to better performance (Hastie et al., 2001). Losses as the hinge loss or the logistic loss, which tend to zero as u tends to $+\infty$, lead to similar performances. In order to study if using such losses might alter the qualitative behavior observed and analyzed for the square loss around extreme testing asymmetries, we use another loss with similar behavior as u tends to $+\infty$, the erf loss. The erf loss is defined as $\phi_{erf}(u) = [u\psi(u) - u + \psi'(u)]$ and leads to simpler computations than the hinge loss or the logistic loss[2].

2. Note that the erf loss ϕ_{erf} is a tight approximation of a rescaled logistic loss $\frac{1}{2}\log(1+e^{-2u})$, with similar derivatives.

We start with an expansion of the unique global minimum (w,b) of the ϕ-cost with asymmetry γ. For the square loss, (w,b) can be obtained in closed form for any class conditional densities so the expansion is easy to obtain, while for the erf loss, an asymptotic analysis of the optimality conditions has to be carried through, and is only valid for mixtures of Gaussians (see Appendix A for a proof).

We use the following usual asymptotic notations, that is, if f and g are two functions defined around $x = 0$, with g everywhere nonnegative then, $f = O(g)$ if and only if there exists an A positive such that $|f(x)| \leqslant Ag(x)$ for all x sufficiently small, $f = o(g)$ if and only if $f(x)/g(x)$ tends to zero when x tends to zero, $f \sim g$ if and only if $f(x)/g(x)$ tends to one when x tends to zero.

Proposition 1 (square loss) *Under previous assumptions, we have the following expansions:*

$$\begin{aligned} w(\gamma) &= 2\frac{p_+}{p_-}\gamma\Sigma_-^{-1}(\mu_+ - \mu_-) + O(\gamma^2), \\ b(\gamma) &= -1 + \frac{p_+}{p_-}\gamma[2 - 2\mu_-^\top(\mu_+ - \mu_-)] + O(\gamma^2), \end{aligned}$$

where $m = \mu_+ - \mu_-$, and $\Sigma_\pm$ and $\mu_\pm$ are the class conditional means and covariance matrices. For mixtures of Gaussians, we have $\Sigma_\pm = \sum_i \pi_\pm^i \Sigma_\pm^i + M_\pm(\mathrm{diag}(\pi_\pm) - \pi_\pm\pi_\pm^\top)M_\pm^\top$ and $\mu_\pm = \sum_i \pi_\pm^i \mu_\pm^i$.

Proposition 2 (erf loss) *Under previous assumptions, we have the following expansions:*

$$\begin{aligned} w(\gamma) &= (2\log(1/\gamma))^{-1/2}\widetilde{\Sigma}_-^{-1}(\tilde{\mu}_+ - \tilde{\mu}_-) + o\left(\log(1/\gamma)^{-1/2}\right), \\ b(\gamma) &= -(2\log(1/\gamma))^{1/2} + o\left(\log(1/\gamma)^{1/2}\right), \end{aligned}$$

with $\tilde{m} = \mu_+ - \tilde{\mu}_-$, $\tilde{\mu}_- = \sum_i \xi_i \mu_-^i$ and $\widetilde{\Sigma}_- = \sum_i \xi_i \Sigma_-^i$, where $\xi \in \mathbb{R}_+^n$ verifies $\sum_i \xi_i = 1$ and ξ is the unique solution of a convex optimization problem defined in Appendix A.

Note that when there is only one mixture component (Gaussian densities), then $\xi_1 = 1$, and the expansion obtained in Proposition 2 has a simpler expression similar to the one from Proposition 1.

The previous propositions provide a closed-form expansion of the solutions of the convex optimization problems defined by the square loss and the erf loss. In the next section, we compute the testing costs using those classifiers.

4.2 Expansion of Testing Asymmetries

Using the expansions of Proposition 1 and 2, we can readily derive an expansion of the ROC coordinates for small γ, as well as the testing asymmetry $\beta(\gamma)$. We have (see Appendix B for a proof):

Proposition 3 (square loss) *Under previous assumptions, we have the following asymptotic expansion:*

$$\log\left(\frac{p_-}{p+}(\beta(\gamma)^{-1} - 1)\right) = A\frac{p_-^2}{8p_+^2}\frac{1}{\gamma^2} + o(1/\gamma^2),$$

where

$$A = \left(\max_{i_-} \frac{1}{m^\top\Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m} - \max_{i_+} \frac{1}{m^\top\Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m}\right).$$

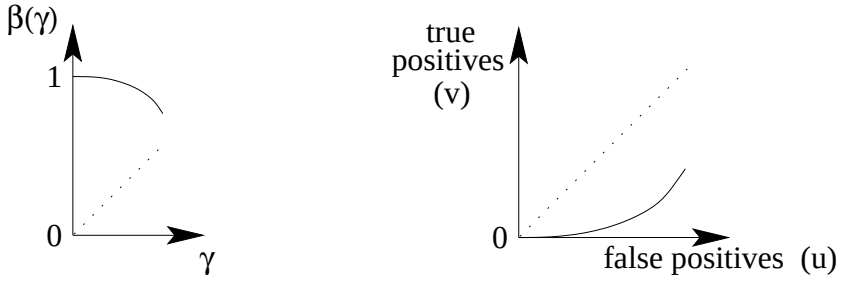


Figure 5: Case $A < 0$: (left) plot of the optimal testing asymmetry $\beta(\gamma)$ as a function of the training asymmetry γ, (right) ROC curve around (0,0) as γ varies close to zero. Classifiers corresponding to γ close to zero are ROC-inconsistent.

Proposition 4 (erf loss) *Under previous assumptions, we have the following asymptotic expansion:*

$$\log\left(\frac{p_-}{p+}(\beta(\gamma)^{-1}-1)\right) = 2\widetilde{A}\log(1/\gamma) + o(\log(1/\gamma)), \tag{4}$$

where

$$\widetilde{A} = \left(\max_{i_-}\frac{1}{\tilde{m}^\top\widetilde{\Sigma}_-^{-1}\Sigma_-^{i_-}\widetilde{\Sigma}_-^{-1}\tilde{m}} - \max_{i_+}\frac{1}{\tilde{m}^\top\widetilde{\Sigma}_-^{-1}\Sigma_+^{i_+}\widetilde{\Sigma}_-^{-1}\tilde{m}}\right).$$

The rest of the analysis is identical for both losses and thus, for simplicity, we focus primarily on the square loss. For the square loss, we have two different regimes, depending on the sign of $A = \left(\max_{i_-}\frac{1}{m^\top\Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m} - \max_{i_+}\frac{1}{m^\top\Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m}\right)$:

- $A < 0$: from the expansion in Eq. (3), we see that $\log\left(\frac{p_-}{p+}(\beta(\gamma)^{-1}-1)\right)$ is asymptotically equivalent to a negative constant times $1/\gamma^2$, implying that the testing asymmetry $\beta(\gamma)$ tends to one exponentially fast. In addition, from Eq. (1), $\frac{p_-}{p+}(\beta(\gamma)^{-1}-1)$ is equal to $\frac{dv/d\gamma}{du/d\gamma}$, which is the slope of the ROC curve. Because this is an expansion around the null training asymmetry, the ROC curve must be starting from the point (0,0); since the slope at that point is zero, that is, proportional to the horizontal axis, the ROC curve is on the bottom right part of the main diagonal and the points corresponding to training asymmetries close to $\gamma = 0$ are not ROC-consistent, that is, the classifiers with training asymmetry too close to zero are useless as they are too extreme. See Figure 5 for a plot of the ROC curve around (0,0) and of $\beta(\gamma)$ around $\gamma = 0$. In this situation, better performance for a given testing asymmetry can be obtained by using a less extreme training asymmetry, simply because too extreme training asymmetries lead to classifier who perform worse than trivial classifiers.

- $A > 0$: from the expansion in Eq. (3), we see that the testing asymmetry tends to 0 exponentially fast, in particular, the derivative $d\beta/d\gamma$ is null at $\gamma = 0$, meaning, that the testing asymmetry is significantly smaller than the training asymmetry, that is, more extreme. The slope of the ROC curve around (0,0) is then vertical. See Figure 6 for a plot of the ROC curve

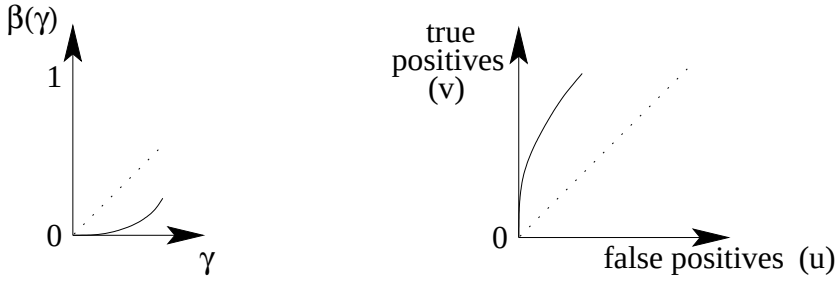


Figure 6: Case $A > 0$. Left: Plot of the optimal testing asymmetry $\beta(\gamma)$ as a function of the training asymmetry γ; Right: ROC curve around (0,0) as γ varies close to zero. Classifiers corresponding to γ close to zero are ROC-consistent, but since $\beta(\gamma) < \gamma$, best performance for a given testing asymmetry is obtained for less extreme training asymmetries.

around (0,0) and of $\beta(\gamma)$ around zero. In this situation, better performance for a given testing asymmetry can be obtained by using a less extreme training asymmetry.

- $A = 0$: the asymptotic expansion does not provide any information relating to the behavior of the testing asymmetry. We are currently investigating higher-order expansions in order to study the behavior of this limiting case. Given two random class-conditional distributions, this regime is unlikely. Precise measure theoretic statements regarding conditions under which the measure of the set of pairs of class-conditional distributions that belong to this regime is zero, are beyond the scope of this paper.

  Note that when the two class conditional densities are Gaussians with identical covariance (a case where the Bayes optimal classifier is indeed linear for all asymmetries), we are in the present case.

Overall, in the two more likely regimes where $A \neq 0$, we have shown that, given an extreme testing asymmetry, the training asymmetry should be chosen less extreme. Because we have considered mixtures of Gaussians, this result applies to a wide variety of distributions. Moreover, this result is confirmed empirically in Section 4.3, where we show in Table 2 examples of the mismatch between testing asymmetry and training asymmetry. Moreover, the strength of the effects we have described above depends on the norm of $m = \mu_+ - \mu_-$: if m is large, that is, the classification problem is simple, then those effects are less strong, while when m is small, they are stronger.

For the erf loss, there are also three regimes, but with weaker effects since the testing asymmetry $\beta(\gamma)$ tends to zero or one polynomially fast, instead of exponentially fast. However, the qualitative result remains the same: there is a mismatch between testing and training asymmetries.

In Figure 7 and Figure 8, we provide several examples for the square loss and the erf loss, with the two regimes $A > 0$ and $A < 0$ and different strengths. Those figures (as well as the bottom of Figure 4) are obtained by solving the respective convex optimization problems with population densities separately for each asymmetry, using Newton's method (Boyd and Vandenberghe, 2003). It is worth noting, that, although the theoretical results obtained in this section are asymptotic expansions around the corners (*i.e.*, extreme asymmetries), the effects also often remain valid far from

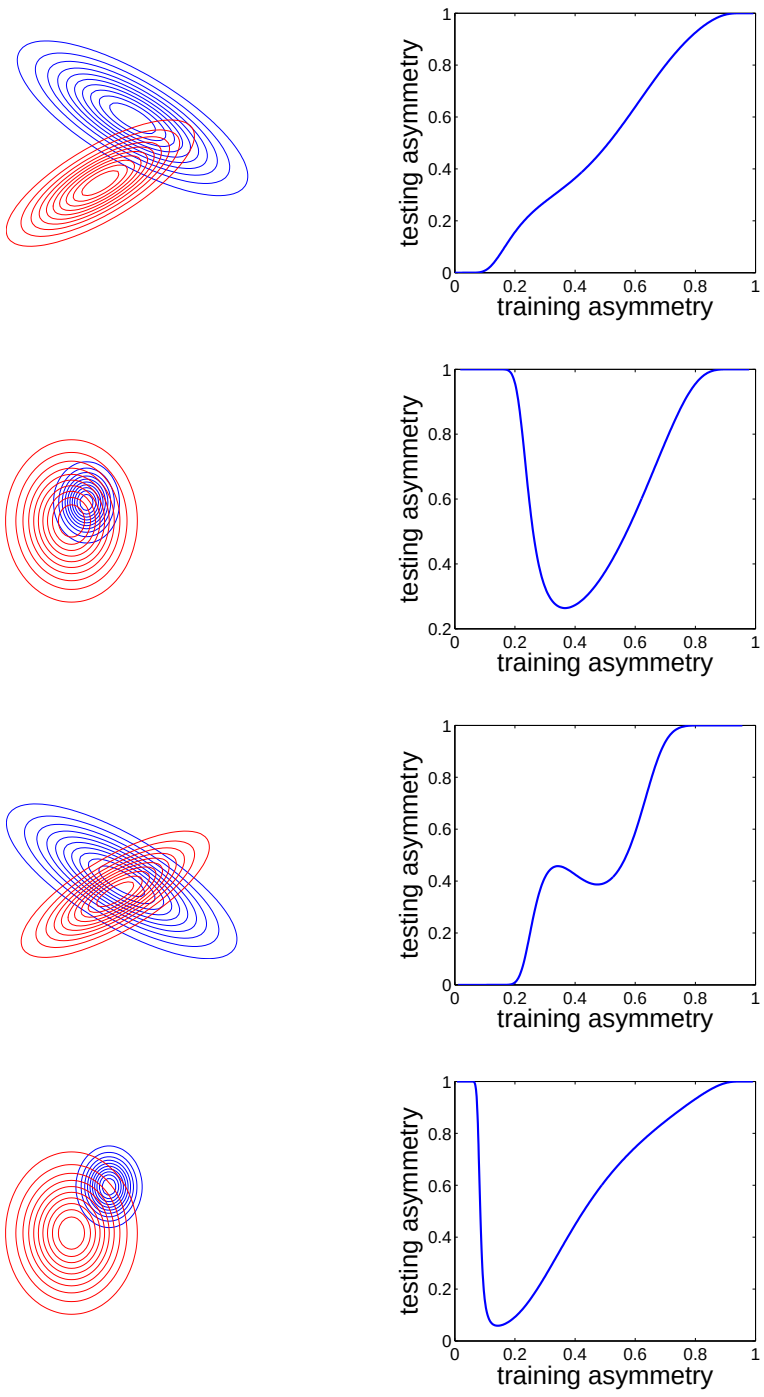


Figure 7: Training asymmetry vs. testing asymmetry, **square loss**: (Left) Gaussian class conditional densities, (right) testing asymmetry vs. training asymmetry; from top to bottom, the values of A are 0.12, -6, 3, -0.96.

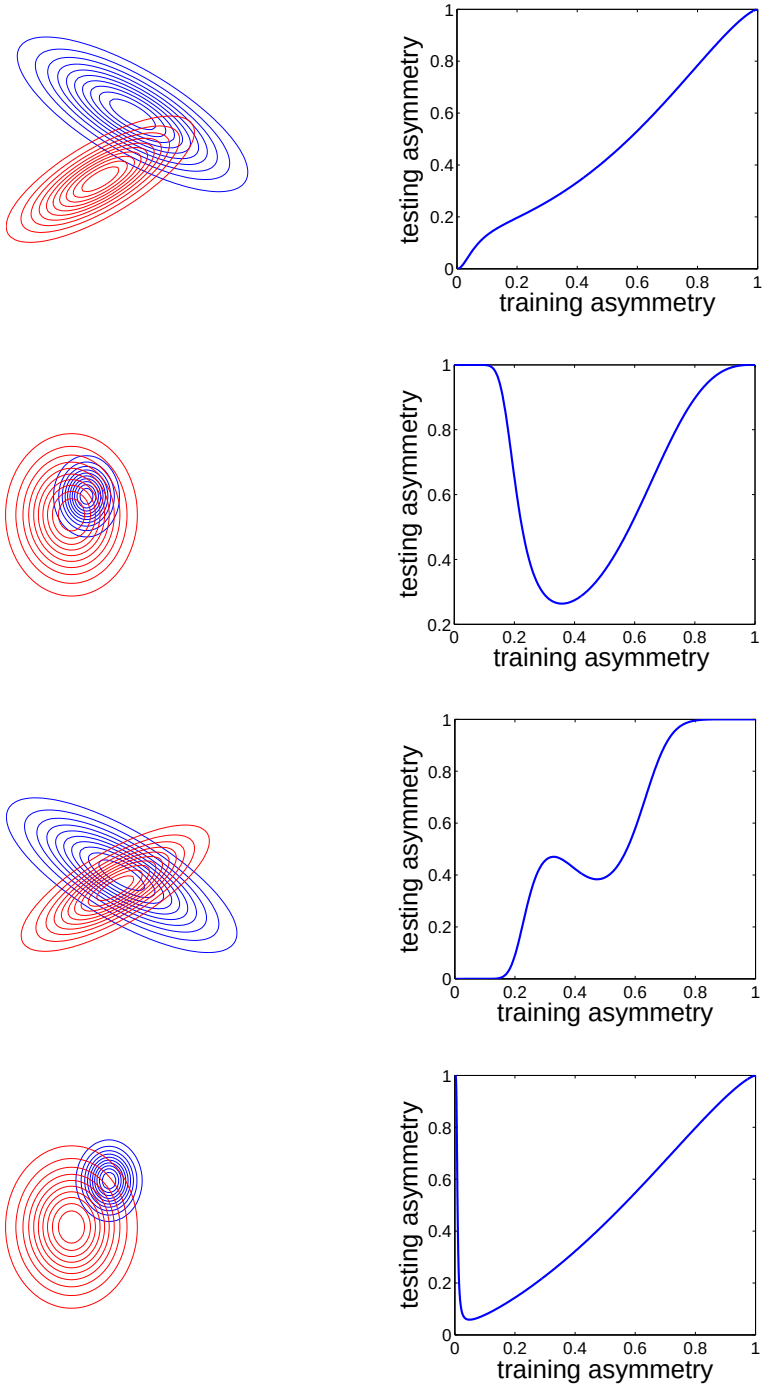


Figure 8: Training asymmetry vs. testing asymmetry, **erf loss**. Left: Gaussian class conditional densities; Right: Testing asymmetry vs. training asymmetry; from top to bottom, the values of $\widetilde{A}$ are 0.12, -6, 3, -0.96.

the corners. However, in general, for non-extreme asymmetries, the mismatch might be inverted, that is, a more extreme training asymmetry might lead to better performance.

Although the results presented in this section show that there are two regimes, given data, they do not readily provide a test to determine which regime is applicable and compute the corresponding optimal training asymmetry for a given testing asymmetry; an approach similar to the one presented by Tong and Koller (2000) could be followed, that is, we could estimate class conditional Gaussian mixture densities and derive the optimal hyperplane from those densities using the tests presented in this paper (which can be performed in closed form for the square loss, while for the erf loss, they require to solve a convex optimization problem). However, this approach would only applicable to extreme asymmetries (where the expansions are valid). Instead, since we have developed an efficient algorithm to obtain linear classifiers for all possible training asymmetries, it is simpler to choose among all asymmetries the one that works best for the problem at hand, which we explore in the next section.

4.3 Building the Entire ROC Curve for a Single Point

As shown empirically in Section 3.2, and demonstrated theoretically in the previous section, training and testing asymmetries may differ; this difference suggests that even when the user is interested in only one cost asymmetry, the training procedure should explore more cost asymmetries, that is, build the ROC curve as presented in Section 3.2 and chose the best classifier as follows: a given testing asymmetry leads to a unique slope in the ROC space, and the optimal point for this asymmetry is the point on the ROC curve whose tangent has the corresponding slope and which is closest to the upper-left corner. In our simulations, we compare two types of ROC curves, one which is obtained by varying the intercept, and one which is obtained by varying both the training asymmetry and the intercept. In our context, using the usual ROC curve obtained by varying only the intercept is essentially equivalent to the common practice of keeping the slope optimized with the convex risk and optimizing the intercept directly using the 0-1 loss instead of the convex surrogate (which is possible by grid search since only one real variable is considered for optimization).

We thus compare in Table 1 and 2, for various data sets and linear classifiers, and for all testing cost asymmetries[3] γ, the three different classifiers: (a) the classifier obtained by optimizing the convex risk with training cost asymmetry γ (a classifier referred to as "one"), (b) the classifier selected from the ROC curve obtained from varying the intercept of the previous classifier (a classifier referred to as "int"), and (c) the classifier selected from the ROC curve obtained by varying both the training asymmetry and the intercept (a classifier referred to as "all").

The ROC curves are obtained by 10-fold cross validation with a wrapper approach to handle the selection of the training asymmetries. The goals of these simulations are to show (1) that using only an approximation to the performance on unseen data is appropriate to select classifiers, and (2) that the performance can be significantly enhanced by simply looking at more training asymmetries. We use several data sets from the UCI data set repository (Blake and Merz, 1998), as well as the mixture of Gaussians shown in Figure 4 and the first Gaussian density in Figure 7.

We used the following wrapper approach to build the ROC curves and compare the performance of the three methods by 10-fold cross validation (Kohavi and John, 1997): for each of the

3. More precisely, 1000 values of testing and training asymmetries were tried, uniformly spread in $[0,1]$. Note that our algorithmic framework allows us to obtain classifiers for all asymmetries in $[0,1]$; therefore considering 1000 values is not computationally expensive.

10 outer folds, we select the best parameters (*i.e.*, training asymmetry and/or intercept) by 10-fold cross validation on the training data. The best classifier for each of the two ROC curves is kept if the difference in performance with the usual classifier "one" on the ten inner folds is statistically significant, as measured by one-tailed Wilcoxon signed rank tests with 5% confidence level (Hollander and Wolfe, 1999). In cases where statistical significance is not found, the original classifier obtained with the training asymmetry equal to the testing asymmetry is used (in this case, the classifier is identical to the classifier "one"). The selected classifier is then trained on the entire training data of the outer fold and tested on the corresponding testing data. We thus obtain for each outer fold, the performance of the three classifiers ("one", "int","all").

In Table 1, we report the results of statistical tests performed to compare the three classifiers. For each testing asymmetry, we performed one-tailed Wilcoxon signed rank tests on the testing costs of the outer folds, with 5% confidence levels (Hollander and Wolfe, 1999). The proportions of acceptance (*i.e.*, the number of acceptances divided by the number of testing asymmetries that are considered) for all one-sided tests are reported in Table 1. The empirical results from columns "all>one" and "int>one" show that comparing classifiers with only an approximation of the testing cost on unseen data, namely cross-validation scores, leads to classifiers that most of the time perform no worse than the usual classifier (based only on the testing asymmetry and the convex risk), which strongly suggests that we are not overfitting the cross-validation data. Moreover, the column "one>int" shows that there is a significant gain in simply optimizing the intercept using the non-convex non-differentiable risk based on the 0-1 loss, and columns "one>all" and "int>all" show that there is even more gain in considering all training asymmetries[4]. As mentioned earlier, when the increase of performance of the classifiers "all" and "int" is not deemed statistically significant on the inner cross-validation folds, the classifier "one" is used instead. It is interesting to note that, when we do not allow the possibility to keep "one", we obtain significantly worse performance on the outer-fold tests

To highlight the potential difference between testing and training asymmetries, Table 2 gives some examples of mismatches between the testing asymmetry and the training asymmetry selected by the cross validation procedure. Since the training asymmetries corresponding to a given testing asymmetry may differ for each outer fold, we use the training asymmetry that was selected by the first (*i.e.*, a random) outer fold of cross validation, and compute the cross-validation scores corresponding to this single training asymmetry for all nine other outer folds. For each data set, testing asymmetries were chosen so that the difference in testing performance between the classifier "one" and "all" on the first fold is greatest. Note that the performance of the classifier "int" is different from the performance of "all" only if the difference was deemed significant on the outer fold used to select the training asymmetry. Results in Table 2 show that in some cases, the difference is large, and that a simple change in the training procedure may lead to substantial gains. Moreover, when the testing asymmetry is extreme, such as for the GAUSSIAN and PIMA data sets, the training asymmetry is less extreme and leads to better performance, illustrating the theoretical results of Section 4. Note also, that for other data sets, such as TWONORM or IONOSPHERE, the optimal training asymmetry is very different from the testing asymmetry: we find that using all asymmetries

4. In some cases, considering all training asymmetries performs worse than using a single asymmetry with optimized intercept (column "all>int" in Table 1); in those cases, the difference in performance, although statistically significant, is small and we conjecture it is due to using an approximation of expected testing performance to select the training asymmetry.

Data set	d	n	one>all	all>one	int>all	all>int	one>int	int>one
BREAST	9	683	8.7	0.0	4.1	0.0	5.2	0.0
DERMATOLOGY	34	358	0.0	0.0	0.0	0.0	0.0	0.0
GAUSSIANS	2	2000	17.6	0.0	14.2	0.0	9.5	0.0
MIXTURES	2	2000	49.7	0.0	25.8	0.0	41.2	0.0
LIVER	6	345	0.0	0.0	0.0	0.0	0.0	0.0
VEHICLE	18	416	13.1	0.0	13.1	0.0	0.0	0.0
PIMA	8	768	9.2	0.0	0.0	0.0	5.6	0.0
RINGNORM	2	2000	48.6	0.0	5.9	0.0	40.4	0.0
TWONORM	2	2000	97.4	0.0	15.8	5.2	91.3	0.0
ADULT	13	2000	12.0	0.0	8.8	0.0	3.2	0.0
IONOSPHERE	33	351	42.0	0.0	13.2	0.0	22.5	0.0

Table 1: Comparison of performances of classifiers: for each data set, the number of features is d, the total number of data points is n, and the proportions of acceptance (*i.e.*, the number of acceptances divided by the number of testing asymmetries that are considered) of the six one-sided tests between the three classifiers are given in the last six columns (Wilcoxon signed rank tests with 5% confidence level comparing cross-validation scores). Note that we compare testing by using costs, and hence better performance corresponds to smaller costs. See text for details.

Data set	γ (test)	"one"	"int"	"all"	γ (train)
BREAST	0.008	0.51 ± 0.07	0.55 ± 0.96	0.51 ± 0.07	0.995
DERMATOLOGY	0.991	0.37 ± 0.05	0.37 ± 0.05	0.37 ± 0.05	0.991
GAUSSIANS	0.067	6.66 ± 0.00	5.10 ± 0.59	6.66 ± 0.00	0.128
MIXTURES	0.327	29.80 ± 2.61	16.73 ± 2.20	19.70 ± 1.82	0.990
LIVER	0.192	16.10 ± 0.96	15.96 ± 4.44	16.10 ± 0.96	0.560
PIMA	0.917	10.83 ± 0.35	9.61 ± 2.32	10.83 ± 0.35	0.500
RINGNORM	0.327	33.18 ± 1.07	25.53 ± 3.95	25.15 ± 4.86	0.500
VEHICLE	0.010	0.21 ± 0.15	0.21 ± 0.15	0.21 ± 0.15	0.010
TWONORM	0.382	36.98 ± 0.34	1.46 ± 1.00	1.55 ± 1.35	0.872
ADULT	0.947	2.67 ± 0.09	2.50 ± 0.08	2.67 ± 0.09	0.500
IONOSPHERE	0.933	4.77 ± 0.21	2.30 ± 2.22	4.77 ± 0.21	0.067

Table 2: Training with the testing asymmetry γ versus training with all cost asymmetries: we report cross-validation testing costs (multipled by 100). Only the asymmetry with the largest difference in testing performance between the classifier "one" and "all" is reported. We also report the training asymmetry which led to the best performance. Given an asymmetry γ we use the cost settings $C_+ = 2\gamma$, $C_- = 2(1-\gamma)$ (which leads to the misclassification error if $\gamma = 1/2$). See text for details.

leads to better performance. However, we have not identified general rules for selecting the best training asymmetry.

5. Conclusion

We have presented an efficient algorithm to build ROC curves by varying the training cost asymmetries for SVMs. The algorithm is based on the piecewise linearity of the path of solutions when the cost of false positives and false negatives vary. We have also provided a theoretical analysis of the relationship between the potentially different cost asymmetries assumed in training and testing classifiers, showing that they differ under certain circumstances. In particular, in case of extreme asymmetries, our theoretical analysis suggests that training asymmetries should be chosen less extreme than the testing asymmetry.

We have characterized key relationships, and have worked to highlight the potential practical value of building the entire ROC curve even when a single point may be needed. All learning algorithms considered in this paper involve using a convex surrogate to the correct non differentiable non convex loss function. Our theoretical analysis implies that because we use a convex surrogate, using the testing asymmetry for training leads to non-optimal classifiers. We thus propose to generate all possible classifiers corresponding to all training asymmetries, and select the one that optimizes a good approximation to the true loss function on unseen data (*i.e.*, using held out data or cross validation). As shown in Section 3, it turns out that this can be done efficiently for the support vector machine. Such an approach can lead to a significant improvement of performance with little added computational cost.

Finally, we note that, although we have focused in this paper on the single kernel learning problem, our approach can be readily extended to the multiple kernel learning setting (Bach et al., 2005b) with appropriate numerical path following techniques.

Acknowledgments

We thank John Platt for sharing his experiences and insights with considerations of cost functions in training and testing support vector machines.

Appendix A. Proof of Expansion of Optimal Solutions

In this appendix, we give the proof of the expansions of optimal solutions for extreme asymmetries, for the square and erf loss. We perform the expansions using the variable $\rho(\gamma) = \dfrac{C_+ p_+}{C_- p_-} = \dfrac{\gamma p_+}{(1-\gamma) p_-} = \dfrac{\gamma p_+}{p_-} + O(\gamma^2)$ around zero.

A.1 Square Loss. Proof of Proposition 1.

In this case, the classifier is simply a linear regression on y and (w,b) can be obtained in closed form as the solution of the following linear system (obtained from the normal equations):

$$\begin{aligned} b &= \frac{\rho-1}{\rho+1} - w^\top \frac{\rho\mu_+ + \mu_-}{\rho+1}, \\ \left(\rho\Sigma_+ + \Sigma_- + \frac{\rho}{\rho+1}(\mu_+-\mu_-)(\mu_+-\mu_-)^\top\right) w &= \frac{2\rho}{\rho+1}(\mu_+-\mu_-), \end{aligned}$$

where Σ_+ and Σ_- are the class conditional means and covariance matrices.

The first two terms of the Taylor expansions around $\rho = 0$ (*i.e.*, around $\gamma = 0$) are straightforward to obtain:

$$\begin{aligned} w &= 2\rho\Sigma_-^{-1}m - 2\rho^2\left[\Sigma_-^{-1}m + \Sigma_-^{-1}(\Sigma_+ + mm^\top)\Sigma_-^{-1}m\right] + O(\rho^3), \\ b &= -1 + \rho\left[2 - 2\mu_-^\top\Sigma_-^{-1}m\right] + O(\rho^2). \end{aligned}$$

A.2 Erf Loss. Proof of Proposition 2.

We begin by proving Proposition 2 in the Gaussian case, where the proof is straightforward, and we then extend to the Gaussian mixture case. In order to derive optimality conditions for the erf loss, we first need to compute expectations of the erf loss and its derivatives for Gaussian densities.

A.2.1 EXPECTATION OF THE ERF LOSS AND ITS DERIVATIVES FOR GAUSSIAN DENSITIES

A short calculation shows that, when expectations are taken with respect to a normal distribution with mean μ and covariance matrix Σ, we have:

$$\begin{aligned} E\phi_{erf}(w^\top x+b) &= (-w^\top\mu - b)\psi\left(\frac{-w^\top\mu-b}{(1+w^\top\Sigma w)^{1/2}}\right) \\ &\qquad + (1+w^\top\Sigma w)^{1/2}\psi'\left(\frac{-w^\top\mu-b}{(1+w^\top\Sigma w)^{1/2}}\right), \\ E\frac{\partial\phi_{erf}(w^\top x+b)}{\partial w} &= -\mu\psi\left(\frac{-w^\top\mu-b}{(1+w^\top\Sigma w)^{1/2}}\right) + \frac{\Sigma w}{(1+w^\top\Sigma w)^{1/2}}\psi'\left(\frac{-w^\top\mu-b}{(1+w^\top\Sigma w)^{1/2}}\right), \\ E\frac{\partial\phi_{erf}(w^\top x+b)}{\partial b} &= -\psi\left(\frac{-w^\top\mu-b}{(1+w^\top\Sigma w)^{1/2}}\right). \end{aligned}$$

A.2.2 GAUSSIAN CASE

In order to derive the asymptotic expansion, we derive the optimality conditions for (w,b) and study the behavior as ρ tends to zero. Let's define $t_- = \dfrac{w^\top\mu_- + b}{(1+w^\top\Sigma_- w)^{1/2}}$ and $t_+ = \dfrac{-w^\top\mu_+ - b}{(1+w^\top\Sigma_+ w)^{1/2}}$.

The optimality conditions for (w,b) are the following (obtained by zeroing derivatives with respect to b and w):

$$p_+C_+\left(-\mu_+\psi(t_+) + \frac{\Sigma_+ w}{(1+w^\top\Sigma_+ w)^{1/2}}\psi'(t_+)\right)$$

$$+p_- C_- \left(\mu_- \psi(t_-) + \frac{\Sigma_- w}{(1+w^\top \Sigma_- w)^{1/2}} \psi'(t_-) \right) = 0,$$

$$-p_+ C_+ \psi(t_+) + p_- C_- \psi(t_-) = 0.$$

They can be rewritten as follows (with $\rho = \frac{C_+ p_+}{C_- p_-}$):

$$\rho \left(-\mu_+ \psi(t_+) + \frac{\Sigma_+ w}{(1+w^\top \Sigma_+ w)^{1/2}} \psi'(t_+) \right) + \left(\mu_- \psi(t_-) + \frac{\Sigma_- w}{(1+w^\top \Sigma_- w)^{1/2}} \psi'(t_-) \right) = 0, \quad (5)$$

$$\rho \psi(t_+) = \psi(t_-). \quad (6)$$

Since the cumulative function ψ is always between zero and one, Eq. (6) implies that as ρ tends to zero, $\psi(t_-)$ tends to zero, and thus t_- tends to $-\infty$. This in turn implies that $b/(1+||w||)$ tends to $-\infty$, which in turn implies that t_+ tends to infinity and $\psi(t_+)$ tends to 1.

It is well known that as z tends to $-\infty$, we have $\psi(z) \approx \frac{\psi'(z)}{-z}$ (see, for example, Bleistein and Handelsman (1986)). Thus, if we divide Eq. (5) by $\psi(t_-)$, we get:

$$-\rho \frac{\psi(t_+)}{\psi(t_-)} \mu_+ + \rho \frac{\Sigma_+ w}{(1+w^\top \Sigma_+ w)^{1/2}} \frac{\psi'(t_+)}{\psi(t_+)} \frac{\psi(t_+)}{\psi(t_-)} + \mu_- + \frac{\Sigma_- w}{(1+w^\top \Sigma_- w)^{1/2}} \frac{\psi'(t_-)}{\psi(t_-)} = 0, \text{ that is,}$$

$$\frac{\Sigma_+ w}{(1+w^\top \Sigma_+ w)^{1/2}} \frac{\psi'(t_+)}{\psi(t_+)} + \frac{\Sigma_- w}{(1+w^\top \Sigma_- w)^{1/2}} (-t_-) \sim \mu_+ - \mu_-. \quad (7)$$

The first term in the left hand side of Eq. (7) is the product of a bounded term and a term that goes to zero, it is thus tending to zero as ρ tends to zero. Since the right hand side is constant, this implies that the second term in the left hand side must be bounded, and thus, since $|t_-|$ tends to infinity, that w tends to zero. By removing all negligible terms in Eq. (7), we have the following expansion for w:

$$w \approx \frac{1}{-t_-} \Sigma_-^{-1} (\mu_+ - \mu_-).$$

In order to obtain the expansion as a function of ρ, we need to expand t_-. From Eq. (6) and the fact that $\psi(t_+)$ tends to one, we obtain $\psi(t_-) \sim \rho$, A classical asymptotic result on the inverse erf function shows that $t_- \sim -\sqrt{2\log(1/\rho)}$. This finally leads to:

$$\begin{aligned} b(\rho) &\sim -(2\log(1/\rho))^{1/2}, \\ w(\rho) &\sim (2\log(1/\rho))^{-1/2} \Sigma_-^{-1} (\mu_+ - \mu_-). \end{aligned}$$

A.2.3 GENERAL CASE

We assume that each $\pi_\pm^i$ is strictly positive and that each covariance matrix $\Sigma_\pm^i$ is positive definite. We define $t_-^i = \frac{w^\top \mu_-^i + b}{(1+w^\top \Sigma_-^i w)^{1/2}}$ and $t_+^i = \frac{-w^\top \mu_+^i - b}{(1+w^\top \Sigma_+^i w)^{1/2}}$.

Without loss of generality, we can assume that the positive class is centered, that is, $\mu_+ = \sum_i \pi_+^i \mu_+^i = 0$. The optimality conditions for (w,b) are the following (obtained by zeroing derivatives with respect to b and w):

$$\rho\left(\sum_i \pi_+^i \left\{-\mu_+^i \psi(t_+^i) + \frac{\Sigma_+^i w}{(1+w^\top \Sigma_+^i w)^{1/2}}\psi'(t_+^i)\right\}\right) + \left(\sum_i \pi_-^i \left\{\mu_-^i \psi(t_-^i) + \frac{\Sigma_-^i w}{(1+w^\top \Sigma_- w)^{1/2}}\psi'(t_-^i)\right\}\right) = 0, \tag{8}$$

$$-\rho \sum_i \pi_+^i \psi(t_+^i) + \sum_i \pi_-^i \psi(t_-^i) = 0. \tag{9}$$

From Eq. (9), we obtain that $\psi(t_-^i)$ tends to zero for all i, and this implies that $b/(1+||w||)$ tends to $-\infty$, and in turn that $\psi(t_+^i)$ tends to 1 for all i. We can now divide Eq. (8) by $\sum_i \pi_-^i \psi(t_-^i) = \rho \sum_i \pi_+^i \psi(t_+^i)$, to obtain:

$$\frac{\sum_i \pi_-^i \psi(t_-^i)\mu_-^i}{\sum_i \pi_-^i \psi(t_-^i)} - \frac{\sum_i \pi_+^i \psi(t_+^i)\mu_+^i}{\sum_i \pi_+^i \psi(t_+^i)} = -\frac{1}{\sum_i \pi_+^i \psi(t_+^i)} \sum_i \pi_+^i \psi(t_+^i) \left\{\frac{\Sigma_+^i w}{(1+w^\top \Sigma_+^i w)^{1/2}} \frac{\psi'(t_+^i)}{\psi(t_+^i)}\right\} - \frac{1}{\sum_i \pi_-^i \psi(t_-^i)} \sum_i \pi_-^i \psi(t_-^i) \left\{\frac{\Sigma_-^i w}{(1+w^\top \Sigma_-^i w)^{1/2}} \frac{\psi'(t_-^i)}{\psi(t_-^i)}\right\}.$$

Let $\tilde{\pi}_i^+ = \frac{\pi_+^i \psi(t_+^i)}{\sum_j \pi_+^j \psi(t_+^j)}$ and $\tilde{\pi}_i^- = \frac{\pi_-^i \psi(t_-^i)}{\sum_j \pi_-^j \psi(t_-^j)}$. We can rewrite the preceding equation as

$$\sum_i \tilde{\pi}_-^i \mu_-^i - \sum_i \tilde{\pi}_+^i \mu_+^i = -\sum_i \tilde{\pi}_+^i \left\{\frac{\Sigma_+^i w}{(1+w^\top \Sigma_+^i w)^{1/2}} \frac{\psi'(t_+^i)}{\psi(t_+^i)}\right\} - \sum_i \tilde{\pi}_-^i \left\{\frac{\Sigma_-^i w}{(1+w^\top \Sigma_-^i w)^{1/2}} \frac{\psi'(t_-^i)}{\psi(t_-^i)}\right\}.$$

As in the Gaussian case, the first term of the right hand is negligible compared to the second term when ρ goes to zero. Moreover, for all i, we have $\psi'(t_-^i)/\psi(t_-^i) \approx -t_-^i$ and we thus get:

$$\sum_i \tilde{\pi}_-^i \mu_-^i - \sum_i \tilde{\pi}_+^i \mu_+^i = -\sum_i \tilde{\pi}_-^i \left\{\frac{-t_-^i}{(1+w^\top \Sigma_-^i w)^{1/2}}\right\} \Sigma_-^i w. \tag{10}$$

The left hand side of Eq. (10) is bounded; Because $\tilde{\pi}^-$ sums to one, the lowest eigenvalue of the matrix $\sum_i \tilde{\pi}_-^i \Sigma_-^i$ is lower-bounded by the smallest of the smallest eigenvalues of Σ_-^i, $i = 1,\ldots,k_-$. Thus Eq. (10) implies that $t_-^i w$ is bounded as ρ tends to zero. This in turn implies that w tends to zero, that $b \sim t_-^i$ for all i, and that bw is bounded.

The quantities bw, $\tilde{\pi}_+$ and $\tilde{\pi}_-$ are functions of ρ. As ρ tends to zero, they all remain bounded. We now proceed to prove that all points of accumulation of those quantities as ρ tends to zero satisfy a set of equations with an unique solution. This will imply that those quantities converge as ρ tends to zero.

Equation for $\tilde{\pi}_+$ Since t_+^i tends to $+\infty$, $\tilde{\pi}_+^i$ tends to π_+^i, and $\sum_i \tilde{\pi}_+^i \mu_+^i$ tends to zero, since we have assumed that $\sum_i \pi_+^i \mu_+^i = 0$.

Equation for $\tilde{\pi}_-$ and bw Let θ and ξ be points of accumulation of bw and $\tilde{\pi}_-$ as ρ tends to zero (*i.e.*, θ and ξ are limits of sequences $b(\rho_k)w(\rho_k)$ and $\tilde{\pi}_-(\rho_k)$ as k tends to ∞, with $\rho_k \to 0$). From Eq. (10), we get:

$$\sum_i \xi_i \mu_-^i - 0 = \left(\sum_i \xi_i \Sigma_-^i \right) \theta. \tag{11}$$

We can now expand $(t_-^i)^2 - (t_-^j)^2$ for all i, j by keeping the leading terms, noting that $w \to 0$, $|b| \to +\infty$, and bw is bounded:

$$\begin{aligned}
(t_-^i)^2 - (t_-^j)^2 &= \frac{(w^\top \mu_-^i + b)^2}{1 + w^\top \Sigma_-^i w} - \frac{(w^\top \mu_-^j + b)^2}{1 + w^\top \Sigma_-^j w} \\
&\sim 2bw^\top(\mu_-^i - \mu_-^j) - b^2 w^\top(\Sigma_-^i - \Sigma_-^j)w \\
&\to (2\theta^\top \mu_-^i - \theta^\top \Sigma_-^i \theta) - (2\theta^\top \mu_-^j - \theta^\top \Sigma_-^j \theta) \text{ as } \rho_k \to 0.
\end{aligned}$$

We thus have

$$\begin{aligned}
\frac{\tilde{\pi}_-^i}{\tilde{\pi}_-^j} &= \frac{\pi_-^i \psi(t_-^i)}{\pi_-^j \psi(t_-^j)} = \frac{\pi_-^i \psi'(t_-^i)}{\pi_-^j \psi'(t_-^j)} \frac{\psi'(t_-^j)}{\psi(t_-^j)} \\
&\sim \frac{\pi_-^i}{\pi_-^j} \frac{\psi'(t_-^i)}{\psi'(t_-^j)} \frac{t_-^j}{t_-^i} \sim \frac{\pi_-^i}{\pi_-^j} \exp(-\frac{1}{2}[(t_-^i)^2 - (t_-^j)^2]) \text{ since } t_-^i \sim t_-^j \\
&\to \frac{\pi_-^i \exp(-\theta^\top \mu_-^i + \frac{1}{2}\theta^\top \Sigma_-^i \theta)}{\pi_-^j \exp(-\theta^\top \mu_-^j + \frac{1}{2}\theta^\top \Sigma_-^j \theta)} \text{ as } \rho_k \to 0,
\end{aligned}$$

which implies that the vector ξ is proportional to the vector with components $\pi_-^i \exp(-\theta^\top \mu_-^i + \frac{1}{2}\theta^\top \Sigma_-^i \theta)$, that is:

$$\forall i,\ \xi_i = \frac{\pi_i \exp(-\theta^\top \mu_-^i + \frac{1}{2}\theta^\top \Sigma_-^i \theta)}{\sum_j \pi_j \exp(-\theta^\top \mu_-^j + \frac{1}{2}\theta^\top \Sigma_-^j \theta)}. \tag{12}$$

We have shown that points of accumulation (θ, ξ) must verify two equations, Eq. (11) and Eq. (12).

Unique solution of Eq. (11) and Eq. (12) We now prove that Eq. (11) and Eq. (12) together have an unique solution, obtained as the optimum solution of a strictly convex problem. From Eq. (11), we can write θ as a function of ξ as:

$$\theta(\xi) = \left(\sum_i \xi_i \Sigma_-^i \right)^{-1} \sum_i \xi_i \mu_-^i. \tag{13}$$

Let us define the following function defined on the positive orthant $\{\xi, \xi_i > 0, \forall i\}$:

$$H(\xi) = \sum_i \xi_i \log \xi_i - \sum_i \xi_i \left\{ \log \pi_-^i - \theta(\xi)^\top \mu_-^i + \frac{1}{2}\theta(\xi)^\top \Sigma_-^i \theta(\xi) \right\}.$$

Short calculations show that:

$$\begin{aligned}
\frac{\partial \theta}{\partial \xi_i} &= \left(\sum_k \xi_k \Sigma_-^k\right)^{-1} \left(\mu_-^i - \Sigma_-^i \theta(\xi)\right), \\
\frac{\partial \left\{-\theta(\xi)^\top \mu_-^i + \frac{1}{2}\theta(\xi)^\top \Sigma_-^i \theta(\xi)\right\}}{\partial \xi_j} &= -\left(\mu_-^i - \Sigma_-^i \theta(\xi)\right) \left(\sum_k \xi_k \Sigma_-^k\right)^{-1} \left(\mu_-^j - \Sigma_-^j \theta(\xi)\right), \\
\frac{\partial H}{\partial \xi_i} &= \log \xi_i + 1 - \left(\log \pi_-^i - \theta(\xi)^\top \mu_-^i + \frac{1}{2}\theta(\xi)^\top \Sigma_-^i \theta(\xi)\right), \\
\frac{\partial^2 H}{\partial \xi_i \partial \xi_j} &= \delta_{ij} \frac{1}{\xi_i} + \left(\mu_-^i - \Sigma_-^i \theta(\xi)\right) \left(\sum_k \xi_k \Sigma_-^k\right)^{-1} \left(\mu_-^j - \Sigma_-^j \theta(\xi)\right).
\end{aligned}$$

The last equation shows that the function H is strictly convex in the positive orthant. Thus, minimizing $H(\xi)$ subject to $\sum_i \xi_i = 1$ has an unique solution. Optimality conditions are derived by writing down the Lagrangian:

$$\mathcal{L}(\xi, \alpha) = H(\xi) + \alpha\left(\sum_i \xi_i - 1\right),$$

which leads to the following optimality conditions:

$$\forall i,\ \frac{\partial H}{\partial \xi_i} + \alpha = 0, \tag{14}$$

$$\sum_i \xi_i = 1. \tag{15}$$

The last two equations are exactly equivalent to Eq. (12). We have thus proved that the system defining θ and ξ (Eq. (11) and Eq. (12)) has an unique solution obtained from the solution of the convex optimization problem:

$$\begin{array}{rl}
\text{minimize} & \displaystyle\sum_i \xi_i \log \xi_i - \sum_i \xi_i \left\{\log \pi_-^i - \theta(\xi)^\top \mu_-^i + \frac{1}{2}\theta(\xi)^\top \Sigma_-^i \theta(\xi)\right\} \\
\text{with respect to} & \xi \\
\text{such that} & \xi_i \geqslant 0, \forall i \\
& \displaystyle\sum_i \xi_i = 1
\end{array}$$

with

$$\theta(\xi) = \left(\sum_i \xi_i \Sigma_-^i\right)^{-1} \sum_i \xi_i \mu_-^i.$$

Asymptotic equivalent From the value of θ and ξ obtained above, we can derive the asymptotic expansions of w and b. From Eq. (9) and the fact that t_+^i tends to $+\infty$, we get $\sum_i \pi_-^i \psi(t_-^i) \sim \rho \sum_i \pi_+^i = \rho$. In addition, we can show by expanding $(t_-^i)^2$ that $(t_-^i)^2 - b^2$ has a limit when ρ tends to 0, which in turn implies that $\psi(b)/\rho$ has a finite limit. This implies that $b \approx -(2\log(1/\rho))^{1/2}$. From the fact that bw tends to a limit θ, we immediately obtain that $w \approx \theta/b$, which completes the proof of Proposition 2.

Appendix B. Proof of Expansion of Testing Asymmetries

For the two losses we considered (square and erf), the expansions of w and b around $\gamma = 0$ lead to

$$\frac{w(\gamma)}{b(\gamma)} \approx -c(\gamma)a,$$

where $c(\gamma) = 2\frac{p_+}{p_-}\gamma$, $a = \Sigma_-^{-1}(\mu_+ - \mu_-)$ for the square loss and $c(\gamma) = (2\log(1/\gamma))^{-1}$, $a = \widetilde{\Sigma}_-^{-1}(\tilde{\mu}_+ - \tilde{\mu}_-)$ for the erf loss.

The proportion of false positives $u(\gamma)$ and true positives $v(\gamma)$ can be obtained as:

$$\begin{aligned} u(\gamma) &= P(w^\top x + b \geqslant 0 | y = -1) = \sum_i \pi_-^i \psi\left(\frac{w^\top \mu_-^i + b}{(w^\top \Sigma_-^i w)^{1/2}}\right) = \psi(t_u^i(\gamma)), \\ v(\gamma) &= P(w^\top x + b \geqslant 0 | y = 1) = \sum_i \pi_+^i \psi\left(\frac{w^\top \mu_+^i + b}{(w^\top \Sigma_+^i w)^{1/2}}\right) = \psi(t_v^i(\gamma)), \end{aligned}$$

and we have the expansions

$$\begin{aligned} t_u^i(\gamma) \triangleq \frac{w^\top \mu_-^i + b}{(w^\top \Sigma_-^i w)^{1/2}} &\approx \frac{-1}{c(\gamma)(a^\top \Sigma_-^i a)^{1/2}}, \\ t_v^i(\gamma) \triangleq \frac{w^\top \mu_+^i + b}{(w^\top \Sigma_+^i w)^{1/2}} &\approx \frac{-1}{c(\gamma)(a^\top \Sigma_+^i a)^{1/2}}, \\ \frac{du}{d\gamma} &= \sum_i \pi_-^i \frac{dt_u^i}{dc}\frac{dc}{d\gamma}\psi'(t_u^i(\gamma)) \sim \frac{dc}{d\gamma}\sum_i \pi_-^i \frac{1}{\sqrt{2\pi}}\frac{\exp(-(a^\top \Sigma_-^i a)^{-1}/2c(\gamma)^2)}{c(\gamma)^2 (a^\top \Sigma_-^i a)^{-1/2}}, \\ \frac{dv}{d\gamma} &= \sum_i \pi_+^i \frac{dt_v^i}{dc}\frac{dc}{d\gamma}\psi'(t_v^i(\gamma)) \sim \frac{dc}{d\gamma}\sum_i \pi_+^i \frac{1}{\sqrt{2\pi}}\frac{\exp(-(a^\top \Sigma_+^i a)^{-1}/2c(\gamma)^2)}{c(\gamma)^2 (a^\top \Sigma_+^i a)^{-1/2}}. \end{aligned}$$

The expansions of $\frac{du}{d\gamma}$ and $\frac{dv}{d\gamma}$ are each dominated by a single term, corresponding to indices i_- and i_+ that respectively maximized $(a^\top \Sigma_-^i a)^{-1}$ and $(a^\top \Sigma_+^i a)^{-1}$ (we assume for simplicity that all values of $a^\top \Sigma_\pm^i a$ are distinct). We then obtain

$$\log\left(\frac{dv}{d\gamma}\Big/\frac{du}{d\gamma}\right) \sim \frac{1}{2c(\gamma)^2}\left(\frac{1}{a^\top \Sigma_-^{i_-} a} - \frac{1}{a^\top \Sigma_-^{i_+} a}\right).$$

Proposition 3 and 4 follows from $\frac{dv}{d\gamma}\Big/\frac{du}{d\gamma} = \frac{p_-}{p_+}(\beta(\gamma)^{-1} - 1)$, which is a consequence of Eq. (1).

References

D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms ? In *Computational Geometry: Theory and Applications*, volume 7, 1997.

F. R. Bach, D. Heckerman, and E. Horvitz. On the path to an ideal ROC curve: considering cost asymmetry in learning classifiers. In *Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2005a.

F. R. Bach, R. Thibaux, and M. I. Jordan. Computing regularization paths for learning multiple kernels. In *Advances in Neural Information Processing Systems, 17*. MIT Press, 2005b.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Large margin classifiers: convex loss, low noise, and convergence rates. In *Advances in Neural Information Processing Systems, 16*. MIT Press, 2004.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

N. Bleistein and R. A. Handelsman. *Asymptotic Expansions of Integrals*. Dover, 1986.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

P. A. Flach. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In *International Conference on Machine Learning (ICML)*, 2003.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2005.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

M. Hollander and D. A. Wolfe. *Nonparametric statistical inference*. John Wiley & Sons, 1999.

R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.

I. Maros. *Computational Techniques of the Simplex Method*. Klüwer Academic Publishers, 2002.

A. Y. Ng. Preventing overfitting of cross-validation data. In *International Conference on Machine Learning (ICML)*, 1997.

M. S. Pepe. Receiver operating characteristic methodology. *Journal of the American Statistical Association*, 95(449):308–311, 2000.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1998.

F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning Journal*, 42(3):203–231, 2001.

K. Scheinberg. An efficient implementation of an active set method for SVM. *Journal of Machine Learning Research*, to appear, 2006.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

S. Tong and D. Koller. Restricted Bayes optimal classifiers. In *American Conference on Artificial Intelligence (AAAI-00)*, 2000.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–85, 2004.

Journal of Machine Learning Research 7 (2006) 1743-1788 Submitted 11/05; Revised 4/06; Published 8/06

Learning Factor Graphs in Polynomial Time and Sample Complexity

Pieter Abbeel PABBEEL@CS.STANFORD.EDU
Daphne Koller KOLLER@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford, CA 94305, USA

Editor: Sanjoy Dasgupta

Abstract

We study the computational and sample complexity of parameter and structure learning in graphical models. Our main result shows that the class of factor graphs with bounded degree can be learned in polynomial time and from a polynomial number of training examples, assuming that the data is generated by a network in this class. This result covers both parameter estimation for a known network structure and structure learning. It implies as a corollary that we can learn factor graphs for both Bayesian networks and Markov networks of bounded degree, in polynomial time and sample complexity. Importantly, unlike standard maximum likelihood estimation algorithms, our method does not require inference in the underlying network, and so applies to networks where inference is intractable. We also show that the error of our learned model degrades gracefully when the generating distribution is not a member of the target class of networks. In addition to our main result, we show that the sample complexity of parameter learning in graphical models has an $O(1)$ dependence on the number of variables in the model when using the KL-divergence normalized by the number of variables as the performance criterion.[1]

Keywords: probabilistic graphical models, parameter and structure learning, factor graphs, Markov networks, Bayesian networks

1. Introduction

Graphical models are widely used to compactly represent structured probability distributions over (large) sets of random variables. Learning a graphical model from data is important for many applications. This learning problem can vary along several axes, including whether the data is fully or partially observed, and whether the structure of the network is given or needs to be learned from data.

In this paper, we focus on the problem of learning both network structure and parameters from fully observable data, restricting attention to discrete probability distributions over finite sets. We focus on the problem of learning a factor graph representation (Kschischang et al., 2001) of the distribution. Factor graphs subsume both Bayesian networks and Markov networks, in that every Bayesian network or Markov network can be written as a factor graph of (essentially) the same size.[2]

1. A preliminary version of some of this work was reported in Abbeel et al. (2005).
2. The factor graph corresponding to either a Bayesian network or a Markov network can be constructed in linear time (as a function of the size of the original network). See, for example, Kschischang et al. (2001), and Yedidia et al.

We provide a new parameterization of factor graph distributions, which forms the basis for our results. In this new parameterization, every factor is written as a product of probabilities over the variables in the factor and its neighbors. We will refer to such subsets of variables as "local subsets of variables." These local subsets of variables are of size at most d^2 for factor graphs of bounded degree d. Thus, for factor graphs of bounded degree d, the probabilities appearing in our new parameterization are over at most d^2 variables and can be estimated efficiently from training examples.[3] Hence this new parameterization naturally leads to an algorithm that solves the *parameter learning* problem *in closed-form* by estimating the probabilities over these local subsets of variables from training examples. We show that our closed-form estimation procedure results in a good estimate of the true distribution. More specifically, for factor graphs of bounded degree, if the generating distribution falls into the target class, we show that our estimation procedure returns an accurate solution—one of low KL-divergence from the true distribution—given a *polynomial number of training examples*.

In contrast to our new parameterization, the factors in a factor graph (or a Markov network) are typically considered to have no probabilistic interpretation at all. One exception is the canonical parameterization used in the Hammersley-Clifford theorem for Markov networks (Hammersley and Clifford, 1971; Besag, 1974b). The Hammersley-Clifford canonical parameterization expresses the distribution as a product of probabilities over *all* variables. However, the number of different instantiations is exponential in the number of variables. Therefore such probabilities over all variables cannot be estimated accurately from a small number of training examples. As a consequence the Hammersley-Clifford canonical parameterization is not suited for parameter learning.

Our closed-form parameter learning algorithm is the first polynomial-time and polynomial sample-complexity parameter learning algorithm for factor graphs of bounded degree, and thereby for Markov networks of bounded degree. In contrast, we do not know how to do maximum likelihood (ML) estimation in Markov networks or factor graphs without evaluating the likelihood. Evaluating the likelihood is equivalent to evaluating the partition function. Evaluating the partition function is known to be NP-hard, both exactly and approximately (Jerrum and Sinclair, 1993; Barahona, 1982). Indeed, all known exact algorithms grow exponentially in the tree-width of the graph, making the computation of the partition function intractable for many, even moderately sized, factor graphs. (See, for example, Cowell et al., 1999, for more details on such exact algorithms.) For example, n by n grids over binary variables (which have degree bounded by 4, independently of n) have tree-width n and the computational complexity of known algorithms for computing the partition function (and thus of known ML algorithms) is $O(2^n)$.

We analyze the sample complexity of parameter learning as a function of the number of variables in the network. We show that (under some mild assumptions) the sample complexity of parameter learning in graphical models has on $O(1)$ dependence on the number of variables in the graphical model when using KL-divergence normalized by the number of variables as the performance criterion. This result is important since it gives theoretical support for the common practice of learning large graphical models from a relatively small number of training examples. More specifically, the number of training examples can be much smaller than the number of parameters when learning large graphical models.

(2001), for more details on the equivalence and conversion between factor graphs, Bayesian networks and Markov networks.

3. For a pairwise Markov network with degree of the undirected graph bounded by d, the local subsets are of size at most $2d$.

Building on our closed-form parameter learning algorithm, we provide an algorithm for learning not only the parameters, but also the *structure*. In our new parameterization, factors that are not present in the distribution can be computed in the same way from local probabilities as factors that are present in the distribution. As will become clear later, a key property of our new parameterization is that the factors not present in the distribution have all entries equal to one. This gives a very simple test to decide whether or not a factor is present in the distribution. Thus no iterative search procedure—as is common for most structure learning algorithms—is needed. However, to compute all the factors from local probabilities, we need to know which variables are its neighbors. So to complete the structure learning algorithm, we need to show how to find each factor's neighbors. We show that local independence tests can be used to find the neighbors of each factor. Since local independence tests use statistics over a small number of variables only, the neighbors can be found efficiently from a small number of training examples.

Our structure learning algorithm provides the first polynomial-time and polynomial sample-complexity structure learning algorithm for factor graphs, and thereby for Markov networks. Note that our algorithm applies to any factor graph of bounded degree, including those (such as grids) where inference is intractable.

We also show that our algorithms degrade gracefully, in that they return reasonable answers even when the underlying distribution does not come exactly from the target class of networks.

We note that the proposed algorithms are unlikely to be useful in practice in their current form. The structure learning algorithm does an exhaustive enumeration over the possible neighbor sets of factors in the factor graph, a process which is—although polynomial—generally infeasible even in moderately sized networks. Both the parameter and the structure learning algorithm do not make good use of all the available data. Nevertheless, the techniques used in our analysis open new avenues towards efficient parameter and structure learning in undirected, intractable models.

The remainder of this paper is organized as follows. Section 2 provides necessary background about Gibbs distributions, the factor graph associated with a Gibbs distribution, Markov blankets and the Hammersley-Clifford canonical parameterization. In its original form, the Hammersley-Clifford theorem applies to Markov networks only. We provide an extension that applies to factor graphs. In Section 3, building on the canonical parameterization for factor graphs, we derive our novel parameterization, which forms the basis of our parameter estimation algorithm. We present our algorithm and provide formal running time and sample complexity guarantees. We conclude the section with an in-depth analysis of the relationship between the sample complexity and the number of random variables. In Section 4, we present our structure learning algorithm, and its formal guarantees. Section 5 discusses related work. For clarity of exposition, we provide the complete proofs of all theorems and propositions in the appendix.

Table 1 gives an overview of the notation we use throughout this paper.

2. Preliminaries

In this section we first introduce Gibbs distributions, the factor graph associated with a Gibbs distribution, Markov blankets and the canonical parameterization. Then we present an extension of the Hammersley-Clifford theorem—which in its original form only applies to Markov networks—to factor graphs. Throughout the paper we restrict attention to discrete probability distributions over finite sets.

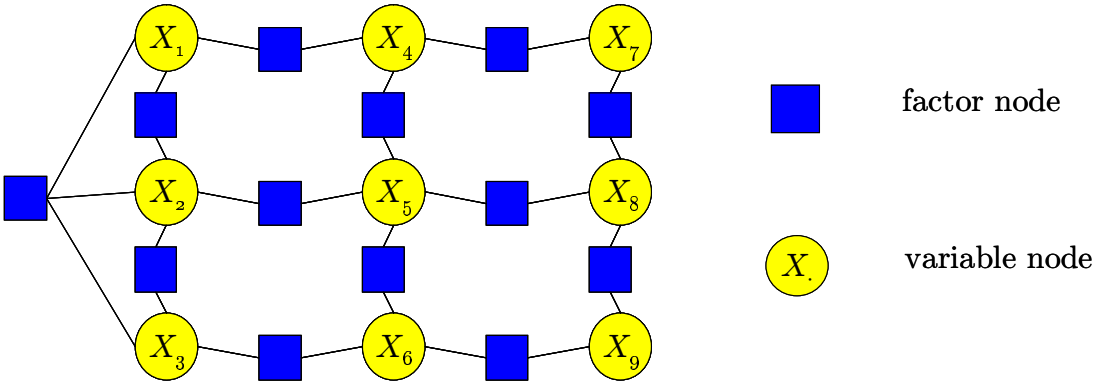


Figure 1: Example factor graph.

2.1 Gibbs Distributions

The probability distributions we consider are referred to as Gibbs distributions.

Definition 1 (Gibbs distribution) *A* factor f *with* scope[4] $\mathbf{D}$ *is a mapping from* val($\mathbf{D}$) *to* $\mathbb{R}^+$. *A* Gibbs *distribution P over a set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ is associated with a set of factors $\{f_j\}_{j=1}^J$ with scopes $\{\mathbf{C}_j\}_{j=1}^J$, such that*

$$P(X_1 = x_1, \ldots, X_n = x_n) = \frac{1}{Z} \prod_{j=1}^{J} f_j(\mathbf{C}_j[x_1, \ldots, x_n]).$$

The normalizing constant Z is the partition function.

The *factor graph* associated with a Gibbs distribution is a bipartite graph whose nodes correspond to variables and factors, with an edge between a variable X and a factor f_j if the scope of f_j contains X. There is one-to-one correspondence between factor graphs and the sets of scopes. Figure 1 gives an example of a factor graph. Here the Gibbs distribution is over the variables $X_1, \cdots, X_9$, which are represented by circles in the factor graph. The factors are represented by squares and have the following respective scopes: $\{X_1, X_2, X_3\}$, $\{X_1, X_2\}$, $\{X_2, X_3\}$, $\{X_1, X_4\}$, $\{X_2, X_5\}$, $\{X_3, X_6\}$, $\{X_4, X_5\}$, $\{X_5, X_6\}$, $\{X_4, X_7\}$, $\{X_5, X_8\}$, $\{X_7, X_9\}$, $\{X_7, X_8\}$, $\{X_8, X_9\}$. The corresponding Gibbs distribution is given by

$$P(X_1 = x_1, \cdots, X_9 = x_9) = \frac{1}{Z} f_{\{X_1, X_2, X_3\}}(x_1, x_2, x_3) f_{\{X_1, X_2\}}(x_1, x_2) \cdots f_{\{X_8, X_9\}}(x_8, x_9).$$

A Gibbs distribution also induces a Markov network—an undirected graph whose nodes correspond to the random variables $\mathcal{X}$ and where there is an edge between two variables if there is a factor in which they both participate. The set of scopes uniquely determines the structure of the Markov network, but several different sets of scopes can result in the same Markov network. For example, a fully connected Markov network can correspond both to a Gibbs distribution with $\binom{n}{2}$ factors over pairs of variables, and to a distribution with a factor which is a joint distribution over $\mathcal{X}$. We will

4. A function has *scope* $\mathbf{X}$ if its domain is val($\mathbf{X}$), the set of possible instantiations of the set of random variables $\mathbf{X}$.

use the more precise factor graph representation in this paper. Our results are easily translated into results for Markov networks.

Definition 2 (Markov blanket) *Let a set of scopes $\mathcal{C} = \{\mathbf{C}_j\}_{j=1}^{J}$ be given. The* Markov blanket *of a set of random variables $\mathbf{D} \subseteq \mathcal{X}$ is defined as*

$$\mathrm{MB}(\mathbf{D}) = \cup\{\mathbf{C}_j : \mathbf{C}_j \in \mathcal{C}\ ,\ \mathbf{C}_j \cap \mathbf{D} \neq \emptyset\} - \mathbf{D}.$$

Thus, the Markov blanket of a set of variables $\mathbf{D}$ is the minimal set of variables that separates $\mathbf{D}$ from the other variables in the factor graph. For the factor graph distribution of Figure 1 we have, for example, $\mathrm{MB}(\{X_1\}) = \{X_2, X_3, X_4\}$, $\mathrm{MB}(\{X_1, X_2\}) = \{X_3, X_4, X_5\}$, and $\mathrm{MB}(\{X_5\}) = \{X_2, X_4, X_6, X_8\}$.

For any Gibbs distribution, we have, for any subset of random variables $\mathbf{D}$, that

$$\mathbf{D} \perp \mathcal{X} - \mathbf{D} - \mathrm{MB}(\mathbf{D}) \mid \mathrm{MB}(\mathbf{D}), \tag{1}$$

or in words: given its Markov blanket $\mathrm{MB}(\mathbf{D})$, the set of variables $\mathbf{D}$ is independent of all other variables $\mathcal{X} - \mathbf{D} - \mathrm{MB}(\mathbf{D})$.[5]

A standard assumption for a Gibbs distribution, which is critical for identifying its structure (see Lauritzen, 1996, Ch. 3), is that the distribution be *positive*—all of its entries be non-zero. Our results use a quantitative measure for how positive P is. Let $\gamma = \min_{\mathbf{x},i} P(X_i = x_i | \mathcal{X}_{-i} = \mathbf{x}_{-i})$, where the $-i$ subscript denotes all entries but entry i. Note that, if we have a fixed bound on the number of factors in which a variable can participate, a fixed bound on the domain size for each variable, and a fixed bound on how skewed each factor is (more specifically a bound on the ratio of its lowest and highest entries), we are guaranteed a bound on γ that is independent of the number n of variables in the network. Thus, under these assumptions, our sample complexity results, which are expressed as a function of γ, have *no* hidden dependence on the number of variables n. In contrast, $\tilde{\gamma} = \min_{\mathbf{x}} P(\mathbf{X} = \mathbf{x})$ generally has an exponential dependence on n. For example, if we have n independent and identically distributed (i.i.d.) Bernoulli($\frac{1}{2}$) random variables, then $\gamma = \frac{1}{2}$ (independent of n) but $\tilde{\gamma} = \frac{1}{2^n}$.

2.2 The Canonical Parameterization

A Gibbs distribution is generally over-parameterized relative to the structure of the underlying factor graph, in that a continuum of possible parameterizations over the graph can all encode the same distribution. The *canonical parameterization* (Hammersley and Clifford, 1971; Besag, 1974b) provides one specific choice of parameterization for a Gibbs distribution, with some nice properties (see below). The canonical parameterization forms the basis for the Hammersley-Clifford theorem, which asserts that any distribution that satisfies the independence assumptions encoded by a Markov network can be represented as a Gibbs distribution with factors corresponding to each of the cliques in the Markov network. In its original formulation, the canonical distribution is defined for Gibbs distributions over Markov networks. We use a more refined parameterization, defined at the factor level; results at the clique level (or, equivalently, results for Markov networks) are trivial corollaries.

The canonical parameterization is defined relative to an arbitrary (but fixed) set of "default" assignments $\bar{\mathbf{x}} = (\bar{x}_1, \ldots, \bar{x}_n)$. Let any subset of variables $\mathbf{D} = \langle X_{i_1}, \ldots, X_{i_{|\mathbf{D}|}} \rangle$, and any assignment

5. By $\mathbf{X} \perp \mathbf{Y}$ we denote that $\mathbf{X}$ is independent of $\mathbf{Y}$. By $\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}$ we denote that $\mathbf{X}$ is conditionally independent of $\mathbf{Y}$ given $\mathbf{Z}$.

$\mathbf{d} = \langle x_{i_1}, \ldots, x_{i_{|\mathbf{D}|}} \rangle$ be given. Let any $\mathbf{U} \subseteq \mathbf{D}$ be given. We define $\sigma_{\cdot}[\cdot]$ such that for all $i \in \{1, \ldots, n\}$:

$$(\sigma_{\mathbf{U}}[\mathbf{d}])_i = \begin{cases} x_i & \text{if } X_i \in \mathbf{U}, \\ \bar{x_i} & \text{if } X_i \notin \mathbf{U}. \end{cases}$$

In words, $\sigma_{\mathbf{U}}[\mathbf{d}]$ keeps the assignments to the variables in $\mathbf{U}$ as specified in $\mathbf{d}$, and augments it to form a full assignment using the default values in $\bar{\mathbf{x}}$. Note that the assignments to variables outside $\mathbf{U}$ are always ignored, and replaced with their default values. Thus, the scope of $\sigma_{\mathbf{U}}[\cdot]$ is always $\mathbf{U}$.

Let P be a positive Gibbs distribution.The *canonical factor* for $\mathbf{D} \subseteq \mathcal{X}$ is defined as follows:

$$f^*_{\mathbf{D}}(\mathbf{d}) = \exp\left(\textstyle\sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{U}|} \log P(\sigma_{\mathbf{U}}[\mathbf{d}])\right). \tag{2}$$

The sum is over all subsets of $\mathbf{D}$, including $\mathbf{D}$ itself and the empty set $\emptyset$.

The following theorem extends the Hammersley-Clifford theorem (which applies to Markov networks) to factor graphs.

Theorem 3 *Let P be a positive Gibbs distribution with factor scopes $\{\mathbf{C}_j\}_{j=1}^J$. Let $\{\mathbf{C}^*_j\}_{j=1}^{J^*} = \cup_{j=1}^J 2^{\mathbf{C}_j} - \emptyset$ (where $2^{\mathbf{X}}$ is the power set of $\mathbf{X}$—the set of all of its subsets). Then*

$$P(\mathbf{x}) = P(\bar{\mathbf{x}}) \textstyle\prod_{j=1}^{J^*} f^*_{\mathbf{C}^*_j}(\mathbf{c}^*_j),$$

where $\mathbf{c}^*_j$ is the instantiation of $\mathbf{C}^*_j$ consistent with $\mathbf{x}$.

The proof is in the appendix.

The parameterization of P using the canonical factors $\{f^*_{\mathbf{C}^*_j}\}_{j=1}^{J^*}$ is called the *canonical parameterization* of P. Although typically $J^* > J$, the additional factors are all subfactors of the original factors. Note that first transforming a factor graph into a Markov network and then applying the Hammersley-Clifford theorem to the Markov network generally results in a significantly less sparse canonical parameterization than the canonical parameterization from Theorem 3.

We now give an example to clarify the definition of canonical factors and canonical parameterization.

Example 1 *Consider again the factor graph of Figure 1. Assume we take the fixed assignment to be all zeros, namely we have $\bar{x_1} = 0, \bar{x_2} = 0, \cdots, \bar{x_9} = 0$. Then the canonical factor $f^*_{\{X_1,X_2\}}$ over the variables X_1, X_2 instantiated to x_1, x_2 is given by*

$$\begin{aligned} \log f^*_{\{X_1,X_2\}}(x_1,x_2) \;=\; & \log P(X_1 = x_1, X_2 = x_2, X_3 = 0, X_4 = 0, \cdots, X_9 = 0) \\ & - \log P(X_1 = 0, X_2 = x_2, X_3 = 0, X_4 = 0, \cdots, X_9 = 0) \\ & - \log P(X_1 = x_1, X_2 = 0, X_3 = 0, X_4 = 0, \cdots, X_9 = 0) \\ & + \log P(X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 0, \cdots, X_9 = 0). \end{aligned} \tag{3}$$

So to compute the canonical factor, we start with the joint instantiation of the factor variables $\{X_1, X_2\}$ with all other variables $\{X_3, \cdots, X_9\}$ set to their default instantiations. Then we subtract out the instantiations for which one of the factor variables is changed to its default instantiation. Crudely speaking, we subtract out the interactions that are already captured by a canonical factor over a smaller set of variables. Then we adjust for double counting by adding back in the instantiation where both factor variables have been set to their default instantiation.

Similarly, the canonical factor $f^*_{\{X_1,X_2,X_3\}}$ over the variables X_1,X_2,X_3 instantiated to x_1,x_2,x_3 is given by

$$\begin{aligned}\log f^*_{\{X_1,X_2,X_3\}}(x_1,x_2,x_3) &= \log P(X_1=x_1,X_2=x_2,X_3=x_3,X_4=0,\cdots,X_9=0)\\ &\quad -\log P(X_1=0,X_2=x_2,X_3=x_3,X_4=0,\cdots,X_9=0)\\ &\quad -\log P(X_1=x_1,X_2=0,X_3=x_3,X_4=0,\cdots,X_9=0)\\ &\quad -\log P(X_1=x_1,X_2=x_2,X_3=0,X_4=0,\cdots,X_9=0)\\ &\quad +\log P(X_1=0,X_2=0,X_3=x_3,X_4=0,\cdots,X_9=0)\\ &\quad +\log P(X_1=0,X_2=x_2,X_3=0,X_4=0,\cdots,X_9=0)\\ &\quad +\log P(X_1=x_1,X_2=0,X_3=0,X_4=0,\cdots,X_9=0)\\ &\quad -\log P(X_1=0,X_2=0,X_3=0,X_4=0,\cdots,X_9=0).\end{aligned}$$

The canonical factor over just the variable X_1 instantiated to x_1 is given by

$$\begin{aligned}\log f^*_{\{X_1\}}(x_1) &= \log P(X_1=x_1,X_2=0,X_3=0,X_4=0,\cdots,X_9=0)\\ &\quad -\log P(X_1=0,X_2=0,X_3=0,X_4=0,\cdots,X_9=0).\end{aligned}$$

Theorem 3 applied to our example gives the following expression for the probability distribution:

$$\begin{aligned}P(X_1=x_1,\cdots,X_9=x_9) &= P(X_1=0,\cdots,X_9=0)\\ &\quad \times f^*_{\{X_1,X_2,X_3\}}(x_1,x_2,x_3)\\ &\quad \times f^*_{\{X_1,X_2\}}(x_1,x_2)f^*_{\{X_2,X_3\}}(x_2,x_3)\cdots f^*_{\{X_8,X_9\}}(x_8,x_9)\\ &\quad \times f^*_{\{X_1\}}(x_1)f^*_{\{X_2\}}(x_2)\cdots f^*_{\{X_9\}}(x_9)\\ &= \frac{1}{Z}\\ &\quad \times f^*_{\{X_1,X_2,X_3\}}(x_1,x_2,x_3)\\ &\quad \times f^*_{\{X_1,X_2\}}(x_1,x_2)f^*_{\{X_2,X_3\}}(x_2,x_3)\cdots f^*_{\{X_8,X_9\}}(x_8,x_9)\\ &\quad \times f^*_{\{X_1\}}(x_1)f^*_{\{X_2\}}(x_2)\cdots f^*_{\{X_9\}}(x_9). \qquad (4)\end{aligned}$$

3. Parameter Estimation

In this section we first introduce the parameter estimation ideas informally by expanding on Example 1. Then we formally introduce the key idea of Markov blanket canonical factors, which give a parameterization of a factor graph distribution only in terms of local probabilities. This new parameterization directly results in the proposed parameter estimation algorithm. We analyze the algorithm's computational and sample complexity. In addition, we show an $O(1)$ dependence on the number of variables in the network for the sample complexity when using the KL-divergence normalized by the number of variables in the network as performance criterion.

3.1 Parameter Estimation by Example

Consider the problem of estimating the parameters of the distribution in Figure 1 from training examples. From Eqn. (4) we have that it is sufficient to estimate all the canonical factors. Each canonical factor is expressed in terms of probabilities. So one could estimate the canonical factors

(and thus the distribution) in closed-form by estimating these probabilities from data. Unfortunately the probabilities appearing in the canonical factors are over *full joint instantiations of all variables*. As a consequence, these probabilities can not be estimated accurately from a small amount of data.

However, we will now consider the factor $f^*_{\{X_1,X_2\}}$ more carefully and show it can be estimated from probabilities over small subsets of the variables only. The factor $f^*_{\{X_1,X_2\}}$ contains an equal number of terms with positive and negative sign. For the sum of two such terms, we now derive a novel expression which contains local probabilities only (instead of probabilities of full joint instantiations of all variables).

$$\begin{aligned}
& \log P(X_1 = x_1, X_2 = x_2, X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0) \\
& - \log P(X_1 = x_1, X_2 = 0, X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0) \\
= \; & \log P(X_1 = x_1, X_2 = x_2 | X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0) \\
& + \log P(X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0) \\
& - \log P(X_1 = x_1, X_2 = 0 | X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0) \\
& - \log P(X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0) \\
= \; & \log P(X_1 = x_1, X_2 = x_2 | X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0) \\
& - \log P(X_1 = x_1, X_2 = 0 | X_3 = 0, X_4 = 0, X_5 = 0, X_6 = 0, X_7 = 0, X_8 = 0, X_9 = 0) \\
= \; & \log P(X_1 = x_1, X_2 = x_2 | \mathrm{MB}(\{X_1, X_2\}) = \vec{0}) \\
& - \log P(X_1 = x_1, X_2 = 0 | \mathrm{MB}(\{X_1, X_2\}) = \vec{0}) \\
= \; & \log P(X_1 = x_1, X_2 = x_2 | X_3 = 0, X_4 = 0, X_5 = 0) \\
& - \log P(X_1 = x_1, X_2 = 0 | X_3 = 0, X_4 = 0, X_5 = 0). \qquad (5)
\end{aligned}$$

Here we used in order: the definition of conditional probability; same terms with opposite sign cancel; conditioning on the Markov blanket is equivalent to conditioning on all other variables; $\mathrm{MB}(\{X_1,X_2\}) = \{X_3,X_4,X_5\}$ in our example.

The last expression in Eqn. (5) contains local probabilities only, which can be estimated accurately from a small number of training examples. Using a similar reasoning as above for the other two terms of the factor $f^*_{\{X_1,X_2\}}$, we get the following expression for $f^*_{\{X_1,X_2\}}$, which contains local probabilities only:

$$\begin{aligned}
\log f^*_{\{X_1,X_2\}}(x_1,x_2) &= \log P(X_1 = x_1, X_2 = x_2 | X_3 = 0, X_4 = 0, X_5 = 0) \\
&\quad - \log P(X_1 = x_1, X_2 = 0 | X_3 = 0, X_4 = 0, X_5 = 0) \\
&\quad - \log P(X_1 = 0, X_2 = x_2 | X_3 = 0, X_4 = 0, X_5 = 0) \\
&\quad + \log P(X_1 = 0, X_2 = 0 | X_3 = 0, X_4 = 0, X_5 = 0) \\
&= \log f^*_{\{X_1,X_2\}|\{X_3,X_4,X_5\}}(x_1,x_2). \qquad (6)
\end{aligned}$$

The last line defines $f^*_{\{X_1,X_2\}|\{X_3,X_4,X_5\}}(x_1,x_2)$ (which we refer to as the Markov blanket canonical factor for $\{X_1,X_2\}$). Although $f^*_{\{X_1,X_2\}}(x_1,x_2) = f^*_{\{X_1,X_2\}|\{X_3,X_4,X_5\}}(x_1,x_2)$ when exact probabilities are used, we use different notation to explicitly distinguish how they are computed from probabilities. The Markov blanket canonical factor $f^*_{\{X_1,X_2\}|\{X_3,X_4,X_5\}}(x_1,x_2)$ is computed from local probabilities as given in Eqn. (6). The (original) canonical factor $f^*_{\{X_1,X_2\}}(x_1,x_2)$ is computed from probabilities over full joint instantiations as given in Eqn. (3).

Similarly, the other canonical factors have equivalent Markov blanket canonical factors which involve local probabilities only. This gives us an efficient closed-form parameter estimation algorithm for our example. In the next few sections we formalize this idea for general factor graphs and analyze the computational and sample complexity.

3.2 Markov Blanket Canonical Factors

Considering the definition of the canonical parameters, we note that all of the terms in Eqn. (2) can be estimated from empirical data using simple counts, without requiring inference over the network. Thus, it appears that we can use the canonical parameterization as the basis for our parameter estimation algorithm. However, as written, this estimation process is statistically infeasible, as the terms in Eqn. (2) are probabilities over full instantiations of all variables, which can never be estimated from a reasonable number of training examples.

We now generalize our observation from the example in the previous section: namely, that we can express the canonical factors using only probabilities over much smaller instantiations—those corresponding to a factor and its Markov blanket. Let $\mathbf{D} = \langle X_{i_1}, \ldots, X_{i_{|\mathbf{D}|}} \rangle$ be any subset of variables, and $\mathbf{d} = \langle x_{i_1}, \ldots, x_{i_{|\mathbf{D}|}} \rangle$ be any assignment to $\mathbf{D}$. For any $\mathbf{U} \subseteq \mathbf{D}$, we define $\sigma_{\mathbf{U}:\mathbf{D}}[\mathbf{d}]$ to be the restriction of the full instantiation $\sigma_{\mathbf{U}}[\mathbf{d}]$ of all variables in $\mathcal{X}$ to the corresponding instantiation of the subset $\mathbf{D}$. In other words, $\sigma_{\mathbf{U}:\mathbf{D}}[\mathbf{d}]$ keeps the assignments to the variables in $\mathbf{U}$ as specified in $\mathbf{d}$, and changes the assignment to the variables in $\mathbf{D} - \mathbf{U}$ to the default values in $\bar{\mathbf{x}}$. Let $\mathbf{D} \subseteq \mathcal{X}$ and $\mathbf{Y} \subseteq \mathcal{X} - \mathbf{D}$. Then the factor $f^*_{\mathbf{D}|\mathbf{Y}}$ over the variables in $\mathbf{D}$ is defined as follows:

$$f^*_{\mathbf{D}|\mathbf{Y}}(\mathbf{d}) = \exp\left(\textstyle\sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{U}|} \log P(\sigma_{\mathbf{U}:\mathbf{D}}[\mathbf{d}] | \mathbf{Y} = \bar{\mathbf{y}})\right), \tag{7}$$

where the sum is over all subsets of $\mathbf{D}$, including $\mathbf{D}$ itself and the empty set $\emptyset$.

For example, we have that $f^*_{\{X_1,X_2\}|\{X_3,X_4,X_5\}}$ of the factor graph in Figure 1 is given by Eqn. (6) in the previous section.

The following proposition shows an equivalence between the factors computed using Eqn. (2) and Eqn. (7).

Proposition 4 *Let P be a positive Gibbs distribution with factor scopes $\{\mathbf{C}_j\}_{j=1}^{J}$, and $\{\mathbf{C}^*_j\}_{j=1}^{J^*}$ as above (i.e., $\{\mathbf{C}^*_j\}_{j=1}^{J^*} = \cup_{j=1}^{J} 2^{\mathbf{C}_j} - \emptyset$). Then for any $\mathbf{D} \subseteq \mathcal{X}$, we have:*

$$f^*_{\mathbf{D}} = f^*_{\mathbf{D}|\mathcal{X}-\mathbf{D}} = f^*_{\mathbf{D}|\mathrm{MB}(\mathbf{D})}, \tag{8}$$

and (as a direct consequence)

$$P(\mathbf{x}) = P(\bar{\mathbf{x}}) \textstyle\prod_{j=1}^{J^*} f^*_{\mathbf{C}^*_j|\mathcal{X}-\mathbf{C}^*_j}(\mathbf{c}^*_j) \tag{9}$$

$$= P(\bar{\mathbf{x}}) \textstyle\prod_{j=1}^{J^*} f^*_{\mathbf{C}^*_j|\mathrm{MB}(\mathbf{C}^*_j)}(\mathbf{c}^*_j), \tag{10}$$

where $\mathbf{c}^*_j$ is the instantiation of $\mathbf{C}^*_j$ consistent with $\mathbf{x}$.

Proposition 4 shows that we can compute the canonical parameterization factors using probabilities over factor scopes and their Markov blankets only. From a sample complexity point of view, this is a significant improvement over the standard definition which uses joint instantiations over all variables. Using Eqn. (7) we can expand the Markov blanket canonical factors in Proposition 4 and we see that *any factor graph distribution can be parameterized as a product of local probabilities only*.

$X, Y, \ldots$	random variables
$x, y, \ldots$	instantiations of the random variables
$\mathbf{X}, \mathbf{Y}, \ldots$	sets of random variables
$\mathbf{x}, \mathbf{y}, \ldots$	instantiations of sets of random variables
$\text{val}(X)$	set of values the variable X can take
$\mathbf{D}[\mathbf{x}]$	instantiation of $\mathbf{D}$ consistent with $\mathbf{x}$ (abbreviated as $\mathbf{d}$ when no ambiguity is possible)
$\mathbf{X} \perp \mathbf{Y}$	$\mathbf{X}$ is independent of $\mathbf{Y}$
$\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}$	$\mathbf{X}$ is conditionally independent of $\mathbf{Y}$ given $\mathbf{Z}$
f	factor
P	positive Gibbs distribution over a set of random variables $\mathcal{X} = \langle X_1, \ldots, X_n \rangle$
$\{f_j\}_{j=1}^J$	factors of P
$\{\mathbf{C}_j\}_{j=1}^J$	scopes of factors of P
$\hat{P}$	empirical (sample) distribution
$\tilde{P}$	distribution returned by learning algorithm
$f^*_{\cdot}$	canonical factor as defined in Eqn. (2)
$f^*_{\cdot\|\cdot}$	canonical factor as defined in Eqn. (7)
$\hat{f}^*_{\cdot\|\cdot}$	canonical factor as defined in Eqn. (7), but using the empirical distribution $\hat{P}$
$\text{MB}(\mathbf{D})$	Markov blanket of $\mathbf{D}$
k	$\max_j \lvert\mathbf{C}_j\rvert$
γ	$\min_{\mathbf{x},i} P(X_i = x_i \mid \mathcal{X}_{-i} = \mathbf{x}_{-i})$
v	$\max_i \lvert\text{val}(\text{X}_\text{i})\rvert$
b	$\max_j \lvert\text{MB}(\mathbf{C}_j)\rvert$
m	number of training examples
$D(\cdot\|\cdot)$	KL-divergence, $D(P\|Q) = \sum_{\mathbf{x} \in \text{val}\mathcal{X}} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})}$
$\mathcal{C}$	the set of candidate factor scopes for the structure learning algorithm, Factor-Graph-Structure-Learn ($\mathcal{C} = \{\mathbf{C}_j^* : \mathbf{C}_j^* \subseteq \mathcal{X}, \mathbf{C}_j^* \neq \emptyset, \lvert\mathbf{C}_j^*\rvert \leq k\}$)

Table 1: Notational conventions.

3.3 Parameter Estimation Algorithm

Based on the parameterization above, we propose the following Factor-Graph-Parameter-Learn algorithm. The algorithm takes as inputs: the scopes of the factors $\{\mathbf{C}_j\}_{j=1}^J$, training examples $\{\mathbf{x}^{(i)}\}_{i=1}^m$, a baseline instantiation $\bar{\mathbf{x}}$. Then for $\{\mathbf{C}_j^*\}_{j=1}^{J^*}$ as above (i.e., $\{\mathbf{C}_j^*\}_{j=1}^{J^*} = \cup_{j=1}^J 2^{\mathbf{C}_j} - \emptyset$), Factor-Graph-Parameter-Learn does the following:

- Compute the estimates of the canonical factors $\{\hat{f}^*_{\mathbf{C}_j^* \mid \text{MB}(\mathbf{C}_j^*)}\}_{j=1}^{J^*}$ as in Eqn. (7), but using the empirical estimates based on the training examples.
- Return the probability distribution $\tilde{P}(\mathbf{x}) \propto \prod_{j=1}^{J^*} \hat{f}^*_{\mathbf{C}_j^* \mid \text{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)$.

Theorem 5 (Parameter learning: computational complexity) *The running time of the* Factor-Graph-Parameter-Learn *algorithm is in* $O(m2^kJ(k+b)+2^{2k}Jv^k)$.[6]

The proof is given in the appendix.

Note the representation of the factor graph distribution is $\Omega(Jv^k)$, thus exponential dependence on k is unavoidable for any algorithm. More importantly, there is no dependence on the running time of evaluating the partition function. On the other hand, all currently known maximum likelihood estimation algorithms require evaluating the partition function, which is known to be NP-hard, both exactly and approximately (Jerrum and Sinclair, 1993; Barahona, 1982).

3.4 Sample Complexity

We now analyze the sample complexity of the Factor-Graph-Parameter-Learn algorithm, showing that it returns a distribution that is a good approximation of the true distribution when given only a "small" number of training examples. We will use the sum of KL-divergences $D(P\|\tilde{P})+D(\tilde{P}\|P)$ to measure how well the distribution $\tilde{P}$ approximates the distribution P.[7]

Theorem 6 (Parameter learning: sample complexity) *Let any* $\varepsilon,\delta>0$ *be given. Let* Factor-Graph-Parameter-Learn *be given (a) m training examples* $\{\mathbf{x}^{(i)}\}_{i=1}^m$ *drawn i.i.d. from a distribution P and (b) the factor graph structure according to which the distribution P factors. Let* $\tilde{P}$ *be the probability distribution returned by* Factor-Graph-Parameter-Learn*. Then, we have that, for*

$$D(P\|\tilde{P})+D(\tilde{P}\|P)\leq J\varepsilon$$

to hold with probability at least $1-\delta$*, it suffices that the number of training examples m satisfies:*

$$m\geq(1+\tfrac{\varepsilon}{2^{2k+2}})^2\ \tfrac{2^{4k+3}}{\gamma^{2k+2b}\varepsilon^2}\log\tfrac{2^{k+2}Jv^{k+b}}{\delta}. \tag{11}$$

A complete proof is given in the appendix.

Theorem 6 shows that—assuming the true distribution P factors according to the given structure—Factor-Graph-Parameter-Learn returns a distribution that is $J\varepsilon$-close in KL-divergence. The sample complexity scales exponentially in the maximum number of variables per factor k, and polynomially in $\frac{1}{\varepsilon},\frac{1}{\gamma}$.

The error in the KL-divergence grows linearly with the number of factors J. This is a consequence of the fact that the number of terms in the distributions is equal to the number of factors J, and each term can accrue an error. We can obtain a more refined analysis if we eliminate this dependence by considering the KL-divergence normalized by the number of variables, $D_n(P\|\tilde{P})=\frac{1}{n}D(P\|\tilde{P})$. We return to this topic in Section 3.5.

We now sketch the proof idea. The Markov blanket canonical factors are a product of local conditional probabilities. These local conditional probabilities can be estimated accurately from a "small" number of training examples. Thus the Markov blanket canonical factors can be estimated accurately from a small number of training examples. Thus the factor graph distribution—which is just a product of the Markov canonical factors—can be estimated accurately from a small number of training examples.

6. The upper bound is based on a very naive implementation's running time. It assumes that operations on numbers (such as reading, writing, adding, etc.) take constant time.

7. $D(P\|Q)=\sum_{\mathbf{x}\in\mathrm{val}\mathcal{X}}P(\mathbf{x})\log\frac{P(\mathbf{x})}{Q(\mathbf{x})}$.

Theorem 6 considers the case when P factors according to the given structure. The following theorem shows that our error degrades gracefully even if the training examples are generated by a distribution Q that does not factor according to the given structure.

Theorem 7 (Parameter learning: graceful degradation) *Let any $\varepsilon, \delta > 0$ be given. Let $\{\mathbf{x}^{(i)}\}_{i=1}^m$ be i.i.d. samples from a distribution Q. Let* MB *and* $\widehat{\text{MB}}$ *be the Markov blankets according to the distribution Q and the given structure respectively. Let $\{f^*_{\mathbf{D}^*_j|\text{MB}(\mathbf{D}^*_j)}\}_{j=1}^J$ be the non-trivial Markov blanket canonical factors of Q (those factors with not all entries equal to one). Let $\{\mathbf{C}^*_j\}_{j=1}^{J^*}$ be the scopes of the canonical factors in the factor graph given to the algorithm. Let $\tilde{P}$ be the probability distribution returned by* Factor-Graph-Parameter-Learn*. Then we have that for*

$$D(Q\|\tilde{P}) + D(\tilde{P}\|Q) \leq J\varepsilon + 2\sum_{j:\mathbf{D}^*_j \notin \{\mathbf{C}^*_k\}_{k=1}^{J^*}} \max_{\mathbf{d}^*_j} \left|\log f^*_{\mathbf{D}^*_j}(\mathbf{d}^*_j)\right| + 2\sum_{j\,:\,\text{MB}(\mathbf{C}^*_j)\neq\widehat{MB}(\mathbf{C}^*_j)} \max_{\mathbf{c}^*_j} \left|\log \frac{f^*_{\mathbf{C}^*_j|\text{MB}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)}{f^*_{\mathbf{C}^*_j|\widehat{MB}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)}\right|$$

to hold with probability at least $1-\delta$, it suffices that the number of training examples m satisfies Eqn. (11) of Theorem 6.

Note the sample complexity depends on parameters $k = \max_j |\mathbf{C}^*_j|$ and $b = \max_j |\text{MB}(\mathbf{C}^*_j)|$ of the given target structure (rather than the true structure). The graceful degradation result is important, as it shows that each canonical factor that is incorrectly captured by our target structure adds at most a constant (namely, $l2^{l+1}\log\frac{1}{\gamma}$ for an incorrectly captured factor over l variables) to our bound on the KL-divergence.[8] This constant can be large, so we discuss the actual error contribution in more detail. A canonical factor could be incorrectly captured when the corresponding factor scope is not included in the given structure. Canonical factors are designed so that a factor over a set of variables captures only the residual interactions between the variables in its scope, once all interactions between its subsets have been accounted for in other factors. Thus, canonical factors over large scopes are often close to the trivial all-ones factor in practice. Therefore, if our structure approximation is such that it only ignores some of the larger-scope factors, the error in the approximation may be quite limited. A canonical factor could also be incorrectly captured when the given structure does not have the correct Markov blanket for that factor. The resulting error depends on how good an approximation of the Markov blanket we do have. See Section 4 for more details on the error caused by incorrect Markov blankets.

3.5 Reducing the Dependence on Network Size

Our previous analysis showed a linear dependence of the sample complexity on the number of factors J in the network (for parameter learning). In a sense, this dependence is inevitable. To understand why, consider a distribution P defined by a set of n independent Bernoulli random variables $X_1, \ldots, X_n$, each with parameter 0.5. Assume that Q is an approximation to P, where the X_i are still independent, but have parameter 0.4999. Intuitively, a Bernoulli(0.4999) distribution is a very good

8. Each factor over l variables is a fraction of a product of 2^{l-1} conditional probabilities over another product of 2^{l-1} conditional probabilities. Recall that $\gamma = \min_{\mathbf{x},i} P(X_i = x_i | \mathcal{X}_{-i} = \mathbf{x}_{-i}) > 0$, so we have that each conditional probability over l variables lies in the interval $[\gamma^l, 1]$. Thus we have for a factor over l variables that $\max_{\mathbf{d}^*_j} \left|\log f^*_{\mathbf{D}^*_j}(\mathbf{d}^*_j)\right| \leq \log \frac{1}{\gamma^{2^{l-1}}} = l2^{l-1}\log\frac{1}{\gamma}$. Similarly, we have that $\max_{\mathbf{c}^*_j} \left|\log \frac{f^*_{\mathbf{C}^*_j|\text{MB}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)}{f^*_{\mathbf{C}^*_j|\widehat{MB}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)}\right| \leq l2^l \log\frac{1}{\gamma}$.

estimate of a Bernoulli(0.5); thus, for most applications, Q can safely be considered to be a very good estimate of P. However, the KL-divergence $D(P(X_{1:n})\|Q(X_{1:n})) = \sum_{i=1}^{n} D(P(X_i)\|Q(X_i)) = \Omega(n)$. Thus, if n is large, the KL divergence between P and Q would be large, even though Q is a good estimate for P. To remove such unintuitive scaling effects when studying the dependence on the number of variables, we can consider instead the normalized KL divergence criterion:

$$D_n(P(X_{1:n})\|Q(X_{1:n})) = \tfrac{1}{n} D(P(X_{1:n})\|Q(X_{1:n})).$$

As we now show, with a slight modification to the algorithm, we can achieve a bound of ε for our normalized KL-divergence while eliminating the logarithmic dependence on J in our sample complexity bound. Specifically, we can modify our algorithm so that it clips probability estimates $\in [0, \gamma^{k+b})$ to γ^{k+b}. The clipping procedure is motivated by the proof of Theorem 8 and effectively ensures that the KL-divergence is bounded.[9] Note that—since true probabilities which we are trying to estimate are never in the interval $[0, \gamma^{k+b})$—this change can only improve the estimates.[10]

For this slightly modified version of the algorithm, the following theorem shows the dependence on the size of the network is $O(1)$, which is tighter than the logarithmic dependence shown in Theorem 6.[11]

Theorem 8 (Parameter learning: size of the network) *Let any $\varepsilon, \delta > 0$ be given and fixed. Let $\{\mathbf{x}^{(i)}\}_{i=1}^{m}$ be i.i.d. samples from P. Let the domain size of each variable be fixed. Let the degree of both the factor and variable nodes be bounded by a fixed constant. Let $\gamma = \min_{\mathbf{x},i} P(X_i = x_i | \mathcal{X}_{-i} = \mathbf{x}_{-i})$ be fixed. Let $\tilde{P}$ be the probability distribution returned by* Factor-Graph-Parameter-Learn. *Then we have that, for*

$$D_n(P\|\tilde{P}) + D_n(\tilde{P}\|P) \leq \varepsilon$$

to hold with probability at least $1-\delta$, it suffices that we have a certain number of training examples that does not depend on the number of variables in the network.

The following theorem shows a similar result for Bayesian networks, namely that for a fixed bound on the number of parents per node, the sample complexity dependence on the size of the network is $O(1)$.[12]

9. In particular, we first show that the error contribution from any fixed factor is small with high probability. Then—rather than using a Union bound to ensure the error contributions from all factors are small, which would result in a logarithmic dependence of the sample complexity on the number of factors (or variables)—we use Markov's inequality to show that the error contribution of almost all factors is small with high probability. This leaves us to bound the error contribution of the (few) remaining factors, for which the error contribution is not small. By clipping the probability estimates, we can ensure their error contribution is bounded. A very similar reasoning applies to the case of Theorem 9. (See the proofs of Theorems 8 and 9, given in the appendix, for more details.)
10. This solution assumes that γ is known. If not, we can use a clipping threshold as a function of the number of training examples. Such an adaptive clipping procedure was used by Dasgupta (1997) to derive sample complexity bounds for learning fixed structure Bayesian networks.
11. We note that Theorem 8 assumes the maximum number of factors a variable can participate in is fixed (i.e., it cannot grow with the number of variables in the network). As a consequence, the dependence on the number of factors J and the dependence on the number of variables n are equivalent (up to a constant factor).
12. Complete proofs for Theorems 8 and 9 (and all other results in this paper) are given in the appendix of this paper. In the appendix we actually give a much stronger version of Theorem 9, including dependencies of m on ε, δ, k and a graceful degradation result. We note that for non-binary random variables the clipping procedure is a bit more subtle than for binary random variables. In particular, to ensure that the resulting clipped probabilities sum to one, we might have to subtract a small quantity from the highest probability estimate after the clipping. For example, for

Theorem 9 *Let any $\varepsilon > 0$ and $\delta > 0$ be given. Let any Bayesian network (BN) structure over n variables with at most k parents per variable be given. Let P be a probability distribution that factors over the BN. Let $\tilde{P}$ denote the probability distribution obtained by fitting the conditional probability tables (CPT) entries via maximum likelihood and then clipping each CPT entry to the interval $[\frac{\varepsilon}{8|\text{val}(X_j)|^3}, 1-\frac{\varepsilon}{8|\text{val}(X_j)|^3}]$. Then we have that for*

$$D_n(P\|\tilde{P}) \leq \varepsilon$$

to hold with probability at least $1-\delta$, it suffices that we have a certain number of training examples that does not depend on the number of variables in the network.

Theorems 8 and 9 provide theoretical support for the common practice of learning large graphical models from a relatively small number of training examples. More specifically, the number of training examples can be much smaller than the number of parameters when learning large graphical models. In contrast, for many problems in machine learning, the sample complexity grows roughly linearly or at most as some low-order polynomial in the number of parameters (Vapnik, 1998). The difference in sample complexity relates to the discussion of generative versus discriminative training. Indeed our result generalizes and even strengthens the results of Ng and Jordan (2002). They showed a logarithmic dependence on the number of variables for the very specific case of a graphical model with the naive Bayes structure.

4. Structure Learning

The algorithm described in the previous section uses the known network to establish a Markov blanket for each factor. This Markov blanket is then used to estimate the canonical parameters from empirical data. In this section, we show how we can build on this algorithm to perform structure learning, by first identifying (from the data) an approximate Markov blanket for each candidate factor, and then using this approximate Markov blanket to compute the parameters of that factor from a "small" number of training examples.

4.1 Identifying Markov Blankets

In the parameter learning results, the Markov blanket $\text{MB}(\mathbf{C}_j^*)$ is used to efficiently estimate the conditional probability $P(\mathbf{C}_j^*|\mathcal{X}-\mathbf{C}_j^*)$, which is equal to $P(\mathbf{C}_j^*|\text{MB}(\mathbf{C}_j^*))$. This suggests to measure the quality of a candidate Markov blanket $\mathbf{Y}$ by how well $P(\mathbf{C}_j^*|\mathbf{Y})$ approximates $P(\mathbf{C}_j^*|\mathcal{X}-\mathbf{C}_j^*)$. In this section we show how conditional entropy can be used to find a candidate Markov blanket that gives a good approximation for this conditional probability.[13]

ε sufficiently small, we have that naively clipping the probability estimates $(0,0,1/4,3/4)$ to the interval $(\varepsilon,1-\varepsilon)$ results in $(\varepsilon,\varepsilon,1/4,3/4)$, which does not sum to one (but rather to $1+2\varepsilon$). Subtracting the additional probability mass 2ε from the highest entry fixes this problem. For this example we get $(\varepsilon,\varepsilon,1/4,3/4-2\varepsilon)$. In general, for v-valued random variables, the probability estimates can be made to sum to one (after clipping) by subtracting at most $(v-1)\varepsilon$ from the highest probability estimate. In the appendix we expand more on the topic of clipping for non-binary random variables.

13. For some readers, some intuition might be gained from the fact that the conditional entropy of $\mathbf{C}_j^*$ given the candidate Markov blanket $\mathbf{Y}$ corresponds to the log-loss of predicting $\mathbf{C}_j^*$ given the candidate Markov blanket $\mathbf{Y}$.

Definition 10 (Conditional Entropy) *Let P be a probability distribution over over $\mathbf{X}, \mathbf{Y}$. Then the conditional entropy $H(\mathbf{X}|\mathbf{Y})$ of $\mathbf{X}$ given $\mathbf{Y}$ is defined as*

$$-\sum_{\mathbf{x}\in \mathrm{val}(\mathbf{X}), \mathbf{y}\in \mathrm{val}(\mathbf{Y})} P(\mathbf{X}=\mathbf{x}, \mathbf{Y}=\mathbf{y}) \log P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}).$$

Proposition 11 (Cover & Thomas, 1991) *Let P be a probability distribution over $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$. Then we have $H(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) \leq H(\mathbf{X}|\mathbf{Y})$.*

Proposition 11 shows that conditional entropy can be used to find the Markov blanket for a given set of variables. Namely, let $\mathbf{D}, \mathbf{Y} \subseteq \mathcal{X}$, $\mathbf{D} \cap \mathbf{Y} = \emptyset$, then we have

$$H(\mathbf{D}|\mathrm{MB}(\mathbf{D})) = H(\mathbf{D}|\mathcal{X} - \mathbf{D}) \leq H(\mathbf{D}|\mathbf{Y}), \tag{12}$$

where the equality follows from the Markov blanket property stated in Eqn. (1) and the inequality follows from Proposition 11. Thus, we can select the set of variables $\mathbf{Y}$ that minimizes $H(\mathbf{D}|\mathbf{Y})$ as our candidate Markov blanket for the set of variables $\mathbf{D}$.

Our first difficulty is that, when learning from data, we do not have the true distribution, and hence the exact conditional entropies are unknown. The following lemma shows that the conditional entropy can be efficiently estimated from samples.

Lemma 12 *Let P be a probability distribution over $\mathbf{X}, \mathbf{Y}$ such that for all instantiations $\mathbf{x}, \mathbf{y}$ we have $P(\mathbf{X}=\mathbf{x}, \mathbf{Y}=\mathbf{y}) \geq \lambda$. Let $\widehat{H}$ be the conditional entropy computed based upon m i.i.d. samples from P. Then for*

$$\left| H(\mathbf{X}|\mathbf{Y}) - \widehat{H}(\mathbf{X}|\mathbf{Y}) \right| \leq \varepsilon$$

to hold with probability $1-\delta$, it suffices that:

$$m \geq \frac{8|\mathrm{val}(\mathbf{X})|^2 |\mathrm{val}(\mathbf{Y})|^2}{\lambda^2 \varepsilon^2} \log \frac{4|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|}{\delta}.$$

However, as the empirical estimates of the conditional entropy are noisy, the true Markov blanket is *not* guaranteed to achieve the minimum of $H(\mathbf{D}|\mathbf{Y})$. In fact, in some probability distributions, many sets of variables could be arbitrarily close to reaching equality in Eqn. (12). Thus, in many cases, our procedure will not recover the actual Markov blanket, when given only a finite number of training examples. Fortunately, as we show in the next lemma, any set of variables $\mathbf{U} \cup \mathbf{W}$ that is close to achieving equality in Eqn. (12) gives an accurate approximation $P(\mathbf{C}_j|\mathbf{U}, \mathbf{W})$ of the probabilities $P(\mathbf{C}_j|\mathcal{X} - \mathbf{C}_j)$ used in the canonical parameterization.

Lemma 13 *Let any $\varepsilon > 0$ be given. Let P be a distribution over disjoint sets of random variables $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{X}, \mathbf{Y}$. Let $\lambda_1 = \min_{\mathbf{u}\in\mathrm{val}(\mathbf{U}), \mathbf{v}\in\mathrm{val}(\mathbf{V}), \mathbf{w}\in\mathrm{val}(\mathbf{W})} P(\mathbf{u}, \mathbf{v}, \mathbf{w})$, and let $\lambda_2 = \min_{\mathbf{x}\in\mathrm{val}(\mathbf{X}), \mathbf{u}\in\mathrm{val}(\mathbf{U}), \mathbf{v}\in\mathrm{val}(\mathbf{V}), \mathbf{w}\in\mathrm{val}(\mathbf{W})} P(\mathbf{x}|\mathbf{u}, \mathbf{v}, \mathbf{w})$. Assume the following holds:*

$$\mathbf{X} \perp \mathbf{Y}, \mathbf{W} \mid \mathbf{U}, \mathbf{V}, \tag{13}$$

$$H(\mathbf{X}|\mathbf{U}, \mathbf{W}) \leq H(\mathbf{X}|\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{Y}) + \varepsilon. \tag{14}$$

Then we have that $\forall\, \mathbf{x}, \mathbf{y}, \mathbf{u}, \mathbf{v}, \mathbf{w}$

$$\left| \log P(\mathbf{x}|\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{y}) - \log P(\mathbf{x}|\mathbf{u}, \mathbf{w}) \right| \leq \frac{\sqrt{2\varepsilon}}{\lambda_2 \sqrt{\lambda_1}}. \tag{15}$$

In other words, if a set of variables $\mathbf{U} \cup \mathbf{W}$ looks like a Markov blanket for $\mathbf{X}$, as evaluated by the conditional entropy $H(\mathbf{X}|\mathbf{U},\mathbf{W})$, then the conditional distribution $P(\mathbf{X}|\mathbf{U},\mathbf{W})$ must be close to the conditional distribution $P(\mathbf{X}|\mathcal{X}-\mathbf{X})$. Thus, it suffices to find such an approximate Markov blanket $\mathbf{U} \cup \mathbf{W}$ as a substitute for knowing the true Markov blanket $\mathbf{U} \cup \mathbf{V}$. This makes conditional entropy suitable for structure learning.

4.2 Structure Learning Algorithm

We propose the following Factor-Graph-Structure-Learn algorithm. The algorithm receives as input: training examples $\{\mathbf{x}^{(i)}\}_{i=1}^{m}$; k: the maximum number of variables per factor; b: the maximum number of variables per Markov blanket for any set of variables up to size k; $\bar{\mathbf{x}}$: a base instantiation.[14]

Let $\mathcal{C}$ be the set of candidate factor scopes, let $\mathcal{Y}$ be the set of candidate Markov blankets. I.e., we have

$$\mathcal{C} = \{\mathbf{C}_j^* : \mathbf{C}_j^* \subseteq \mathcal{X}, \mathbf{C}_j^* \neq \emptyset, |\mathbf{C}_j^*| \leq k\}, \quad (16)$$

$$\mathcal{Y} = \{\mathbf{Y} : \mathbf{Y} \subseteq \mathcal{X}, |\mathbf{Y}| \leq b\}. \quad (17)$$

The algorithm does the following:

- $\forall\, \mathbf{C}_j^* \in \mathcal{C}$, find $\widehat{\mathrm{MB}}(\mathbf{C}_j^*) = \arg\min_{\mathbf{Y} \in \mathcal{Y}, \mathbf{Y} \cap \mathbf{C}_j^* = \emptyset} \widehat{H}(\mathbf{C}_j^*|\mathbf{Y})$, which is the best candidate Markov blanket.
- $\forall\, \mathbf{C}_j^* \in \mathcal{C}$, compute the estimates $\{\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}\}_j$ of the canonical factors as defined in Eqn. (7) using the empirical distribution.
- Threshold to one the factor entries $\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)$ satisfying $|\log \hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)| \leq \frac{\varepsilon}{2^{k+2}}$, and discard the factors that have all entries equal to one.
- Return the probability distribution $\tilde{P}(\mathbf{x}) \propto \prod_j \hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)$.

The thresholding step finds the factors that actually contribute to the distribution. The specific threshold is chosen to suit the proof of Theorem 15. If no thresholding were applied, the error in Eqn. (18) would be $\frac{|\mathcal{C}|}{2^k}\varepsilon$ instead of $J\varepsilon$, which is much larger in case the true distribution has a relatively small number of factors J.

Theorem 14 (Structure learning: computational complexity) *The running time*[15] *of* Factor-Graph-Structure-Learn *is in* $O\big(mkn^k bn^b(k+b) + kn^k bn^b v^{k+b} + kn^k 2^k v^k\big)$.

Thus the running time is exponential in the maximum factor scope size k and the maximum Markov blanket size b, polynomial in the number of variables n and the maximum domain size v, and linear in the number of training examples m.

The first two terms in Theorem 14 result from going through the data and computing the empirical conditional entropies. Since the algorithm considers all combinations of candidate factors and Markov blankets, we have an exponential dependence on the maximum scope size k and the

14. Note in the parameter learning setting we had b equal to the size the largest Markov blanket for an *actual* factor in the distribution. In contrast, now b corresponds to the size of the largest Markov blanket for any *candidate* factor up to size k.

15. The upper bound is based on a very naive implementation's running time.

maximum Markov blanket size b. The last term comes from computing the Markov blanket canonical factors. Importantly, unlike for currently-known (exact) ML approaches, the running time does not depend on the tractability of inference in the (unknown) factor graph from which the data was sampled, nor on the tractability of inference in the recovered factor graph.

Theorem 15 (Structure learning: sample complexity) *Let any* $\varepsilon, \delta > 0$ *be given. Let* Factor-Graph-Structure-Learn *be given (a) m training examples* $\{\mathbf{x}^{(i)}\}_{i=1}^{m}$ *drawn i.i.d. from a distribution P, (b) an upper bound k on the number of variables per factor in the factor graph for P, and (c) an upper bound b on the number of variables per Markov blanket for any set of variables up to size k in the factor graph for P. Let* $\tilde{P}$ *be the distribution returned by* Factor-Graph-Structure-Learn*. Then for*

$$D(P\|\tilde{P}) + D(\tilde{P}\|P) \leq J\varepsilon \tag{18}$$

to hold with probability $1-\delta$*, it suffices that the number of training examples m satisfies:*

$$m \geq (1 + \frac{\varepsilon\gamma^{k+b}}{2^{2k+3}})^2 \frac{v^{2k+2b}2^{8k+19}}{\gamma^{6k+6b}\min\{\varepsilon^2,\varepsilon^4\}} \log \frac{8kbn^{k+b}v^{k+b}}{\delta}. \tag{19}$$

Proof *(sketch). From Lemmas 12 and 13 we have that the conditioning set chosen by* Factor-Graph-Structure-Learn *results in a good approximation of the true canonical factor. At this point the structure is fixed, and we can use the sample complexity theorem for parameter learning to finish the proof.* ■

Theorem 15 shows that the sample complexity depends exponentially on the maximum factor size k and the maximum Markov blanket size b; and polynomially on $\frac{1}{\gamma}$ and $\frac{1}{\varepsilon}$. If we modify the analysis to consider the normalized KL-divergence, as in Section 3.5, we obtain a logarithmic dependence on the number of variables in the network.

To understand the implications of this theorem, consider the class of Gibbs distributions where every variable can participate in at most d factors and every factor can have at most k variables in its scope. Then we have that the Markov blanket size $b \leq dk^2$. Bayesian network probability distributions can also be represented using factor graphs.[16] If the number of parents per variable is bounded by numP and the number of children per variable is bounded by numC, then we have $k \leq \text{numP} + 1$, and that $b \leq (\text{numC}+1)(\text{numP}+1)^2$. Thus our factor graph structure learning algorithm allows us to efficiently learn distributions that can be represented by Bayesian networks with a bounded number of children and parents per variable. Note that our algorithm recovers a distribution which is close to the true generating distribution, but the distribution it returns is encoded as a factor graph, which may not be representable as a compact Bayesian network.

Theorem 15 considers the case where the generating distribution P factors according to a structure with factor scope sizes bounded by k and size of Markov blankets (of any subset of variables of size less than k) bounded by b. As we did in the case of parameter estimation, we can show that we have graceful degradation of performance for distributions that do not satisfy these assumptions.

16. Given a Bayesian network (BN), the following factor graph represents the same distribution: The factor graph has one variable node per variable in the BN. The factor graph has one factor for each variable in the BN. Each factor's scope is equal to the union of the corresponding variable itself and its parents. Each factor's entries are equal to the corresponding conditional probability table entries of the BN.

Theorem 16 (Structure learning: graceful degradation) *Let any $\varepsilon, \delta > 0$ be given. Let $\{\mathbf{x}^{(i)}\}_{i=1}^m$ be training examples drawn i.i.d. from a distribution Q. Let* MB *and* $\widehat{\text{MB}}$ *be the Markov blankets according to the distributions Q and found by* Factor-Graph-Structure-Learn *respectively. Let $\{f^*_{\mathbf{D}^*_j|\text{MB}(\mathbf{D}^*_j)}\}_j$ be the non-trivial Markov blanket canonical factors of Q (those factors with not all entries equal to one). Let J be the number of non-trivial Markov blanket canonical factors in Q with scope size smaller than k. Let $\tilde{P}$ be the probability distribution returned by* Factor-Graph-Parameter-Learn*. Then we have that for*

$$\begin{aligned} D(Q\|\tilde{P}) + D(\tilde{P}\|Q) \quad \leq \quad & (J + |S|)\varepsilon + \sum_{j:|\mathbf{D}^*_j|>k} 2 \max_{\mathbf{d}^*_j} \left| \log f^*_{\mathbf{D}^*_j}(\mathbf{d}_j) \right| \\ & + \sum_{\mathbf{C}^*_j \in \mathcal{C} \,:\, |\text{MB}(\mathbf{C}^*_j)|>b} 2 \max_{\mathbf{c}^*_j} \left| \log \frac{f^*_{\mathbf{C}^*_j|\text{MB}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)}{f^*_{\mathbf{C}^*_j|\widehat{MB}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)} \right| \end{aligned}$$

to hold with probability at least $1-\delta$, it suffices that the number of training examples m satisfies Eqn. (19) of Theorem 15. Here $S = \{j : \mathbf{C}^*_j \notin \{\mathbf{D}_l\}_l, |\text{MB}(\mathbf{C}^*_j)| > b\}$ is the set that indexes over the subsets of variables of size smaller than k over which there is no factor in the true distribution and for which the Markov blanket in the true distribution is larger than b; $\mathcal{C}$ is the set of candidate factor scopes $\mathcal{C} = \{\mathbf{C}^*_j : \mathbf{C}^*_j \subseteq \mathcal{X}, \mathbf{C}^*_j \neq \emptyset, |\mathbf{C}^*_j| \leq k\}$.

Theorem 16 shows that (similar to the parameter learning setting) each canonical factor that is not captured by our learned structure contributes at most a constant to our bound on the KL-divergence (namely $l2^{l+1}\log\frac{1}{\gamma}$ for a factor over l variables, see footnote 8 for details) to our bound on the KL-divergence. This bound on the error contribution can be large, so we discuss the actual error contribution in more detail. The reason a canonical factor is not captured could be two-fold. First, the scope of the factor could be too large. The paragraph after Theorem 7 discusses when the resulting error is expected to be small. Second, the Markov blanket of the factor could be too large. As shown in Lemma 13, a good approximate Markov blanket is sufficient to get a good approximation. So we can expect these error contributions to be small if the true distribution is mostly determined by interactions between small sets of variables.

Recall that the structure learning algorithm correctly clips all estimates of trivial canonical factors to the trivial all-ones factor, when the structural assumptions are satisfied. I.e., trivial factors are correctly estimated as trivial if their Markov blanket is of size smaller than b. The additional term $|S|\varepsilon$ corresponds to estimation error on the factors that are trivial in the true distribution but that have a Markov blanket of size larger than b, and are thus not correctly estimated and clipped to trivial all-ones factors.

5. Related Work

Tables 2 and 3 summarize the prior work on Markov network and Bayesian network learning that comes with formal guarantees. In the following two sections we discuss the prior work on Markov network (factor graph) learning and Bayesian network learning in more detail. We also discuss algorithms that do not have formal guarantees.

Target distribution	True distribution	Structure/Parameter	Samples	Time	Graceful degradation	Reference
ML tree	any	structure	poly	poly	yes	[1]
ML bounded tree-width	any	structure	poly	NP-hard	yes	[2]
Bounded tree-width	same	structure	poly	poly	no	[3]
Factor graph	same	parameter	infinite	convex	no	[4], [5]
Factor graph	same	parameter	poly	poly	yes	[6]
Factor graph	same	structure	poly	poly	yes	[6]

Table 2: Overview of prior work on learning Markov networks that has formal guarantees. More details are given in Section 5.1. The references in the table are: [1]: Chow and Liu (1968); [2] Srebro (2001); [3]: Narasimhan and Bilmes (2004); [4]: Besag (1974b); [5]: Gidas (1988); [6]: this paper. "Convex" refers to the time of solving a convex optimization problem.

5.1 Markov Networks

We split the discussion into two parts: parameter learning and structure learning.

5.1.1 PARAMETER LEARNING

The most natural algorithm for parameter estimation in undirected graphical models is maximum likelihood (ML) estimation (possibly with some regularization). Unfortunately, evaluating the likelihood of such a model requires evaluating the partition function. All currently known ML algorithms for undirected graphical models require evaluating the partition function. Therefore, they are computationally tractable only for networks in which inference is computationally tractable. In contrast, our closed form solution can be efficiently computed from the data, even for Markov networks where inference is intractable. Note that our estimator does not return the ML solution, so that our result does not contradict the "hardness" of ML estimation. However, it does provide a low KL-divergence estimate of the probability distribution, with high probability, from a "small" number of training examples, assuming the true distribution approximately factors according to the given structure.

Criteria different from ML have been proposed for learning Markov networks. The most prominent one is *pseudo-likelihood* (Besag, 1974b), and its extension, generalized pseudo-likelihood (Huang and Ogata, 2002). The pseudo-likelihood criterion gives rise to a tractable convex optimization problem. Pseudo-likelihood estimation is consistent, that is, in the infinite sample limit it returns the true distribution, when the assumed structure is correct. (See, for example, Gidas, 1988, .) However, in the finite sample case the pseudo-likelihood estimate is often significantly worse than the maximum likelihood estimate. More information on the statistical efficiency of the pseudo-likelihood estimate can be found in, for example, Besag (1974a); Geyer and Thompson (1992); Guyon and Künsch (1992). In contrast to our results, no finite sample bounds have been provided for pseudo-likelihood estimation. Moreover, the theoretical analyses (e.g., Geman and Graffigne, 1986; Comets, 1992; Guyon and Künsch, 1992) only apply when the generating model is in the true target class.

5.1.2 STRUCTURE LEARNING

Structure learning for Markov networks is notoriously difficult, as it is generally based on using ML estimation of the parameters (with smoothing), often combined with a penalty term for structure complexity. As evaluating the likelihood is only possible for the class of Markov networks in which

Target distribution	True distribution	Structure/Parameter	Samples	Time	Graceful degradation	Reference
ML polytree	any	structure	poly	NP-hard	yes	[1], [2]
ML BN	any	structure	poly	NP-hard	yes	[1], [3]
BN	same	structure	infinite	poly	yes	[4], [5]
Factor graph	BN (same)	structure	poly	poly	yes	[6]

Table 3: Overview of prior work on learning Bayesian networks that has formal guarantees. More details are given in Section 5.2. The references in th table are: [1]: Höffgen (1993); [2]: Dasgupta (1999); [3]: Chickering et al. (2003); [4]: Spirtes et al. (2000); [5]: Cheng et al. (2002); [6]: this paper.

inference is tractable, there have been two main research tracks for ML structure learning. The first, starting with the work of Della Pietra et al. (1997), uses local-search heuristics to add factors into the network (see also McCallum, 2003). The second searches for a structure within a restricted class of models in which inference is tractable, more specifically, bounded tree-width Markov networks. Indeed, ML learning of the class of tree Markov networks—networks of tree-width 1—can be performed very efficiently (Chow and Liu, 1968). Unfortunately, Srebro (2001) proves that for any tree-width k greater than 1, even finding the ML tree-width-k network is NP-hard. Karger and Srebro (2001) provide an approximation algorithm but the approximation factor is a very large multiplicative factor of the log-likelihood. In particular, for tree-width k, they find a Markov network (of tree-width k) with log-likelihood at least $1/(8^k k!(k+1)!)$ times the optimal log-likelihood. Several heuristic algorithms to learn models with small tree-width have been proposed (Malvestuto, 1991; Bach and Jordan, 2002; Deshpande et al., 2001), but (not surprisingly, given the NP-hardness of the problem) they do not come with any performance guarantees.

Recently, Narasimhan and Bilmes (2004) provided a polynomial time algorithm with a polynomial sample complexity guarantee for the class of Markov networks of bounded tree-width. Their algorithm computes approximate conditional independence information followed by dynamic programming to recover the bounded tree-width structure. The parameters for the recovered bounded tree-width model are estimated by standard ML methods. Our algorithm applies to a different family of distributions: factor graphs of bounded connectivity (including graphs in which inference is intractable). Factor graphs with small connectivity can have large tree-width (e.g., grids) and factor graphs with small tree-width can have large connectivity (e.g., star graphs). Thus, the range of applicability is incomparable. Narasimhan and Bilmes (2004) did not provide any graceful degradation guarantees when the generating distribution is not a member of the target class. However, future research might extend their algorithm to this setting.

Pseudo-likelihood has been extended to a criterion for model selection: the resulting criterion is statistically consistent (Ji and Seymour, 1996). In particular they show that the probability of selecting an incorrect model goes to zero as the number of training examples goes to infinity. They also provide a bound on how fast this probability goes to zero. Importantly, Ji and Seymour (1996) only provide a model selection criterion. They do *not* provide an algorithm to efficiently find the best pseudo-likelihood model (according to their evaluation criterion) over the super-exponentially large set of candidate models from which we want to select in the structure learning problem.

5.2 Bayesian Networks

Again, we split the discussion into two parts: parameter learning and structure learning.

5.2.1 PARAMETER LEARNING

ML parameter learning in Bayesian networks (possibly with smoothing) only requires computing the empirical conditional probabilities of each variable given its parent instantiations. Thus there is no computational challenge.

Dasgupta (1997), following earlier work by Friedman and Yakhini (1996), analyzes the sample complexity of learning Bayesian networks, showing that it is polynomial in the maximal number of different instantiations per family. His sample complexity result has logarithmic dependence on the number of variables in the network, when using the KL-divergence normalized by the number of variables in the network. In this paper, we strengthen his result, showing an $O(1)$ dependence of the number of training examples on the number of variables in the network. So for bounded fan-in Bayesian networks, the sample complexity is independent of the number of variables in the network.

5.2.2 STRUCTURE LEARNING

Results analyzing the complexity of structure learning of Bayesian networks fall largely into two classes. The first class of results assumes that the generating distribution is DAG-perfect with respect to some DAG G with at most k parents for each node. (That is, P and G satisfy precisely the same independence assertions.) In this case, algorithms based on various independence tests (Spirtes et al., 2000; Cheng et al., 2002) can identify the correct network structure in the infinite sample limit (i.e., when given an infinite number of training examples), using a polynomial number of independence tests. The infinite sample limit setting is critical in their analysis since it allows for exact independence tests. Neither Spirtes et al. (2000) nor Cheng et al. (2002) provide guarantees for the case of a finite number of training examples, but future research might extend their results to this setting. Chickering and Meek (2002) relax the assumption that the distribution be DAG-perfect; they show that, under a certain assumption, a simple greedy algorithm will, in the infinite sample limit, identify a network structure which is a minimal I-map of the distribution. They provide no polynomial time guarantees, but future work might provide such guarantees for models with bounded connectedness (such as the ones our algorithm considers).

The second class of results relates to the problem of finding a network structure whose score is high, for a given set of training examples and some appropriate scoring function. Although finding the highest-scoring tree-structured network can be done in polynomial time (Chow and Liu, 1968), Chickering (1996) shows that the problem of finding the highest scoring Bayesian network where each variable has at most k parents is NP-hard, for any $k \geq 2$. (See Chickering et al., 2003, for details.) Even finding the maximum likelihood structure among the class of polytrees (Dasgupta, 1999) or paths (Meek, 2001) is NP-hard. These results do not address the question of the number of training examples for which the highest scoring network is guaranteed to be close to the true generating distribution.

Höffgen (1993) analyzes the problem of PAC-learning the structure of Bayesian networks with bounded fan-in, showing that the sample complexity depends only logarithmically on the number of variables in the network (when considering KL-divergence normalized by the number of variables in the network). Höffgen does not provide an efficient learning algorithm (and to date, no efficient learning algorithm is known), stating only that if the optimal network for a given data set can be found (e.g., by exhaustive enumeration), it will be close to optimal with high probability.

In contrast, we provide a polynomial-time learning algorithm with similar performance guarantees for Bayesian networks with bounded fan-in and bounded fan-out. However, we note that our algorithm does not construct a Bayesian network representation, but rather a factor graph; this factor graph may not be compactly representable as a Bayesian network, but it is guaranteed to encode a distribution which is close to the generating distribution, with high probability.

6. Discussion

We have presented the first polynomial-time and polynomial sample-complexity algorithms for both parameter estimation and structure learning in factor graphs of bounded degree. When the generating distribution is within this class of networks, our algorithms are guaranteed to return a distribution close to it, using a polynomial number of training examples. When the generating distribution is not in this class, our algorithm degrades gracefully. Thus our algorithms and analysis are the first to establish the efficient learnability of an important class of distributions.

While of significant theoretical interest, our algorithms, as described, are probably impractical. From a statistical perspective, our algorithm is based on the canonical parameterization, which is evaluated relative to a canonical assignment $\bar{\mathbf{x}}$. Many of the empirical estimates that we compute in the algorithm use only a subset of the training examples that are (in some ways) consistent with $\bar{\mathbf{x}}$. As a consequence, we make very inefficient use of data, in that many training examples may never be used. In regimes where data is not abundant, this limitation may be quite significant in practice. From a computational perspective, our algorithm uses exhaustive enumeration over all possible factors up to some size k, and over all possible Markov blankets up to size b. When we fix k and b to be constant, the complexity is polynomial. But in practice, the set of all subsets of size k or b is often much too large to search exhaustively.

Nevertheless, aside from proving the efficient learnability of an important class of probability distributions, the algorithms we propose might provide insight into the development of new learning algorithms that do work well in practice. In particular, we might be able to address the statistical limitation by putting together canonical factor estimates from multiple canonical assignments $\bar{\mathbf{x}}$. We might be able to address the computational limitation using a more clever (perhaps heuristic) algorithm for searching over subsets. Given the limitations of existing parameter and structure learning algorithms for undirected models, we believe that the techniques suggested by our theoretical analysis are well worth exploring.

Acknowledgments

This work was supported by the Department of the Interior/DARPA under contract number NBCHD030010, and by DARPA's EPCA program, under subcontract to SRI International.

Appendix A. Proofs for Section 2.2

In this section we give formal proofs of all theorems, propositions and lemmas appearing in Section 2.2.

A.1 Proof of Theorem 3

Proof [Theorem 3] The proof consists of two parts:

1. If we let $\{\mathbf{C}_j^*\}_{j=1}^{J^*} = 2^{\mathcal{X}} - \emptyset$, then $P(\mathbf{x}) = P(\overline{\mathbf{x}}) \prod_{j=1}^{J^*} f^*_{\mathbf{C}_j^*}(\mathbf{c}_j^*)$.

2. If P is a positive Gibbs distribution with factor scopes $\{\mathbf{C}_j\}_{j=1}^{J}$, then the canonical factors $f^*_{\mathbf{D}}$ are trivial all-ones factors, whenever $\mathbf{D} \notin \cup_{j=1}^{J} 2^{\mathbf{C}_j}$.

The first part states that the canonical parameterization gives the correct distribution assuming we use a canonical factor for each subset of variables. It is easily verified by counting how often the probabilities $P(\sigma_{\mathbf{U}}[\mathbf{d}])$ contribute for each $\mathbf{U} \subseteq \mathbf{D} \subseteq \mathcal{X}$, and is a standard part of most Hammersley-Clifford theorem proofs. The second part states that we can ignore canonical factors over subsets of variables that do not appear together in one of the factor scopes $\{\mathbf{C}_j\}_{j=1}^{J}$. We now prove the second part. We have

$$\begin{aligned}
\log f^*_{\mathbf{D}}(\mathbf{d}) &= \sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{U}|} \log P(\sigma_{\mathbf{U}}[\mathbf{d}]) \\
&= \sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{U}|} \left(\sum_{j=1}^{J} \log f_{\mathbf{C}_j}(\mathbf{C}_j[\sigma_{\mathbf{U}}[\mathbf{d}]]) + \log \frac{1}{Z} \right) \\
&= \sum_{j=1}^{J} \sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{U}|} \log f_{\mathbf{C}_j}(\mathbf{C}_j[\sigma_{\mathbf{U}}[\mathbf{d}]]). \qquad (20)
\end{aligned}$$

To obtain the last equality, we used the fact that there is an equal number of terms $(\log \frac{1}{Z})$ and $(-\log \frac{1}{Z})$. Now consider the contribution of one factor $f_{\mathbf{C}_j}$ in the above expression. By assumption we have that $\mathbf{D} \notin \cup_{j=1}^{J} 2^{\mathbf{C}_j}$ and thus $\mathbf{D} - \mathbf{C}_j \neq \emptyset$. Now let Y be any element of $\mathbf{D} - \mathbf{C}_j$. Then we have that

$$\begin{aligned}
\sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{U}|} \log f_{\mathbf{C}_j}(\mathbf{C}_j[\sigma_{\mathbf{U}}[\mathbf{d}]]) = \sum_{\mathbf{U} \subseteq \mathbf{D}-Y} & (-1)^{|\mathbf{D}-\mathbf{U}|} \log f_{\mathbf{C}_j}(\mathbf{C}_j[\sigma_{\mathbf{U}}[\mathbf{d}]]) \\
& + (-1)^{|\mathbf{D}-\mathbf{U}-Y|} \log f_{\mathbf{C}_j}(\mathbf{C}_j[\sigma_{\mathbf{U}\cup Y}[\mathbf{d}]]).
\end{aligned}$$

Now since $Y \notin \mathbf{C}_j$, we have $\mathbf{C}_j[\sigma_{\mathbf{U}}[\mathbf{d}]] = \mathbf{C}_j[\sigma_{\mathbf{U}\cup Y}[\mathbf{d}]]$. And thus we get

$$\sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{U}|} \log f_{\mathbf{C}_j}(\mathbf{C}_j[\sigma_{\mathbf{U}}[\mathbf{d}]]) = 0. \qquad (21)$$

And thus combining Eqn. (21) with Eqn. (20) establishes the second part of the proof. ■

Appendix B. Proofs for Section 3.2

In this section we give formal proofs of all theorems, propositions and lemmas appearing in Section 3.2.

Proof [Proposition 4] In Eqn. (2) the number of terms with a positive sign and a negative sign are both equal to $2^{|\mathbf{D}|-1}$. So we can divide the argument of the log in each term by the same constant $P(\mathcal{X}-\mathbf{D}=(\mathcal{X}-\mathbf{D})[\overline{\mathbf{x}}])$ without changing the factor. The resulting expression is exactly the expression defining $f^*_{\mathbf{D}|\mathcal{X}-\mathbf{D}}$ in Eqn. (7), thus proving the first equality in Eqn. (8). The second equality in Eqn. (8) follows directly from Eqn. (1) and the definition of the factors as functions of probabilities in Eqn. (7). Eqn. (9) and (10) follow directly from Eqn. (8) and Theorem 3. ■

Appendix C. Proofs for Section 3.3

In this section we give formal proofs of all theorems, propositions and lemmas appearing in Section 3.3.

Proof [Theorem 5] The algorithm consists of two parts:

- Collecting the empirical probabilities for each of the factors, jointly with the default instantiation of their Markov blanket. This can be done in three steps. [Below, recall that the maximum factor scope size is k, so there are at most $2^k J$ different canonical factors. Each variable can take on at most v different values.]
  - For all instantiations of all factors initialize the occurrence count to zero. This can be done in $O(2^k J v^k)$.
  - When going through the m data points, we need to add to the counts of the observed instantiation whenever the Markov blanket is in the default instantiation. Reading a specific instantiation of a specific factor and its Markov blanket takes $O(k+b)$ to read every variable. Thus collecting the data counts from which the empirical probabilities will be computed takes $O(m2^k J(k+b))$.
  - Renormalizing all of the entries to get the empirical conditional probabilities takes time $O(2^k J v^k)$.
- Computing the factor entries from the empirical probabilities. To compute one factor entry $f^*_{\mathbf{C}^*_j}(\mathbf{c}^*_j)$, we have to add (and subtract) $2^{|\mathbf{C}^*_j|}$ empirical log-probabilities. (Note this is the case independent of the cardinality of the variables in the factor, as seen from Eqn. (7).) This gives us $O(2^{|\mathbf{C}^*_j|})$ operations per factor entry, and thus $O(J2^{2k}v^k)$ total for computing the canonical factor entries from the empirical probabilities.

Adding up the upper bounds on the running times of each step proves the theorem. ■

Appendix D. Proofs for Section 3.4

In this section we give formal proofs of all theorems, propositions and lemmas appearing in Section 3.4.

D.1 Proof of Theorem 6

The proof of the theorem is based on a series of lemmas.

The following lemma shows that the log of the empirical average is an accurate estimate of the log of the population average, if the population average is bounded away from zero.

Lemma 17 *Let any $\varepsilon > 0, \delta > 0, \lambda \in (0,1)$ be given. Let $\{X_i\}_{i=1}^m$ be i.i.d. Bernoulli(ϕ) random variables, where $\lambda \leq \phi \leq 1-\lambda$. Let $\hat{\phi} = \frac{1}{m}\sum_{i=1}^m X_i$. Then for*

$$|\log\phi - \log\hat{\phi}| \leq \varepsilon$$

to hold w.p. $1-\delta$, it suffices that

$$m \geq \frac{(1+\varepsilon)^2}{2\lambda^2\varepsilon^2}\log\frac{2}{\delta}.$$

Proof *From the Hoeffding inequality we have that for*

$$|\phi - \hat{\phi}| \leq \varepsilon'$$

to hold w.p. $1-\delta$ it suffices that

$$m \geq \frac{1}{2\varepsilon'^2}\log\frac{2}{\delta}. \tag{22}$$

Since the function $f(x) = \log x$ is Lipschitz with Lipschitz-constant smaller than $\frac{1}{\lambda-\varepsilon'}$ over the interval $[\lambda-\varepsilon', 1]$, we have that for

$$|\log\phi - \log\hat{\phi}| \leq \frac{\varepsilon'}{\lambda-\varepsilon'}$$

to hold w.p. $1-\delta$, it suffices that m satisfies Eqn. (22). Now for $\frac{\varepsilon'}{\lambda-\varepsilon'} \leq \varepsilon$ to hold, it suffices that $\varepsilon' \leq \frac{\varepsilon\lambda}{1+\varepsilon}$. Using this choice of ε' in Eqn. (22) gives the condition for m as stated in the lemma. ■

The following lemma shows that for distributions that are bounded away from zero, conditional probabilities can be accurately estimated from a small number of samples.

Lemma 18 *Let any $\varepsilon, \delta > 0$ be given. Let $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^m$ be i.i.d. samples from a distribution P over $\mathbf{X}, \mathbf{Y}$. Let $\hat{P}$ be the empirical distribution. Let $\lambda = \min_{\mathbf{x},\mathbf{y}} P(\mathbf{X}=\mathbf{x}, \mathbf{Y}=\mathbf{y})$. Then for*

$$|\log P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}) - \log\hat{P}(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y})| \leq \varepsilon$$

to hold for all $\mathbf{x}, \mathbf{y}$ with probability $1-\delta$, it suffices that

$$m \geq \frac{(1+\frac{\varepsilon}{2})^2}{2\lambda^2(\frac{\varepsilon}{2})^2}\log\frac{4|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|}{\delta}.$$

Proof *We have (using the definition of conditional probability and the triangle inequality)*

$$\begin{aligned}
&\left|\log P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}) - \log\hat{P}(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y})\right| \\
&\quad= \left|\left(\log P(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y}) - \log P(\mathbf{Y}=\mathbf{y})\right)\right. \\
&\qquad\left.-\left(\log\hat{P}(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y}) - \log\hat{P}(\mathbf{Y}=\mathbf{y})\right)\right| \\
&\quad\leq \left|(\log P(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y}) - \log\hat{P}(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y})\right| \\
&\qquad+\left|(\log P(\mathbf{Y}=\mathbf{y}) - \log\hat{P}(\mathbf{Y}=\mathbf{y})\right|.
\end{aligned}$$

Now using Lemma 17 (note that $\lambda = \min_{\mathbf{x},\mathbf{y}} P(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y}) \leq \min_{\mathbf{y}} P(\mathbf{Y}=\mathbf{y})$*) and the Union bound to bound both terms by* $\varepsilon/2$*, we get that for*

$$\left|\log P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}) - \log \hat{P}(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y})\right| \leq \varepsilon \tag{23}$$

to hold with probability $1-2\delta'$*, it is sufficient that*

$$m \geq \frac{(1+\varepsilon/2)^2}{2\lambda^2(\varepsilon/2)^2}\log\frac{2}{\delta'}. \tag{24}$$

Using the Union bound, we get that for Eqn. (23) to hold with probability $1-2|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|\delta'$ *for all* $\mathbf{x}\in\mathrm{val}(\mathbf{X}),\mathbf{y}\in\mathrm{val}(\mathbf{Y})$ *it suffices that m satisfies Eqn. (24). Choosing* $\delta = 2|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|\delta'$ *gives the statement of the lemma.* ■

Our algorithm uses probability estimates to compute canonical factors. The following lemma shows that accurate probabilities are sufficient to obtain accurate canonical factors.

Lemma 19 *Let any* $\varepsilon > 0$ *be given. Let any* $\mathbf{D},\mathbf{Y},\mathbf{W} \subseteq \mathcal{X}, \mathbf{D}\cap\mathbf{Y}=\emptyset, \mathbf{D}\cap\mathbf{W}=\emptyset$ *be given. Then for all* $\mathbf{d}\in\mathrm{val}(\mathbf{D})$ *for*

$$|\log f^*_{\mathbf{D}|\mathbf{Y}}(\mathbf{d}) - \log \hat{f}^*_{\mathbf{D}|\mathbf{W}}(\mathbf{d})| \leq \varepsilon$$

to hold, it suffices that for all instantiations $\mathbf{d}\in\mathrm{val}(\mathbf{D})$ *we have that*

$$|\log P(\mathbf{d}|\bar{\mathbf{y}}) - \log \hat{P}(\mathbf{d}|\bar{\mathbf{w}})| \leq \tfrac{\varepsilon}{2^{|\mathbf{D}|}}. \tag{25}$$

Proof

$$\begin{aligned}
\left|\log f^*_{\mathbf{D}|\mathbf{Y}}(\mathbf{d}) - \log \hat{f}^*_{\mathbf{D}|\mathbf{W}}(\mathbf{d})\right| &= \Big|\sum_{\mathbf{Z}\subseteq\mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{Z}|}\log P(\sigma_{\mathbf{Z}:\mathbf{D}}[\mathbf{d}]|\mathbf{Y}=\bar{\mathbf{y}}) \\
&\quad - \sum_{\mathbf{Z}\subseteq\mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{Z}|}\log \hat{P}(\sigma_{\mathbf{Z}:\mathbf{D}}[\mathbf{d}]|\mathbf{W}=\bar{\mathbf{w}})\Big| \\
&\leq \sum_{\mathbf{Z}\subseteq\mathbf{D}} \Big|\log P(\sigma_{\mathbf{Z}:\mathbf{D}}[\mathbf{d}]|\mathbf{Y}=\bar{\mathbf{y}}) \\
&\quad - \log \hat{P}(\sigma_{\mathbf{Z}:\mathbf{D}}[\mathbf{d}]|\mathbf{W}=\bar{\mathbf{w}})\Big| \\
&\leq \sum_{\mathbf{Z}\subseteq\mathbf{D}} \frac{\varepsilon}{2^{|\mathbf{D}|}} \\
&= \varepsilon,
\end{aligned}$$

where, in order, we used the definitions of f^* *and* $\hat{f}^*$*; triangle inequality; Eqn. (25); number of subsets of* $\mathbf{D}$ *equals* $2^{|\mathbf{D}|}$*.* ■

The next step is to show that, if we obtain good estimates of the factors, the distributions they induce should be close as well. The following lemma shows that distributions with approximately the same factors are close to each other, by proving a bound on $D(P\|\hat{P}) + D(\hat{P}\|P)$, and thus (since $D(\cdot\|\cdot)\geq 0$) a bound on $D(P\|\hat{P})$.

Lemma 20 *Let* $P(\mathbf{x}) = \frac{1}{Z}\prod_{j=1}^{J} f_j(\mathbf{c}_j)$ *and* $\hat{P}(\mathbf{x}) = \frac{1}{\hat{Z}}\prod_{j=1}^{J} \hat{f}_j(\mathbf{c}_j)$. *Let* $\varepsilon = \max_{j\in\{1,\cdots,J\},\mathbf{c}_j} |\log f_j(\mathbf{c}_j) - \log \hat{f}_j(\mathbf{c}_j)|$. *Then we have that*

$$D(P\|\hat{P}) + D(\hat{P}\|P) \leq 2J\varepsilon.$$

Proof

$$\begin{aligned} D(P\|\hat{P}) + D(\hat{P}\|P) &= \mathrm{E}_{\mathbf{X}\sim P}\left(\log P(\mathbf{X}) - \log \hat{P}(\mathbf{X})\right) + \mathrm{E}_{\mathbf{X}\sim \hat{P}}\left(\log \hat{P}(\mathbf{X}) - \log P(\mathbf{X})\right) \\ &= \mathrm{E}_{\mathbf{X}\sim P}\sum_{j=1}^{J}\left(\log f_j(\mathbf{C}_j^*) - \log \hat{f}_j(\mathbf{C}_j^*)\right) - \log\frac{Z}{\hat{Z}} \\ &\quad + \mathrm{E}_{\mathbf{X}\sim \hat{P}}\sum_{j=1}^{J}\left(\log \hat{f}_j(\mathbf{C}_j^*) - \log f_j(\mathbf{C}_j^*)\right) - \log\frac{\hat{Z}}{Z} \\ &\leq 2J\varepsilon, \end{aligned}$$

where we used in order: the definition of KL-divergence; the definition of P, $\hat{P}$; $\log\frac{Z}{\hat{Z}} + \log\frac{\hat{Z}}{Z} = 0$, *and the fact that each term in the expectation is bounded in absolute value by* ε. ■

Note that (by using the sum of the KL-divergences) we have that the terms that involve the partition functions Z and $\hat{Z}$ cancel. This enables us to prove an error bound without bounding the difference $|\log Z - \log \hat{Z}|$ as a function of the errors in the factors.

We now show how the previous lemmas can be used to prove the parameter learning sample complexity result stated in Theorem 6.

Proof [Theorem 6] First note that since the scopes of the canonical factors used by the algorithm are subsets of the given scopes $\{\mathbf{C}_j\}_{j=1}^{J}$, we have that

$$\max_j |\mathbf{C}_j^* \cup \mathrm{MB}(\mathbf{C}_j^*)| \quad \leq \quad b + k.$$

Let $\hat{P}$ be the empirical distribution as given by the samples $\{\mathbf{x}^{(i)}\}_{i=1}^{m}$. Let $\mathbf{M}_j^* = \mathrm{MB}(\mathbf{C}_j^*)$. Then from Lemma 18 we have that for any $j \in \{1,\ldots,J^*\}$ for

$$\left|\log P(\mathbf{C}_j^* = \mathbf{c}_j^* | \mathbf{M}_j^* = \mathbf{m}_j^*) - \log \hat{P}(\mathbf{C}_j^* = \mathbf{c}_j^* | \mathbf{M}_j^* = \mathbf{m}_j^*)\right| \leq \varepsilon' \tag{26}$$

to hold for all instantiations $\mathbf{c}_j^*, \mathbf{m}_j^*$ with probability $1-\delta'$, it suffices that

$$m \geq \frac{(1+\frac{\varepsilon'}{2})^2}{2\gamma^{2k+2b}(\frac{\varepsilon'}{2})^2}\log\frac{4v^{k+b}}{\delta'}. \tag{27}$$

Using Lemma 19 we obtain that for all instantiations $\mathbf{c}_j^*$ we have that Eqn. (26) implies

$$|\log f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*) - \log \hat{f}^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)| \leq 2^k\varepsilon'. \tag{28}$$

Using the union bound, we get that for Eqn. (28) to hold for all $j \in J^*$ with probability $1 - J^*\delta'$, it suffices that m satisfies Eqn. (27). When Eqn. (28) holds for all $j \in J^*$, Lemma 20 and Proposition 4 give us that

$$D(P\|\tilde{P}) + D(\tilde{P}\|P) \leq 2J^*2^k\varepsilon'. \tag{29}$$

We have that $J^* \leq 2^k J$. Choosing $\varepsilon' = \frac{\varepsilon}{2^{2k+1}}$ and $\delta' = \frac{\delta}{2^k J}$ and substituting these choices into Eqn. (27) and Eqn. (29) gives the theorem. ■

D.2 Proof of Theorem 7

Proof [Theorem 7] From Proposition 4 we have that

$$Q(\mathbf{x}) = \frac{1}{Z} \prod_{j=1}^{\bar{J}} f^*_{\mathbf{D}^*_j|\mathrm{MB}(\mathbf{D}^*_j)}(\mathbf{d}^*_j).$$

We can rewrite this product as follows:

$$\begin{aligned} Q(\mathbf{x}) \;=\; \frac{1}{Z} & \prod_{\substack{j:\ \mathbf{D}^*_j \in \{\mathbf{C}^*_k\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}^*_j) = \widehat{MB}(\mathbf{D}^*_j)}} f^*_{\mathbf{D}^*_j|\mathrm{MB}(\mathbf{D}^*_j)}(\mathbf{d}^*_j) \\ & \prod_{\substack{j:\ \mathbf{D}^*_j \in \{\mathbf{C}^*_k\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}^*_j) \neq \widehat{MB}(\mathbf{D}^*_j)}} f^*_{\mathbf{D}^*_j|\mathrm{MB}(\mathbf{D}^*_j)}(\mathbf{d}^*_j) \\ & \prod_{j:\mathbf{D}^*_j \notin \{\mathbf{C}^*_k\}_{k=1}^{J^*}} f^*_{\mathbf{D}^*_j|\mathrm{MB}(\mathbf{D}^*_j)}(\mathbf{d}^*_j). \end{aligned} \tag{30}$$

We also have

$$\tilde{P}(\mathbf{x}) = \frac{1}{\tilde{Z}} \prod_{j=1}^{J^*} \hat{f}^*_{\mathbf{C}^*_j|\widehat{\mathrm{MB}}(\mathbf{C}^*_j)}(\mathbf{c}^*_j). \tag{31}$$

We can rewrite this product as follows:

$$\begin{aligned} \tilde{P}(\mathbf{x}) \;=\; \frac{1}{\tilde{Z}} & \prod_{\substack{j:\ \mathbf{D}^*_j \in \{\mathbf{C}^*_k\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}^*_j) = \widehat{MB}(\mathbf{D}^*_j)}} \hat{f}^*_{\mathbf{D}^*_j|\mathrm{MB}(\mathbf{D}^*_j)}(\mathbf{d}^*_j) \\ & \prod_{\substack{j:\ \mathbf{D}^*_j \in \{\mathbf{C}^*_k\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}^*_j) \neq \widehat{MB}(\mathbf{D}^*_j)}} \hat{f}^*_{\mathbf{D}^*_j|\widehat{\mathrm{MB}}(\mathbf{D}^*_j)}(\mathbf{d}^*_j) \\ & \prod_{\substack{j:\ \mathbf{C}^*_j \notin \{\mathbf{D}^*_k\}_{k=1}^{\bar{J}} \\ \mathrm{MB}(\mathbf{C}^*_j) = \widehat{MB}(\mathbf{C}^*_j)}} \hat{f}^*_{\mathbf{C}^*_j|\widehat{\mathrm{MB}}(\mathbf{C}^*_j)}(\mathbf{c}^*_j) \\ & \prod_{\substack{j:\ \mathbf{C}^*_j \notin \{\mathbf{D}^*_k\}_{k=1}^{\bar{J}} \\ \mathrm{MB}(\mathbf{C}^*_j) \neq \widehat{MB}(\mathbf{C}^*_j)}} \hat{f}^*_{\mathbf{C}^*_j|\widehat{\mathrm{MB}}(\mathbf{C}^*_j)}(\mathbf{c}^*_j). \end{aligned} \tag{32}$$

We have (adding and subtracting same term):

$$\log \frac{f^*_{\mathbf{D}^*_j|\mathrm{MB}(\mathbf{D}^*_j)}(\mathbf{d}^*_j)}{\hat{f}^*_{\mathbf{D}^*_j|\widehat{\mathrm{MB}}(\mathbf{D}^*_j)}(\mathbf{d}^*_j)} = \log \frac{f^*_{\mathbf{D}^*_j|\mathrm{MB}(\mathbf{D}^*_j)}(\mathbf{d}^*_j)}{f^*_{\mathbf{D}^*_j|\widehat{\mathrm{MB}}(\mathbf{D}^*_j)}(\mathbf{d}^*_j)} + \log \frac{f^*_{\mathbf{D}^*_j|\widehat{\mathrm{MB}}(\mathbf{D}^*_j)}(\mathbf{d}^*_j)}{\hat{f}^*_{\mathbf{D}^*_j|\widehat{\mathrm{MB}}(\mathbf{D}^*_j)}(\mathbf{d}^*_j)}. \tag{33}$$

We also have for $j : \mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_k$ that $\log f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{C}_j^*) = 0$. Thus we have (adding zero and adding and subtracting same term):

$$\log \hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*) = \log \frac{\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)} + \log \frac{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}. \tag{34}$$

Using Eqn. (30), Eqn. (32), Eqn. (33) and Eqn. (34) we get that $D(Q\|\tilde{P}) + D(\tilde{P}\|Q) =$

$$E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{D}_j^* \in \{\mathbf{C}_k^*\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}_j^*) = \widehat{MB}(\mathbf{D}_j^*)}} \log \frac{f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{\hat{f}^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) - E_{\mathbf{X}\sim \tilde{P}}\Big(\sum_{\substack{j:\ \mathbf{D}_j^* \in \{\mathbf{C}_k^*\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}_j^*) = \widehat{MB}(\mathbf{D}_j^*)}} \log \frac{f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{\hat{f}^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) \tag{35}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{D}_j^* \in \{\mathbf{C}_k^*\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}_j^*) \neq \widehat{MB}(\mathbf{D}_j^*)}} \log \frac{f^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{\hat{f}^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) - E_{\mathbf{X}\sim \tilde{P}}\Big(\sum_{\substack{j:\ \mathbf{D}_j^* \in \{\mathbf{C}_k^*\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}_j^*) \neq \widehat{MB}(\mathbf{D}_j^*)}} \log \frac{f^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{\hat{f}^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) \tag{36}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{D}_j^* \in \{\mathbf{C}_k^*\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}_j^*) \neq \widehat{MB}(\mathbf{D}_j^*)}} \log \frac{f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{f^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) - E_{\mathbf{X}\sim \tilde{P}}\Big(\sum_{\substack{j:\ \mathbf{D}_j^* \in \{\mathbf{C}_k^*\}_{k=1}^{J^*} \\ \mathrm{MB}(\mathbf{D}_j^*) \neq \widehat{MB}(\mathbf{D}_j^*)}} \log \frac{f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{f^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) \tag{37}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{j:\mathbf{D}_j \notin \{\mathbf{C}_k^*\}_{k=1}^{J^*}} \log f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)\Big) - E_{\mathbf{X}\sim \tilde{P}}\Big(\sum_{j:\mathbf{D}_j \notin \{\mathbf{C}_k^*\}_{k=1}^{J^*}} \log f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)\Big) \tag{38}$$

$$-E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_{k=1}^{\bar{J}} \\ \mathrm{MB}(\mathbf{C}_j^*) = \widehat{MB}(\mathbf{C}_j^*)}} \log \hat{f}^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)\Big) + E_{\mathbf{X}\sim \tilde{P}}\Big(\sum_{\substack{j:\ \mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_{k=1}^{\bar{J}} \\ \mathrm{MB}(\mathbf{C}_j^*) = \widehat{MB}(\mathbf{C}_j^*)}} \log \hat{f}^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)\Big) \tag{39}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_{k=1}^{\bar{J}} \\ \mathrm{MB}(\mathbf{C}_j^*) \neq \widehat{MB}(\mathbf{C}_j^*)}} \log \frac{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}\Big) - E_{\mathbf{X}\sim \tilde{P}}\Big(\sum_{\substack{j:\ \mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_{k=1}^{\bar{J}} \\ \mathrm{MB}(\mathbf{C}_j^*) \neq \widehat{MB}(\mathbf{C}_j^*)}} \log \frac{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}\Big) \tag{40}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_{k=1}^{\bar{J}} \\ \mathrm{MB}(\mathbf{C}_j^*) \neq \widehat{MB}(\mathbf{C}_j^*)}} \log \frac{f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}\Big) - E_{\mathbf{X}\sim \tilde{P}}\Big(\sum_{\substack{j:\ \mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_{k=1}^{\bar{J}} \\ \mathrm{MB}(\mathbf{C}_j^*) \neq \widehat{MB}(\mathbf{C}_j^*)}} \log \frac{f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}\Big) \tag{41}$$

$$+\log \frac{\tilde{Z}}{Z} + \log \frac{Z}{\tilde{Z}}. \tag{42}$$

Recall $T = \{j : \mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_{k=1}^{\bar{J}}, \mathrm{MB}(\mathbf{C}_j^*) \neq \widehat{MB}(\mathbf{C}_j^*)\}$. Using the same reasoning as in the proof of Theorem 6, we have that for the sum of the terms in lines (35), (36), (39) and (40) to be bounded by $J\varepsilon$ with probability at least $1-\delta$, it suffices that m satisfies the condition on m in Eqn. (11) of Theorem 6 .

The sum of the terms in lines (37) and (41) can be bounded by

$$2 \sum_{j\ :\ \mathrm{MB}(\mathbf{C}_j^*) \neq \widehat{MB}(\mathbf{C}_j^*)} \max_{\mathbf{c}_j^*} \Big| \log \frac{f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{f^*_{\mathbf{C}_j^*|\widehat{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)} \Big|.$$

The sum of the terms in lines (38) can be bounded by

$$2 \sum_{j:\mathbf{D}_j^* \notin \{\mathbf{C}_k^*\}_{k=1}^{J^*}} \max_{\mathbf{d}_j^*} \left| \log f_{\mathbf{D}_j^*}^*(\mathbf{d}_j) \right|.$$

The two terms in line (42) sum to zero.
This establishes the theorem. ■

Appendix E. Proofs for Section 3.5

We will treat the proofs for the factor graph case and the Bayesian network case in two separate sections.

E.1 Proof of Theorem 8

Proof [Theorem 8] Using the same reasoning as in the proof of Theorem 6, we get that for any fixed $j \in \{1, \cdots, J^*\}$ for

$$|\log f_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}^*(\mathbf{c}_j^*) - \log \hat{f}_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}^*(\mathbf{c}_j^*)| \leq \frac{\varepsilon'}{2^{k+1}} \tag{43}$$

to hold for all instantiations $\mathbf{c}_j^*$ with probability $1-\delta'$ it suffices that

$$m \geq \frac{(1+\frac{\varepsilon'}{2^{2k+2}})^2}{2\gamma^{2k+2b}(\frac{\varepsilon'}{2^{2k+2}})^2} \log \frac{4v^{k+b}}{\delta'}. \tag{44}$$

Also, using the same reasoning as in the proof of Lemma 20, we get that

$$D_n(P\|\tilde{P}) + D_n(\tilde{P}\|P) \leq \frac{2}{n} \sum_{j=1}^{J^*} \max_{\mathbf{c}_j^*} |\log f_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}^*(\mathbf{c}_j^*) - \log \hat{f}_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}^*(\mathbf{c}_j^*)|.$$

We have for all factors and instantiations that (recall that 2^k probabilities contribute to each factor, and each probability is over k variables, thus each (log) conditional probability has maximal skew $\log \frac{(1-\gamma)^k}{\gamma^k} \leq k \log \frac{1}{\gamma}$)

$$|\log f_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}^*(\mathbf{c}_j^*) - \log \hat{f}_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}^*(\mathbf{c}_j^*)| \leq k 2^k \log \frac{1}{\gamma}.$$

Note that clipping of the probability estimates ensures this holds with probability one. Thus we get that

$$\begin{aligned} \mathrm{E}\big(D_n(P\|\tilde{P}) + D_n(\tilde{P}\|P)\big) &\leq \frac{2}{n} J^* \frac{\varepsilon'}{2^{k+1}} + \delta' \frac{2}{n} J^* k 2^k \log \frac{1}{\gamma} \\ &\leq \frac{J}{n}\varepsilon' + \frac{J}{n} k 2^{2k+1} \delta' \log \frac{1}{\gamma}, \end{aligned}$$

where for the last inequality we used $J^* \leq 2^k J$. The Markov inequality ($P(X \leq \alpha) \geq 1 - \frac{\mathrm{E}X}{\alpha}$) gives us that

$$D_n(P\|\tilde{P}) + D_n(\tilde{P}\|P) \leq \varepsilon \tag{45}$$

holds with probability

$$1 - \frac{\frac{J}{n}\varepsilon' + \frac{J}{n}k2^{2k+1}\delta' \log \frac{1}{\gamma}}{\varepsilon}.$$

Now choosing ε', δ' such that $\frac{\delta}{2} = \frac{J}{n}\frac{\varepsilon'}{\varepsilon} = \frac{J}{n}\frac{k2^{2k+1}\delta' \log \frac{1}{\gamma}}{\varepsilon}$ and substituting this back into the sufficient condition on m, gives us that for Eqn. (45) to hold with probability $1-\delta$, it suffices that

$$m \geq \frac{(1+\frac{n}{J}\frac{\varepsilon\delta}{2^{2k+3}})^2}{2\gamma^{2k+2b}(\frac{n}{J}\frac{\varepsilon\delta}{2^{2k+3}})^2} \log \frac{k2^{2k+4}v^{k+b}\log\frac{1}{\gamma}}{\frac{n}{J}\varepsilon\delta}.$$

Since the number of factors per variables is bounded by a constant, we have that $\frac{J}{n}$ is bounded by that constant. And thus we have that m is $O(1)$ when considering only the dependence on n, the number of variables. ■

E.2 Proof of Theorem 9

For clarity of the overall proof structure of Theorem 9, we defer the proofs of the helper lemmas to the next section. Note the theorem stated in this section is stronger than Theorem 9: it includes dependencies of m on ε, δ, the maximum domain size of the variables v, and the maximum number of parents k. It also shows the graceful degradation for the case of learning a distribution that does not factor according to the given structure.

For any $\gamma < \frac{1}{v}$, and any multinomial distribution with means $\theta_{1:v}$, the multinomial distribution with means clipped to $[\gamma, 1-\gamma]$ refers to the distribution obtained by clipping every θ_i to $[\gamma, 1-\gamma]$, after which the θ_i are adjusted to sum to one, while kept in the interval $[\gamma, 1-\gamma]$. It is easily verified this is always possible without changing any θ_i by more than $v\gamma$. (Although the adjustment such that the entries sum to one need not be unique, it does not matter for our results which choice is made.) We write $D(\theta^{(1)}_{1:v} \| \theta^{(2)}_{1:v})$ as a shortcut for $D(P_1 \| P_2)$, where P_1, P_2 are multinomial distributions with means $\theta^{(1)}_1, \ldots, \theta^{(1)}_v$ and $\theta^{(2)}_1, \ldots, \theta^{(2)}_v$ respectively. The following lemmas establish the basic results used to prove our main sample complexity bounds for Bayesian networks parameter learning.

Lemma 21 *Let any $\delta > 0, \varepsilon > 0$ be fixed, and let there be m i.i.d. samples drawn from a v-valued multinomial distribution with means $\theta^*_{1:v}$, and let $\tilde{\theta}_{1:v}$ be the empirical distribution, clipped to the interval $[\frac{\varepsilon}{4v^3}, 1-\frac{\varepsilon}{4v^3}]$. Then if $m \geq \frac{8v^4}{\varepsilon^2}\log\frac{2v}{\delta}$, we have that $D(\theta^*_{1:v} \| \tilde{\theta}_{1:v}) \leq \varepsilon$ w.p. $1-\delta$.*

Lemma 22 *Let two v-valued multinomial distributions with means $\theta^{(1)}_{1:v} \in [0,1]^v, \theta^{(2)}_{1:v} \in [\gamma, 1-\gamma]^v$ be given. Then we have that $D(\theta^{(1)}_{1:v} \| \theta^{(2)}_{1:v}) \leq \log\frac{1}{\gamma}$.*

Lemma 23 *Let m_H be the sum of m i.i.d. Bernoulli(p) random variables. If $m \geq \frac{8}{p}\log\frac{1}{\delta}$, then we have that $m_H \geq \frac{mp}{2}$ with probability $1-\delta$.*

Lemma 24 *Let $\{X_i\}_{i=1}^{k+1}$ be a set of $k+1$ random variables with $|\mathrm{val}(X_i)| \leq v$ for all $i = 1:k+1$. Let $\mathbf{u} \in \mathrm{val}(X_{1:k})$. Let any $\varepsilon > 0, \delta > 0$ be given. Let $\tilde{P}(X_{k+1}|X_{1:k} = \mathbf{u})$ be the empirical estimate of $X_{k+1}|X_{1:k} = \mathbf{u}$ (based on m independent samples of $\{X_i\}_{i=1}^{k+1}$ drawn from $P(X_{1:k+1})$) clipped to the interval $[\frac{\varepsilon}{4|\mathrm{val}(X_{k+1})|^3}, 1-\frac{\varepsilon}{4|\mathrm{val}(X_{k+1})|^3}]$. Then to ensure that $D(P(X_{k+1}|X_{1:k} = \mathbf{u}) \| \tilde{P}(X_{k+1}|X_{1:k} = \mathbf{u})) \leq \frac{\varepsilon}{v^{k/2}\sqrt{P(X_{1:k}=\mathbf{u})}}$ w.p. $1-\delta$, it suffices that $m \geq \frac{16\, v^{4+k}}{\varepsilon^2}\log^2\frac{4v^3}{\varepsilon}\log\frac{4v}{\delta}$.*

Lemma 25 *Let $\{X_i\}_{i=1}^{k+1}$ be a set of $k+1$ random variables with $\mathrm{val}(X_i) \leq v$ for all $i = 1 : k+1$. Let any $\varepsilon > 0, \delta > 0$ be given. For all $\mathbf{u} \in \mathrm{val}(X_{1:k})$ let $\tilde{P}(X_{k+1}|X_{1:k} = \mathbf{u})$ be the empirical estimate of $X_{k+1}|X_{1:k} = \mathbf{u}$ (based on m independent samples of $\{X_i\}_{i=1}^{k+1}$ drawn from $P(X_{1:k+1})$) clipped to the interval $[\frac{\varepsilon}{4|\mathrm{val}(X_{k+1})|^3}, 1 - \frac{\varepsilon}{4|\mathrm{val}(X_{k+1})|^3}]$. Then for $\sum_{\mathbf{u} \in \mathrm{val}(X_{1:k})} P(X_{1:k} = \mathbf{u}) D(P(X_{k+1}|X_{1:k} = \mathbf{u}) \| \tilde{P}(X_{k+1}|X_{1:k} = \mathbf{u})) \leq \varepsilon$ to hold with probability $1 - \delta$, it suffices that $m \geq \frac{16\, v^{4+k}}{\varepsilon^2} \log^2 \frac{4v^3}{\varepsilon} \log \frac{4v^{k+1}}{\delta}$.*

Because KL divergence can be unbounded, typically some process, such as clipping, is needed to ensure that our algorithms do not suffer infinite loss. Lemmas 21 and 22 show that we can bound the KL divergence by clipping the (estimated) probabilities away from $\{0,1\}$. (Abe et al. (1991) and Abe et al. (1992) give a more detailed treatment of uniform convergence for KL divergence loss.) Lemma 24 shows how to bound our error on individual conditional probability table (CPT) entries. Note that in Lemma 24, the loss is allowed to be larger for less likely instantiations of the conditioning variables. Also note that Lemma 24 shows that the number of samples *m* required does not depend on the probability of the instantiations of the conditioning variables, no matter how likely/unlikely. Lemma 23 is used in our proof of Lemma 24 to relate the required number of samples with a specific instantiation $\mathbf{u}$ of the conditioning variables to the actual number of training examples required. Lemma 25 relates the loss on individual CPT entries to the conditional KL divergence, and follows directly from Lemma 24 and Cauchy-Schwarz.

Using the lemmas above, we are now ready to prove a bound on the sample complexity of learning a fixed structure BN. We note that Dasgupta (1997) showed a bound on the sample complexity of BN learning that was polynomial in the number of variables *n*. His proof method relied on using a Union bound to show that *all* of the *n* nodes in the BN will have accurate CPT entries, which meant the bound necessarily had to have a dependence on *n* (even if the normalized KL criterion had been used). For the normalized KL criterion, his method gives a logarithmic dependence on *n*. Below, we will derive a strictly stronger bound, which has no dependence on the number of variables in the BN. Our bound is based on showing that (i) Given any fixed node, with high probability, its CPT entries will be accurate (Lemma 25), and (ii) Using the Markov inequality to show that, as a consequence, *almost all* of the nodes in the network will have CPT entries that are accurate. This turns out to be sufficient to ensure the estimated BN parameters will provide a good approximation to the joint distribution, and eliminates the bound's dependence on *n*.

In the theorem below, P is some "true" underlying distribution from which the samples are drawn; P_{BN} is the best possible approximation to P using a given BN structure (in the sense of minimizing $D_n(P\|\cdot)$), and $\tilde{P}_{BN}$ is the learned estimate of P. We give a bound on the number of training examples required for $\tilde{P}_{BN}$'s performance to approach that of P_{BN}.

Theorem 26 *Let any $\varepsilon > 0$ and $\delta > 0$ be fixed. Let P be any probability distribution over n multinomial random variables $X_{1:n}$, where each of the random variables X_i can take on at most v values. Let any BN structure be given, and let k be the maximum number of parents per variable. (P may not factor according to the BN structure.) Let P_{BN} be the best possible estimate of P using a model that factorizes according to the BN structure. (I.e., P_{BN}'s conditional probability distributions satisfy $P_{BN}(X_i|PaX_i) = P(X_i|PaX_i)$.) Let $\tilde{P}_{BN}$ denote the probability distribution obtained by fitting (via maximum likelihood) a BN model with the given structure to the m i.i.d. training examples drawn from P, and then clipping for each X_j each CPT entry to the interval $[\frac{\varepsilon}{8|\mathrm{val}(X_j)|^3}, 1 - \frac{\varepsilon}{8|\mathrm{val}(X_j)|^3}]$. Then, to ensure that with probability $1 - \delta$, $\tilde{P}_{BN}$ is nearly as good an estimate as P_{BN} of the true distribution*

P, that is, we have

$$D_n(P\|\tilde{P}_{BN}) \leq D_n(P\|P_{BN}) + \varepsilon,$$

it suffices that the training set size be

$$m \geq 64\frac{v^{4+k}\log^2\frac{8v^3}{\varepsilon}}{\varepsilon^2}\log(\frac{8v^{k+1}}{\varepsilon\delta}\log\frac{8v^3}{\varepsilon}).$$

Remark. *Note that if P does factor according to the given BN structure, then the term $D_n(P\|P_{BN})$ above equals zero.*

Proof *The following equality is easily verified:*

$$\begin{aligned} D(P\|\tilde{P}_{BN}) &= D(P\|P_{BN}) \\ &\quad + \sum_{j=1}^{n}\sum_{u\in \mathrm{val(PaX_j)}} P(PaX_j=\mathbf{u})D(P(X_j|PaX_j=\mathbf{u})\|\tilde{P}(X_j|PaX_j=\mathbf{u})). \end{aligned} \tag{46}$$

From Lemma 25 we have that for estimates clipped to $[\frac{\varepsilon'}{4|\mathrm{val(X_j)}|^3}, 1-\frac{\varepsilon'}{4|\mathrm{val(X_j)}|^3}]$, that for

$$\sum_{\mathbf{u}\in\mathrm{val(PaX_j)}} P(PaX_j=\mathbf{u})D(P(X_j|PaX_j=\mathbf{u})\|\tilde{P}(X_j|PaX_j=\mathbf{u})) \leq \varepsilon'$$

to hold with probability $1-\tau$, it suffices that

$$m \geq 16\frac{v^{4+k}\log^2\frac{4v^3}{\varepsilon'}}{\varepsilon'^2}\log\frac{4v^{k+1}}{\tau}. \tag{47}$$

Now let $Z=\sum_i \eta_i$ be the sum over indicator variables $\eta_i = \mathbf{1}\{\sum_{\mathbf{u}} P(PaX_i=\mathbf{u})D(P(X_j|PaX_j=\mathbf{u})\|\tilde{P}(X_j|PaX_j=\mathbf{u})) > \varepsilon'\}$, and let τ be as above. Then applying the Markov inequality to the non-negative random variable Z gives

$$P(\textstyle\sum_{i=1}^{n}\eta_i \leq \frac{n\tau}{\delta}) \geq 1-\delta. \tag{48}$$

So, we have that

$$\begin{aligned} D_n(P\|\tilde{P}_{BN}) &\leq D_n(P\|P_{BN}) + \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(1-\eta_i)\varepsilon' + \tfrac{1}{n}\sum_{i=1}^{n}\eta_i\log\frac{4v^3}{\varepsilon'} \\ &\leq D_n(P\|P_{BN}) + \varepsilon' + \tfrac{\tau}{\delta}\log\tfrac{4v^3}{\varepsilon'} \quad \textit{w.p. } 1-\delta. \end{aligned} \tag{49}$$

For the first inequality we used Eqn. (46), the definition of η_i and Lemma 22. The second inequality follows from Eqn. (48). To bound the right hand side of Eqn. (49), we bound each of the terms by $\frac{\varepsilon}{2}$. For the first term this implies $\varepsilon' = \frac{\varepsilon}{2}$, for the second term, this allows us to solve for the free parameter $\tau = \frac{\varepsilon\delta}{2\log\frac{4v^3}{\varepsilon'}} = \frac{\varepsilon\delta}{2\log\frac{8v^3}{\varepsilon}}$. Substituting these expressions for ε' and τ into Eqn. (47, 49), gives the statement of the theorem. ■

Note that Eqn. (46) holds for all distributions $\tilde{P}_{BN}$ that factor according to the BN. Since KL divergence is always non-negative, Eqn. (46) implies that $D(P\|P_{BN}) \leq D(P\|\tilde{P}_{BN})$. So the clipped maximum likelihood learning achieves the minimal KL divergence loss for infinite sample size.

Also note that in general, $D(P\|\tilde{P}_{BN})$ is *not* equal to $D(P\|P_{BN})+D(P_{BN}\|\tilde{P}_{BN})$. In particular, the second term in Eqn. (46) is *not* equal to $D(P_{BN}\|\tilde{P}_{BN})$, since $P_{BN}(PaX_j=\mathbf{u})$ is (in general) not equal to $P(PaX_j=\mathbf{u})$. (In contrast, for log-linear models/undirected graphical models a decomposition of the KL-divergence does hold.[17])

E.3 Proofs of Lemmas 21, 22, 23, 24, 25

We will first state and prove two lemmas that are used to subsequently prove lemmas 21, 22, 23, 24, 25.

Lemma 27 *For any* $\theta^{(1)}_{1:v},\theta^{(2)}_{1:v}\in[0,1]^v,\sum_{i=1}^v\theta^{(1)}_i=1,\sum_{i=1}^v\theta^{(2)}_i=1$, *we have*

$$D(\theta^{(1)}_{1:v}\|\theta^{(2)}_{1:v})\leq\sum_{i=1}^v\frac{(\theta^{(1)}_i-\theta^{(2)}_i)^2}{\theta^{(2)}_i}.$$

Proof *We use the concavity of the* log *function, to upper bound it with a tangent line at* $\theta^{(2)}_i$, *which gives the following inequality:*

$$\log\theta^{(1)}_i\leq\log\theta^{(2)}_i+\frac{1}{\theta^{(2)}_i}(\theta^{(1)}_i-\theta^{(2)}_i). \tag{50}$$

Substituting Eqn. (50) into the definition of $D(\theta^{(1)}_{1:v}\|\theta^{(2)}_{1:v})$ *gives us:*

$$D(\theta^{(1)}_{1:v}\|\theta^{(2)}_{1:v})\leq\sum_{i=1}^v\frac{\theta^{(1)}_i}{\theta^{(2)}_i}(\theta^{(1)}_i-\theta^{(2)}_i).$$

Adding $0=\sum_{i=1}^v\frac{-\theta^{(2)}_i}{\theta^{(2)}_i}(\theta^{(1)}_i-\theta^{(2)}_i)$ *to the right hand side gives:*

$$D(\theta^{(1)}_{1:v}\|\theta^{(2)}_{1:v})\leq\sum_{i=1}^v\frac{1}{\theta^{(2)}_i}(\theta^{(1)}_i-\theta^{(2)}_i)^2,$$

which proves the theorem. ■

Lemma 28 *For any v-valued multinomial distributions with means* $\theta^{(1)}_{1:v}\in[0,1]^v$ *and* $\theta^{(2)}_{1:v}\in[\gamma,1-\gamma]^v$, *with* $\gamma<\frac{1}{v}$, *we have*

$$D(\theta^{(1)}_{1:v}\|\theta^{(2)}_{1:v})\leq\sum_{i=1}^v\frac{(\theta^{(1)}_i-\theta^{(2)}_i)^2}{\gamma}.$$

Proof *Immediately from Lemma 27, since* $\frac{1}{\theta^{(2)}_i}\leq\frac{1}{\gamma}$. ■

17. Let P be any distribution, let $\{P_\theta\}$ be a family of log-linear models parameterized by θ, let $\theta^*=\arg\min_\theta D(P\|P_\theta)$. Then we do have that $D(P\|P_\theta)=D(P\|P_{\theta^*})+D(P_{\theta^*}\|P_\theta)$. The proof relies on the fact that θ^* is such that $E_P[\eta_i]=E_{P_{\theta^*}}[\eta_i],\forall i$, with η_i the natural parameters of the log-linear model (see, for example, Kullback (1959)). Due to the local normalization constraints, this is not true in BN's.

Proof [Lemma 21] Let $\hat{\theta}_{1:v}$ be the unclipped sample means. The triangle inequality gives for any $i \in \{1,\cdots,v\}$:

$$|\theta_i^* - \tilde{\theta}_i| \leq |\theta_i^* - \hat{\theta}_i| + |\hat{\theta}_i - \tilde{\theta}_i|. \tag{51}$$

From the Hoeffding inequality and the Union bound we have that for all $i \in \{1,\cdots,v\}$ for

$$|\theta_i^* - \hat{\theta}_i| \leq \varepsilon' \tag{52}$$

to hold w.p. $1-\delta$, it suffices that

$$m \geq \frac{1}{2(\varepsilon')^2} \log \frac{2v}{\delta}. \tag{53}$$

Since $\tilde{\theta}_{1:v}$ are obtained by clipping $\hat{\theta}_{1:v}$ into $[\gamma, 1-\gamma]$ (for now γ is a free parameter, which will soon be matched to the clipping choice of $\frac{\varepsilon}{4v^3}$ of the lemma), we have that (see introduction of previous section)

$$|\hat{\theta}_i - \tilde{\theta}_i| \leq v\gamma. \tag{54}$$

Using Lemma 28 and then Eqn. (51), (52), and (54) we have that

$$D(\theta_{1:v} \| \tilde{\theta}_{1:v}) \leq \sum_{i=1}^{v} \frac{(\theta_i^* - \tilde{\theta}_i)^2}{\gamma} \leq v \frac{(\varepsilon' + v\gamma)^2}{\gamma} \tag{55}$$

holds w.p. $1-\delta$ if m satisfies Eqn. (53). The choice of $\gamma = \frac{\varepsilon'}{v}$ minimizes the right hand side of Eqn. (55), and gives us that

$$D(\theta_{1:v} \| \tilde{\theta}_{1:v}) \leq 4v^2\varepsilon'. \tag{56}$$

Now choosing $\varepsilon' = \frac{\varepsilon}{4v^2}$ (corresponding to $\gamma = \frac{\varepsilon}{4v^3}$) gives us that for

$$D(\theta_{1:v} \| \tilde{\theta}_{1:v}) \leq \varepsilon$$

to hold w.p. $1-\delta$ it suffices that

$$m \geq \frac{8v^4}{\varepsilon^2} \log \frac{2v}{\delta},$$

which proves the lemma. ■

Proof [Lemma 22] We have

$$\begin{aligned} D(\theta_{1:v}^{(1)} \| \theta_{1:v}^{(2)}) &= \sum_{i=1}^{v} \theta_i^{(1)} \log \frac{\theta_i^{(1)}}{\theta_i^{(2)}} \\ &\leq \max_i \log \frac{\theta_i^{(1)}}{\theta_i^{(2)}} \\ &\leq \max_i \log \frac{1}{\theta_i^{(2)}} \\ &\leq \max_{y \in [\gamma, 1-\gamma]} \log \frac{1}{y} \\ &= \log \frac{1}{\gamma}, \end{aligned}$$

which proves the lemma. ■

Proof [Lemma 23] Let $\hat{p} = \frac{m_H}{m}$, then

$$Pr(m_H \leq \frac{pm}{2}) = Pr(\hat{p} \leq \frac{p}{2}) = Pr(\frac{p-\hat{p}}{\sqrt{p}} \geq \frac{\sqrt{p}}{2}).$$

Applying the (multiplicative) Chernoff bound gives

$$Pr(m_H \leq \frac{pm}{2}) \leq \exp(\frac{-pm}{8}) = \delta,$$

where the last equality defines δ. Solving the last equation for m shows that $m = \frac{8}{p}\log\frac{1}{\delta}$ samples are sufficient to guarantee $Pr(m_H > \frac{pm}{2}) \geq 1-\delta$, which is the statement of the lemma. ■

Proof [Lemma 24] Below let $\theta_i = P(X_{k+1} = i | X_{1:k} = \mathbf{u})$, let $\tilde{\theta}_i = \tilde{P}(X_{k+1} = i | X_{1:k} = \mathbf{u})$, and let $\bar{v} = |\mathrm{val}(\mathrm{X_{k+1}})|$. We split the proof into 2 cases

1. $\frac{\varepsilon}{v^{k/2}\sqrt{P(X_{1:k}=\mathbf{u})}} \geq \log\frac{4v^3}{\varepsilon}$ This case is trivial, since by Lemma 22 we have that $D(\theta_{1:\bar{v}} \| \tilde{\theta}_{1:\bar{v}}) \leq \log\frac{4\bar{v}^3}{\varepsilon} \leq \log\frac{4v^3}{\varepsilon}$ and the statement of the lemma is trivially implied, for all $\tilde{\theta}_{1:\bar{v}} \in [\frac{4\bar{v}^3}{\varepsilon}, 1 - \frac{4\bar{v}^3}{\varepsilon}]^{\bar{v}}$, so $m = 0$ samples is sufficient.

2. $\frac{\varepsilon}{v^{k/2}\sqrt{P(X_{1:k}=\mathbf{u})}} < \log\frac{4v^3}{\varepsilon}$ Let $m_{\mathbf{u}}$ be the number of samples for which $X_{1:k} = \mathbf{u}$. Then (using Lemma 21 and $\bar{v} \leq v$) a number of samples

$$m_{\mathbf{u}} \geq \frac{8v^4\, v^k P(X_{1:k} = \mathbf{u})}{\varepsilon^2} \log\frac{2v}{\delta'} \tag{57}$$

is sufficient to guarantee that $D(\theta_{1:\bar{v}} \| \tilde{\theta}_{1:\bar{v}}) \leq \frac{\varepsilon}{v^{k/2}\sqrt{P(X_{1:k}=\mathbf{u})}}$ with probability $1-\delta'$. To obtain, with probability $1-\delta''$, at least $m_{\mathbf{u}}$ samples from $P(X_{1:k+1})$ for which $X_{1:k} = \mathbf{u}$, it suffices that the total number of samples m from $P(X_{1:k+1})$ satisfies

$$m \geq \max\{\frac{8}{P(X_{1:k}=\mathbf{u})}\log\frac{1}{\delta''}, \frac{2m_{\mathbf{u}}}{P(X_{1:k}=\mathbf{u})}\},$$

where we used Lemma 23. Using $P(X_{1:k} = \mathbf{u}) \geq \frac{\varepsilon^2}{v^k \log^2 \frac{4v^3}{\varepsilon}}$ (we are in case 2) and (57), and setting $\delta' = \delta/2$, $\delta'' = \delta/2$, gives the statement of the lemma.

■

Proof [Lemma 25] Using Lemma 24 and the union bound over all instantiations $\mathbf{u}$ of $X_{1:k}$ (there are at most v^k instantiations) we get that for

$$\forall \mathbf{u} \in \mathrm{val}(\mathrm{X_{1:k}})\;\; D(P(X_{k+1}|X_{1:k}=\mathbf{u}) \| \tilde{P}(X_{k+1}|X_{1:k}=\mathbf{u})) \leq \frac{\varepsilon}{v^{k/2}\sqrt{P(X_{1:k}=\mathbf{u})}} \tag{58}$$

to hold with probability $1-\delta$, it suffices that

$$m \geq \frac{16v^{4+k}\log^2\frac{4v^3}{\varepsilon}}{\varepsilon^2}\log\frac{4v^{k+1}}{\delta}. \tag{59}$$

So we have that the following inequalities hold w.p. $1-\delta$ if m satisfies Eqn. (59):

$$\begin{aligned}
&\sum_{\mathbf{u}\in \mathrm{val}(X_{1:k})} P(X_{1:k}=\mathbf{u})D(P(X_{k+1}|X_{1:k}=\mathbf{u})\|\tilde{P}(X_{k+1}|X_{1:k}=\mathbf{u})) \\
&\leq \sum_{\mathbf{u}} P(X_{1:k}=\mathbf{u})\frac{\varepsilon}{v^{\frac{k}{2}}\sqrt{P(X_{1:k}=\mathbf{u})}} \\
&\leq \sum_{\mathbf{u}} \varepsilon\frac{\sqrt{P(X_{1:k}=\mathbf{u})}}{v^{k/2}} \\
&\leq \varepsilon,
\end{aligned}$$

where we used in order: Eqn. (58), simplification, Cauchy-Schwarz. The last inequality together with the condition in Eqn. (59) prove the lemma. ■

Appendix F. Proofs for Section 4

In this section we give formal proofs of all theorems, propositions and lemmas appearing in Section 4.

F.1 Proof of Lemma 12

We first prove the following lemma.

Lemma 29 *Let any $\varepsilon>0,\delta>0$ be given. Let any $\lambda\in(0,1)$ be given. Let $\{X_i\}_{i=1}^m$ be i.i.d. Bernoulli(ϕ) random variables, where $\lambda\leq\phi\leq 1-\lambda$. Let $\hat{\phi}=\frac{1}{m}\sum_{i=1}^m X_i$. Then for*

$$|\phi\log\phi-\hat{\phi}\log\hat{\phi}|\leq\varepsilon$$

to hold w.p. $1-\delta$, it suffices that

$$m\geq\max\{\frac{2}{\lambda^2}\log\frac{2}{\delta},\frac{2}{\lambda^2\varepsilon^2}\log\frac{2}{\delta}\}.$$

Proof *From the Hoeffding inequality we have that for*

$$|\phi-\hat{\phi}|\leq\varepsilon'$$

to hold w.p. $1-\delta$ it suffices that

$$m\geq\frac{1}{2\varepsilon'^2}\log\frac{2}{\delta}. \tag{60}$$

Now since the function $f(x)=x\log x$ is Lipschitz with Lipschitz-constant smaller than $\max\{1,|\log(\lambda-\varepsilon')|\}$ over the interval $[\lambda-\varepsilon',1]$, we have that for

$$|\phi\log\phi-\hat{\phi}\log\hat{\phi}|\leq\varepsilon'\max\{1,|\log(\lambda-\varepsilon')|\}$$

to hold w.p. $1-\delta$*, it suffices that m satisfies Eqn. (60). If we choose* ε' *such that* $\varepsilon' \leq \lambda/2$ *we get*

$$|\phi \log \phi - \hat{\phi} \log \hat{\phi}| \leq \varepsilon' \max\{1, |\log(\lambda/2)|\}.$$

To ensure the right hand side is smaller than ε*, it suffices that the following three conditions are satisfied:*

$$\begin{aligned} \varepsilon' &\leq \frac{\lambda}{2}, \\ \varepsilon' &\leq \varepsilon, \\ \varepsilon' &\leq \varepsilon\lambda/2 \leq \varepsilon/|\log\frac{\lambda}{2}|. \end{aligned}$$

The last inequality holds since $\lambda \in (0,1)$*. Since* $\lambda \in (0,1)$ *we can simplify this to the following two conditions:*

$$\begin{aligned} \varepsilon' &\leq \frac{\lambda}{2}, \\ \varepsilon' &\leq \varepsilon\lambda/2. \end{aligned}$$

Substituting this into Eqn. (60) gives us the condition for m as in the statement of the lemma. ■

Proof [Lemma 12] We abbreviate $P(\mathbf{X}=\mathbf{x},\mathbf{Y}=\mathbf{y})$ as $P(\mathbf{x},\mathbf{y})$ and similarly for $\hat{P},\mathbf{x},\mathbf{y},\mathbf{x}|\mathbf{y}$. We abbreviate $\sum_{\mathbf{x}\in \mathrm{val}(\mathbf{X})}$ by $\sum_{\mathbf{x}}$ and similarly for $\mathbf{y}$.

$$\begin{aligned} \left|H(\mathbf{X}|\mathbf{Y}) - \widehat{H}(\mathbf{X}|\mathbf{Y})\right| &= \left|\sum_{\mathbf{x},\mathbf{y}} P(\mathbf{x},\mathbf{y}) \log P(\mathbf{x}|\mathbf{y}) - \sum_{\mathbf{x},\mathbf{y}} \hat{P}(\mathbf{x},\mathbf{y}) \log \hat{P}(\mathbf{x}|\mathbf{y})\right| \\ &= \Big|\sum_{\mathbf{x},\mathbf{y}} P(\mathbf{x},\mathbf{y}) \log P(\mathbf{x},\mathbf{y}) - \sum_{\mathbf{y}} P(\mathbf{y}) \log P(\mathbf{y}) \\ &\quad - \sum_{\mathbf{x},\mathbf{y}} \hat{P}(\mathbf{x},\mathbf{y}) \log \hat{P}(\mathbf{x},\mathbf{y}) + \sum_{\mathbf{y}} \hat{P}(\mathbf{y}) \log \hat{P}(\mathbf{y})\Big| \\ &\leq \sum_{\mathbf{x},\mathbf{y}} \left|P(\mathbf{x},\mathbf{y}) \log P(\mathbf{x},\mathbf{y}) - \hat{P}(\mathbf{x},\mathbf{y}) \log \hat{P}(\mathbf{x},\mathbf{y})\right| \\ &\quad + \sum_{\mathbf{y}} \left|P(\mathbf{y}) \log P(\mathbf{y}) - \hat{P}(\mathbf{y}) \log \hat{P}(\mathbf{y})\right| \end{aligned}$$

Now using Lemma 29 (and the Union bound) we get that for

$$\left|H(\mathbf{X}|\mathbf{Y}) - \widehat{H}(\mathbf{X}|\mathbf{Y})\right| \leq |\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|\varepsilon' + |\mathrm{val}(\mathbf{Y})|\varepsilon'$$

to hold w.p. $1 - |\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|\delta' - |\mathrm{val}(\mathbf{X})|\delta'$, it suffices that

$$m \geq \max\{\frac{2}{\lambda^2\varepsilon'^2}\log\frac{2}{\delta'}, \frac{2}{\lambda^2}\log\frac{2}{\delta'}\}.$$

Choosing $\varepsilon = \varepsilon'/(2|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|)$ and $\delta = \delta'/(2|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|)$ gives that for

$$|H(\mathbf{X}|\mathbf{Y}) - \widehat{H}(\mathbf{X}|\mathbf{Y})| \leq \varepsilon$$

to hold with probability $1-\delta$, it suffices to have

$$m \geq \max\{\frac{8|\mathrm{val}(\mathbf{X})|^2|\mathrm{val}(\mathbf{Y})|^2}{\lambda^2\varepsilon^2}\log\frac{4|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|}{\delta}, \frac{2}{\lambda^2}\log\frac{4|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|}{\delta}\}. \quad (61)$$

Now since for any two distributions P and $\hat{P}$ we have $|H(\mathbf{X}|\mathbf{Y}) - \widehat{H}(\mathbf{X}|\mathbf{Y})| \leq \log|\mathrm{val}(\mathbf{X})| \leq 2|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|$, we have that for any $\varepsilon \geq 2|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|$ the statement of the lemma holds trivially independent of the number of samples m. Thus we can simplify the conditions on m in Eqn. (61) to one condition:

$$m \geq \frac{8|\mathrm{val}(\mathbf{X})|^2|\mathrm{val}(\mathbf{Y})|^2}{\lambda^2\varepsilon^2}\log\frac{4|\mathrm{val}(\mathbf{X})||\mathrm{val}(\mathbf{Y})|}{\delta},$$

which proves the lemma. ■

F.2 Proof of Lemma 13

We abbreviate $P(\mathbf{X}=\mathbf{x})$ as $P(\mathbf{x})$ and similarly for other variables.

Proof [Lemma 13] Using Eqn. (14) and the definition of conditional entropy we get that

$$\sum_{\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y}} P(\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y})\log P(\mathbf{x}|\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y}) - \sum_{\mathbf{x},\mathbf{u},\mathbf{w}} P(\mathbf{x},\mathbf{u},\mathbf{w})\log P(\mathbf{x}|\mathbf{u},\mathbf{w}) \leq \varepsilon.$$

We can rewrite this as

$$\sum_{\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y}} P(\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y})\log\frac{P(\mathbf{x}|\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y})}{P(\mathbf{x}|\mathbf{u},\mathbf{w})} \leq \varepsilon.$$

Now using Eqn. (13) ($\mathbf{U}\cup\mathbf{V}$ is the Markov blanket of $\mathbf{X}$) gives us

$$\sum_{\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y}} P(\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y})\log\frac{P(\mathbf{x}|\mathbf{u},\mathbf{v})}{P(\mathbf{x}|\mathbf{u},\mathbf{w})} \leq \varepsilon.$$

We can simplify this to

$$\sum_{\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w}} P(\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{w})\log\frac{P(\mathbf{x}|\mathbf{u},\mathbf{v})}{P(\mathbf{x}|\mathbf{u},\mathbf{w})} \leq \varepsilon.$$

Using the definition of conditional probability and Eqn. (13) ($\mathbf{U}\cup\mathbf{V}$ is the Markov blanket of $\mathbf{X}$) we get

$$\sum_{\mathbf{u},\mathbf{v},\mathbf{w}} P(\mathbf{u},\mathbf{v},\mathbf{w})\sum_{\mathbf{x}} P(\mathbf{x}|\mathbf{u},\mathbf{v})\log\frac{P(\mathbf{x}|\mathbf{u},\mathbf{v})}{P(\mathbf{x}|\mathbf{u},\mathbf{w})} \leq \varepsilon.$$

Now since $\lambda_1 \leq P(\mathbf{u},\mathbf{v},\mathbf{w})$ and each term $\sum_{\mathbf{x}} P(\mathbf{x}|\mathbf{u},\mathbf{v})\log\frac{P(\mathbf{x}|\mathbf{u},\mathbf{v})}{P(\mathbf{x}|\mathbf{u},\mathbf{w})}$ is positive (it's a KL-divergence) we get that for all $\mathbf{u},\mathbf{v},\mathbf{w}$

$$\sum_{\mathbf{x}} P(\mathbf{x}|\mathbf{u},\mathbf{v})\log\frac{P(\mathbf{x}|\mathbf{u},\mathbf{v})}{P(\mathbf{x}|\mathbf{u},\mathbf{w})} \leq \frac{\varepsilon}{\lambda_1}.$$

The left hand side of this equation is the KL-divergence between a distribution $Q_{\mathbf{u},\mathbf{v},\mathbf{w}}(\mathbf{X}) = P(\mathbf{X}|\mathbf{U} = \mathbf{u}, \mathbf{V} = \mathbf{v})$ and a distribution $\hat{Q}_{\mathbf{u},\mathbf{v},\mathbf{w}}(\mathbf{X}) = P(\mathbf{X}|\mathbf{U} = \mathbf{u}, \mathbf{W} = \mathbf{w})$. Now using the KL-divergence property that $\frac{1}{2}(\sum_{\mathbf{x}} |P_1(\mathbf{x}) - P_2(\mathbf{x})|)^2 \leq D(P_1 \| P_2)$ (see, for example, Cover and Thomas, 1991, p. 300), we get that for all $\mathbf{u}, \mathbf{v}, \mathbf{w}$

$$\frac{1}{2}(\sum_{\mathbf{x}} |P(\mathbf{x}|\mathbf{u},\mathbf{v}) - P(\mathbf{x}|\mathbf{u},\mathbf{w})|)^2 \leq \frac{\varepsilon}{\lambda_1}.$$

As a consequence, we have for all $\mathbf{x}, \mathbf{u}, \mathbf{v}, \mathbf{w}$ that

$$|P(\mathbf{x}|\mathbf{u},\mathbf{v}) - P(\mathbf{x}|\mathbf{u},\mathbf{w})| \leq \sqrt{2\frac{\varepsilon}{\lambda_1}}.$$

Now since $\lambda_2 \leq P(\mathbf{x}|\mathbf{u},\mathbf{v})$ and $\lambda_2 \leq P(\mathbf{x}|\mathbf{u},\mathbf{w})$ we have that

$$|\log P(\mathbf{x}|\mathbf{u},\mathbf{v}) - \log P(\mathbf{x}|\mathbf{u},\mathbf{w})| \leq \frac{\sqrt{2\varepsilon}}{\lambda_2\sqrt{\lambda_1}}.$$

Now using Eqn. (13) ($\mathbf{U} \cup \mathbf{V}$ is the Markov blanket of $\mathbf{X}$) to substitute $P(\mathbf{x}|\mathbf{u},\mathbf{v})$ by $P(\mathbf{x}|\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{y})$, we obtain Eqn. (15). ■

F.3 Proof of Theorem 14

Proof [Theorem 14] There are $O(kn^k bn^b)$ ⟨candidate factor, candidate Markov blanket⟩ pairs, each with $O(v^{k+b})$ different instantiations. Collecting the required empirical probabilities from the data takes $O(kn^k bn^b v^{k+b} + mkn^k bn^b(k+b))$. (Similar reasoning as in the proof of Theorem 5.) Computing the empirical entropies from the empirical probabilities takes $O(kn^k bn^b v^{k+b})$. There are $O(kn^k)$ actual factors computed. From (the proof of) Theorem 5, we have that this takes $O(kn^k(m(k+b) + 2^k v^k))$. Putting it all together gives us an upper bound on the running time of

$$\begin{aligned} &O\big(kn^k bn^b v^{k+b} + mkn^k bn^b(k+b) + kn^k bn^b v^{k+b} \\ &+ \quad kn^k(m(k+b) + 2^k v^k)\big). \end{aligned}$$

After simplification we get a running time of

$$O\big(kn^k bn^b v^{k+b} + mkn^k bn^b(k+b) + kn^k 2^k v^k\big).$$

■

F.4 Proof of Theorem 15

Proof [Theorem 15] Let $\mathcal{C}$, $\mathcal{Y}$ be defined as in Eqn. (16) and Eqn. (17) of the structure learning algorithm description. For all $\mathbf{C}_j^* \in \mathcal{C}$ we have by assumption $|\text{val}(\mathbf{C}_j^*)| \leq v^k$. For all $\mathbf{Y} \in \mathcal{Y}$ we have $|\text{val}(\mathbf{Y})| \leq v^b$. Also note that $P(\mathbf{C}_j^* = \mathbf{c}_j^*, \mathbf{Y} = \mathbf{y}) \geq \frac{1}{\gamma^{k+b}}$. Using Lemma 12 we get that for any $\mathbf{C}_j^* \in \mathcal{C}, \mathbf{Y} \in \mathcal{Y}$ for

$$\left|H(\mathbf{C}_j^*|\mathbf{Y}) - \widehat{H}(\mathbf{C}_j^*|\mathbf{Y})\right| \leq \varepsilon' \tag{62}$$

to hold with probability $1-\delta'$ it suffices that

$$m \geq 8\frac{v^{2k}v^{2b}}{\gamma^{2b+2k}\varepsilon'^2}\log 4\frac{v^k v^b}{\delta'}. \tag{63}$$

Taking the union bound we get that for Eqn. (62) to hold for all $\mathbf{C}_j^* \in \mathcal{C}$ and for all $\mathbf{Y} \in \mathcal{Y}$ with probability $1-|\mathcal{C}||\mathcal{Y}|\delta'$ it suffices that m satisfies Eqn. (63).

For $\widehat{\mathrm{MB}}(\mathbf{C}_j^*) = \arg\min_{\mathbf{Y}\in\mathcal{Y},\mathbf{Y}\cap\mathbf{C}_j^*=\emptyset}\widehat{H}(\mathbf{C}_j^*|\mathbf{Y})$ we have $\widehat{\mathrm{H}}(\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)) \leq \widehat{\mathrm{H}}(\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*))$. Combining this with Eqn. (62) gives us

$$H(\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)) \leq H(\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)) + 2\varepsilon'. \tag{64}$$

From Lemma 13 we have that Eqn. (64) implies that

$$\begin{aligned} |\log P(\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)) - \log P(\mathbf{C}_j^*|\mathcal{X}-\mathbf{C}_j^*)| &\leq \frac{\sqrt{4\varepsilon'}}{\gamma^k\sqrt{\gamma^{2b}}} \\ &= \frac{2\sqrt{\varepsilon'}}{\gamma^{k+b}}. \end{aligned} \tag{65}$$

Now from Lemma 18 we have that for

$$|\log P(\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)) - \log \hat{P}(\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*))| \leq \frac{2\sqrt{\varepsilon'}}{\gamma^{k+b}} \tag{66}$$

to hold for all instantiations $\mathbf{c}_j^* \in \mathrm{val}(\mathbf{C}_j^*)$ with probability $1-\delta''$, it suffices that

$$m \geq \frac{(1+\frac{\sqrt{\varepsilon'}}{\gamma^{k+b}})^2}{2\gamma^{2k+2b}(\frac{\sqrt{\varepsilon'}}{\gamma^{k+b}})^2}\log\frac{4v^{k+b}}{\delta''}. \tag{67}$$

Using the Union bound, we get that for Eqn. (66) to hold for all $\mathbf{C}_j^* \in \mathcal{C}$ with probability $1-|\mathcal{C}|\delta''$, it suffices that m satisfies Eqn. (67). Or after simplification (and slightly loosening using $\gamma < 1$), we get the condition

$$m \geq \frac{(1+2\sqrt{\varepsilon'})^2}{2\gamma^{2k+2b}\varepsilon'}\log\frac{4v^{k+b}}{\delta''}. \tag{68}$$

Combining Eqn. (66) and Eqn. (65) gives us

$$|\log P(\mathbf{C}_j^*|\mathcal{X}-\mathbf{C}_j^*) - \log \hat{P}(\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*))| \leq \frac{4\sqrt{\varepsilon'}}{\gamma^{k+b}}.$$

From Lemma 19 we have that this implies

$$|\log f^*_{\mathbf{C}_j^*|\mathcal{X}-\mathbf{C}_j^*}(\mathbf{c}_j^*) - \log \hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)| \leq 2^{k+2}\frac{\sqrt{\varepsilon'}}{\gamma^{k+b}}.$$

Now choosing $\varepsilon' = (\frac{\varepsilon}{2^{k+2}})^2\frac{\gamma^{2k+2b}}{2^{2k+4}}$ gives us that

$$|\log f^*_{\mathbf{C}_j^*|\mathcal{X}-\mathbf{C}_j^*}(\mathbf{c}_j^*) - \log \hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)| \leq \frac{\varepsilon}{2^{k+2}}.$$

The clipping to one of factor entries $\hat{f}^*_{\mathbf{C}^*_j|\widehat{\mathrm{MB}}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)$ satisfying $|\log \hat{f}^*_{\mathbf{C}^*_j|\widehat{\mathrm{MB}}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)| \leq \frac{\varepsilon}{2^{k+2}}$ introduces at most an additional error of $\frac{\varepsilon}{2^{k+2}}$. Thus after the clipping we have,

$$|\log f^*_{\mathbf{C}^*_j|x-\mathbf{C}^*_j}(\mathbf{c}^*_j) - \log \hat{f}^*_{\mathbf{C}^*_j|\widehat{\mathrm{MB}}(\mathbf{C}^*_j)}(\mathbf{c}^*_j)| \leq \frac{\varepsilon}{2^{k+1}}, \tag{69}$$

for all canonical factors of P. We also have that all candidate factors that are not present in the canonical form of the true distribution P will now have been removed and do not contribute to $\tilde{P}$. (By our assumption on b the algorithm considered large enough Markov blanket candidates to include the true Markov blanket. Such a large enough b for these factors (which can be larger than the maximum Markov blanket size for factors present in the distribution) is important. Trivial (all-ones) canonical factors computed using their Markov blanket require a true Markov blanket to be all-ones.)

So far we have shown that Eqn. (69) holds with probability $1-|\mathcal{C}||\mathcal{Y}|\delta'-|\mathcal{C}|\delta''$ if m satisfies both Eqn. (63) and Eqn. (68). Or, after substituting in the choice of ε', if the following hold

$$\begin{aligned} m &\geq \frac{2^{8k+19}\nu^{2k+2b}}{\gamma^{6k+6b}\varepsilon^4}\log\frac{4\nu^{k+b}}{\delta'}, \\ m &\geq 2^{4k+7}\frac{(1+2\frac{\varepsilon\gamma^{k+b}}{2^{2k+4}})^2}{\gamma^{4k+4b}\varepsilon^2}\log\frac{4\nu^{k+b}}{\delta''}. \end{aligned}$$

So choosing $\delta' = \delta'' = \delta/(2|\mathcal{C}||\mathcal{Y}|)$, we have that for Eqn. (69) to hold with probability $1-\delta$, it suffices that

$$m \geq (1+\frac{\varepsilon\gamma^{k+b}}{2^{2k+3}})^2\frac{\nu^{2k+2b}2^{8k+19}}{\gamma^{6k+6b}\min\{\varepsilon^2,\varepsilon^4\}}\log\frac{8|\mathcal{C}||\mathcal{Y}|\nu^{k+b}}{\delta}.$$

Now using the fact that $|\mathcal{C}| \leq kn^k$ and $|\mathcal{Y}| \leq bn^b$ we obtain the following result: with probability $1-\delta$, Eqn. (69) holds for all non-trivial canonical factors in the target distribution if m satisfies the condition on m in the theorem, namely Eqn. (19). Moreover (recall the clipping procedure removed all candidate factors with scope less than k and Markov blanket size less than b that are not present in the canonical form of the true distribution P) we have that zero error is incurred on all other factors. Thus (after using Lemma 20) we have that

$$D(P\|\hat{P}) + D(\hat{P}\|P) \leq 2J^*\frac{\varepsilon}{2^{k+1}} \leq J\varepsilon.$$

The second inequality follows since $J^* \leq 2^k J$.

■

F.5 Proof of Theorem 16

Proof [Theorem 16] From Proposition 4 we have that

$$Q(\mathbf{x}) = \frac{1}{Z}\prod_j f^*_{\mathbf{D}^*_j|\mathrm{MB}(\mathbf{D}^*_j)}(\mathbf{d}^*_j).$$

We can rewrite this product as follows:

$$Q(\mathbf{x}) = \frac{1}{Z} \prod_{j:|\mathbf{D}_j^*|\leq k,|\mathrm{MB}(\mathbf{D}_j^*)|\leq b} f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*) \prod_{j:|\mathbf{D}_j^*|\leq k,|\mathrm{MB}(\mathbf{D}_j^*)|>b} f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*) \prod_{j:|\mathbf{D}_j^*|>k} f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*) \tag{70}$$

The learned distribution $\tilde{P} = \frac{1}{\tilde{Z}} \prod_{j=1}^{J^*} \hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)$, can be rewritten as

$$\tilde{P}(\mathbf{x}) = \frac{1}{\tilde{Z}} \prod_{j:|\mathbf{D}_j^*|\leq k,|\mathrm{MB}(\mathbf{D}_j^*)|\leq b} \hat{f}^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*) \prod_{j:|\mathbf{D}_j^*|\leq k,|\mathrm{MB}(\mathbf{D}_j^*)|>b} \hat{f}^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*) \prod_{j:\mathbf{C}_j^*\notin\{\mathbf{D}_k^*\}_k,|\mathrm{MB}(\mathbf{C}_j^*)|\leq b} \hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*) \prod_{j:\mathbf{C}_j^*\notin\{\mathbf{D}_k^*\}_k,|\mathrm{MB}(\mathbf{C}_j^*)|>b} \hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*). \tag{71}$$

Using Eqn. (70), Eqn. (71), the fact that for all $\mathbf{C}_j^* \notin \{\mathbf{D}_k^*\}_k$ we have that the canonical factor is trivial, namely $\log f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*) = 0$ (and adding and subtracting the same terms) we get: $D(Q\|\tilde{P}) + D(\tilde{P}\|Q) =$

$$E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ |\mathbf{D}_j^*|\leq k\\ |\mathrm{MB}(\mathbf{D}_j^*)|\leq b}} \log\frac{f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{\hat{f}^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) - E_{\mathbf{X}\sim\tilde{P}}\Big(\sum_{\substack{j:\ |\mathbf{D}_j^*|\leq k\\ |\mathrm{MB}(\mathbf{D}_j^*)|\leq b}} \log\frac{f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{\hat{f}^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) \tag{72}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ |\mathbf{D}_j^*|\leq k\\ |\mathrm{MB}(\mathbf{D}_j^*)|> b}} \log\frac{f^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{\hat{f}^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) - E_{\mathbf{X}\sim\tilde{P}}\Big(\sum_{\substack{j:\ |\mathbf{D}_j^*|\leq k\\ |\mathrm{MB}(\mathbf{D}_j^*)|> b}} \log\frac{f^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{\hat{f}^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) \tag{73}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ |\mathbf{D}_j^*|\leq k\\ |\mathrm{MB}(\mathbf{D}_j^*)|> b}} \log\frac{f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{f^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) - E_{\mathbf{X}\sim\tilde{P}}\Big(\sum_{\substack{j:\ |\mathbf{D}_j^*|\leq k\\ |\mathrm{MB}(\mathbf{D}_j^*)|> b}} \log\frac{f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}{f^*_{\mathbf{D}_j^*|\widehat{\mathrm{MB}}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)}\Big) \tag{74}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{j:|\mathbf{D}_j^*|>k} \log f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)\Big) - E_{\mathbf{X}\sim\tilde{P}}\Big(\sum_{j:|\mathbf{D}_j^*|>k} \log f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j^*)\Big) \tag{75}$$

$$-E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{C}_j^*\notin\{\mathbf{D}_k^*\}_k\\ |\mathrm{MB}(\mathbf{C}_j^*)|\leq b}} \log\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)\Big) + E_{\mathbf{X}\sim P}\Big(\sum_{\substack{j:\ \mathbf{C}_j^*\notin\{\mathbf{D}_k^*\}_k\\ |\mathrm{MB}(\mathbf{C}_j^*)|\leq b}} \log\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)\Big) \tag{76}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{C}_j^*\notin\{\mathbf{D}_k^*\}_k\\ |\mathrm{MB}(\mathbf{C}_j^*)|> b}} \log\frac{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}\Big) - E_{\mathbf{X}\sim P}\Big(\sum_{\substack{j:\ \mathbf{C}_j^*\notin\{\mathbf{D}_k^*\}_k\\ |\mathrm{MB}(\mathbf{C}_j^*)|> b}} \log\frac{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{\hat{f}^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}\Big) \tag{77}$$

$$+E_{\mathbf{X}\sim Q}\Big(\sum_{\substack{j:\ \mathbf{C}_j^*\notin\{\mathbf{D}_k^*\}_k\\ |\mathrm{MB}(\mathbf{C}_j^*)|> b}} \log\frac{f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}\Big) - E_{\mathbf{X}\sim P}\Big(\sum_{\substack{j:\ \mathbf{C}_j^*\notin\{\mathbf{D}_k^*\}_k\\ |\mathrm{MB}(\mathbf{C}_j^*)|> b}} \log\frac{f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{f^*_{\mathbf{C}_j^*|\widehat{\mathrm{MB}}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}\Big) \tag{78}$$

$$+\log\frac{Z}{\tilde{Z}} + \log\frac{\tilde{Z}}{Z}. \tag{79}$$

Using the same reasoning as in the proof of Theorem 15 we obtain that for the sum of the terms in lines (72), (73), (76) and (77) to be bounded by $(J+|S|)\varepsilon$ with probability at least $1-\delta$, it suffices that m satisfies Eqn. (19). The additional term in the bound, namely $|S|\varepsilon$, is necessary to bound the error contribution of the terms in line (77).

The sum of the terms in lines (74) and (78) can be bounded by

$$2 \sum_{\mathbf{C}_j^* \in \mathcal{C} \,:\, |\mathrm{MB}(\mathbf{C}_j^*)|>b} \max_{\mathbf{c}_j^*} \Big| \log \frac{f^*_{\mathbf{C}_j^*|\mathrm{MB}(\mathbf{C}_j^*)}(\mathbf{c}_j^*)}{f^*_{\mathbf{C}_j^*|\widehat{MB}(\mathbf{C}_j^*)}(\mathbf{C}_j^*)} \Big|$$

The sum of the terms in line (75) can be bounded by (recall $\mathrm{MB}(\cdot)$ is the true Markov blanket for the true distribution Q, thus $f^*_{\mathbf{D}_j^*}(\mathbf{d}_j) = f^*_{\mathbf{D}_j^*|\mathrm{MB}(\mathbf{D}_j^*)}(\mathbf{d}_j)$)

$$2 \sum_{j:|\mathbf{D}_j^*|>k} \max_{\mathbf{d}_j^*} \Big| \log f^*_{\mathbf{D}_j^*}(\mathbf{d}_j) \Big|.$$

The two terms in line (79) sum to zero.

This establishes the theorem. ■

References

P. Abbeel, D. Koller, and A. Y. Ng. Learning factor graphs in polynomial time & sample complexity. In *Proc. UAI*, 2005.

N. Abe, J. Takeuchi, and M. Warmuth. Polynomial learnability of probabilistic concepts with respect to the Kullback-Leibler divergence. In *Proc. COLT*, 1991.

N. Abe, J. Takeuchi, and M. Warmuth. On the computational complexity of approximating probability distributions by probabilistic automata. *Machine Learning*, 1992.

F. Bach and M. Jordan. Thin junction trees. In *NIPS 14*, 2002.

F. Barahona. On the computational complexity of Ising spin glass models. *J. Phys. A*, 1982.

J. Besag. Efficiency of pseudo-likelihood estimation for simple Gaussian fields. *Biometrika*, 1974a.

J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 1974b.

J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence Journal*, 2002.

D. M. Chickering. Learning Bayesian networks is NP-Complete. In D. Fisher and H.J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

D. M. Chickering and C. Meek. Finding optimal Bayesian networks. In *Proc. UAI*, 2002.

D. M. Chickering, C. Meek, and D. Heckerman. Large-sample learning of Bayesian networks is hard. In *Proc. UAI*, 2003.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 1968.

F. Comets. On consistency of a class of estimators for exponential families of Markov random fields on the lattice. *Annals of Statistics*, 1992.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

S. Dasgupta. The sample complexity of learning fixed structure Bayesian networks. *Machine Learning*, 1997.

S. Dasgupta. Learning polytrees. In *Proc. UAI*, 1999.

S. Della Pietra, V. J. Della Pietra, and J. D. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

A. Deshpande, M. N. Garofalakis, and M. I. Jordan. Efficient stepwise selection in decomposable models. In *Proc. UAI*, pages 128–135. Morgan Kaufmann Publishers Inc., 2001. ISBN 1-55860-800-1.

N. Friedman and Z. Yakhini. On the sample complexity of learning Bayesian networks. In *Proc. UAI*, 1996.

S. Geman and C. Graffigne. Markov random field image models and their applications to computer vision. In *Proc. of the International Congress of Mathematicians*, 1986.

C. J. Geyer and E. A. Thompson. Constrained Monte Carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society, Series B*, 1992.

B. Gidas. Consistency of maximum likelihood and pseudo-likelihood estimators for Gibbsian distributions. In W. Fleming and P.-L. Lions, editors, *Stochastic differential systems, stochastic control theory and applications*. Springer, New York, 1988.

X. Guyon and H. R. Künsch. Asymptotic comparison of estimators in the Ising model. In *Stochastic Models, Statistical Methods, and Algorithms in Image Analysis, Lecture Notes in Statistics*. Springer, Berlin, 1992.

J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished, 1971.

K. L. Höffgen. Learning and robust learning of product distributions. In *Proc. COLT*, 1993.

F. Huang and Y. Ogata. Generalized pseudo-likelihood estimates for Markov random fields on lattice. *Annals of the Institute of Statistical Mathematics*, 2002.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 1993.

C. Ji and L. Seymour. A consistent model selection procedure for Markov random fields based on penalized pseudolikelihood. *Annals of Applied Probability*, 1996.

D. Karger and N. Srebro. Learning Markov networks: maximum bounded tree-width graphs. In *Symposium on Discrete Algorithms*, pages 392–401, 2001.

F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 2001.

S. Kullback. *Probability theory and statistics*. Wiley, 1959.

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

F. M. Malvestuto. Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man and Cybernetics*, 1991.

A. McCallum. Efficiently inducing features of conditional random fields. In *Proc. UAI*, 2003.

C. Meek. Finding a path is harder than finding a tree. *Journal of Artificial Intelligence Research*, 15:383–389, 2001.

M. Narasimhan and J. Bilmes. PAC-learning bounded tree-width graphical models. In *Proc. UAI*, 2004.

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS 14*, 2002.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search (second edition)*. MIT Press, 2000.

N. Srebro. Maximum likelihood bounded tree-width Markov networks. In *Proc. UAI*, 2001.

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

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. Technical report, Mitsubishi Electric Reseach Laboratories, 2001.

Journal of Machine Learning Research 7 (2006) 1789–1828 Submitted 12/05; Published 9/06

Collaborative Multiagent Reinforcement Learning by Payoff Propagation

Jelle R. Kok JELLEKOK@SCIENCE.UVA.NL
Nikos Vlassis VLASSIS@SCIENCE.UVA.NL
Informatics Institute
University of Amsterdam
Kruislaan 403
1098SJ, Amsterdam, The Netherlands

Editor: Michael Littman

Abstract

In this article we describe a set of scalable techniques for learning the behavior of a group of agents in a collaborative multiagent setting. As a basis we use the framework of coordination graphs of Guestrin, Koller, and Parr (2002a) which exploits the dependencies between agents to decompose the global payoff function into a sum of local terms. First, we deal with the single-state case and describe a payoff propagation algorithm that computes the individual actions that approximately maximize the global payoff function. The method can be viewed as the decision-making analogue of belief propagation in Bayesian networks. Second, we focus on learning the behavior of the agents in sequential decision-making tasks. We introduce different model-free reinforcement-learning techniques, unitedly called Sparse Cooperative *Q*-learning, which approximate the global action-value function based on the topology of a coordination graph, and perform updates using the contribution of the individual agents to the maximal global action value. The combined use of an edge-based decomposition of the action-value function and the payoff propagation algorithm for efficient action selection, result in an approach that scales only linearly in the problem size. We provide experimental evidence that our method outperforms related multiagent reinforcement-learning methods based on temporal differences.

Keywords: collaborative multiagent system, coordination graph, reinforcement learning, *Q*-learning, belief propagation

1. Introduction

A multiagent system (MAS) consists of a group of agents that reside in an environment and can potentially interact with each other (Sycara, 1998; Weiss, 1999; Durfee, 2001; Vlassis, 2003). The existence of multiple operating agents makes it possible to solve inherently distributed problems, but also allows one to decompose large problems, which are too complex or too expensive to be solved by a single agent, into smaller subproblems.

In this article we are interested in collaborative multiagent systems in which the agents have to work together in order to optimize a shared performance measure. In particular, we investigate sequential decision-making problems in which the agents repeatedly interact with their environment and try to optimize the long-term reward they receive from the system, which depends on a sequence of joint decisions. Specifically, we focus on *inherently cooperative* tasks involving a large

group of agents in which the success of the team is measured by the specific combination of actions of the agents (Parker, 2002). This is different from other approaches that assume implicit coordination through either the observed state variables (Tan, 1993; Dutta et al., 2005), or reward structure (Becker et al., 2003). We concentrate on model-free learning techniques in which the agents do not have access to the transition or reward model. Example application domains include network routing (Boyan and Littman, 1994; Dutta et al., 2005), sensor networks (Lesser et al., 2003; Modi et al., 2005), but also robotic teams, for example, exploration and mapping (Burgard et al., 2000), motion coordination (Arai et al., 2002) and RoboCup (Kitano et al., 1995; Kok et al., 2005).

Existing learning techniques have been proved successful in learning the behavior of a single agent in stochastic environments (Tesauro, 1995; Crites and Barto, 1996; Ng et al., 2004). However, the presence of multiple learning agents in the same environment complicates matters. First of all, the action space scales exponentially with the number of agents. This makes it infeasible to apply standard single-agent techniques in which an action value, representing expected future reward, is stored for every possible state-action combination. An alternative approach would be to decompose the action value among the different agents and update them independently. However, the fact that the behavior of one agent now influences the outcome of the individually selected actions of the other agents results in a dynamic environment and possibly compromises convergence. Other difficulties, which are outside the focus of this article, appear when the different agents receive incomplete and noisy observations of the state space (Goldman and Zilberstein, 2004), or have a restricted communication bandwidth (Pynadath and Tambe, 2002; Goldman and Zilberstein, 2003).

For our model representation we will use the collaborative multiagent Markov decision process (collaborative multiagent MDP) model (Guestrin, 2003). In this model each agent selects an individual action in a particular state. Based on the resulting joint action the system transitions to a new state and the agents receive an *individual* reward. The global reward is the sum of all individual rewards. This approach differs from other multiagent models, for example, multiagent MDPs (Boutilier, 1996) or decentralized MDPs (Bernstein et al., 2000), in which all agents observe the global reward. In a collaborative MDP, it is still the goal of the agents to optimize the global reward, but the individually received rewards allow for solution techniques that take advantage of the problem structure.

One such solution technique is based on the framework of coordination graphs (CGs) (Guestrin et al., 2002a). This framework exploits that in many problems only a few agents depend on each other and decomposes a coordination problem into a combination of simpler problems. In a CG each node represents an agent and connected agents indicate a local coordination dependency. Each dependency corresponds to a local payoff function which assigns a specific value to every possible action combination of the involved agents. The global payoff function equals the sum of all local payoff functions. To compute the joint action that maximizes the global payoff function, a variable elimination (VE) algorithm can be used (Guestrin et al., 2002a). This algorithm operates by eliminating the agents one by one after performing a local maximization step, and has exponential complexity in the induced tree width (the size of the largest clique generated during the node elimination).

In this article we investigate different distributed learning methods to coordinate the behavior between the agents. The algorithms are distributed in the sense that each agent only needs to communicate with the neighboring agents on which it depends. Our contribution is two-fold. First, we describe a 'payoff propagation' algorithm (max-plus) (Vlassis et al., 2004; Kok and Vlassis, 2005) to find an approximately maximizing joint action for a CG in which all local functions are speci-

fied beforehand. Our algorithm exploits the fact that there is a direct duality between computing the maximum a posteriori configuration in a probabilistic graphical model and finding the optimal joint action in a CG; in both cases we are optimizing over a function that is decomposed in local terms. This allows message-passing algorithms that have been developed for inference in probabilistic graphical models to be directly applicable for action selection in CGs. Max-plus is a popular method of that family. In the context of CG, it can therefore be regarded as an approximate alternative to the exact VE algorithm for multiagent decision making. We experimentally demonstrate that this method, contrary to VE, scales to large groups of agents with many dependencies.

The problem of finding the maximizing joint action in a fixed CG is also related to the work on distributed constraint satisfaction problems (CSPs) in constraint networks (Pearl, 1988). These problems consist of a set of variables which each take a value from a finite, discrete domain. Predefined constraints, which have the values of a subset of all variables as input, specify a cost. The objective is to assign values to these variables such that the total cost is minimized (Yokoo and Durfee, 1991; Dechter, 2003).

As a second contribution, we study sequential decision-making problems in which we learn the behavior of the agents. For this, we apply model-free reinforcement-learning techniques (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998). This problem is different than finding the joint action that maximizes predefined payoff relations, since in this case the payoff relations themselves have to be learned. In our approach, named *Sparse Cooperative Q-learning* (Kok and Vlassis, 2004), we analyze different decompositions of the global action-value function using CGs. The structure of the used CG is determined beforehand, and reflects the specific problem under study. For a given CG, we investigate both a decomposition in terms of the nodes (or agents), as well as a decomposition in terms of the edges. In the agent-based decomposition the local function of an agent is based on its own action and those of its neighboring agents. In the edge-based decomposition each local function is based on the actions of the two agents forming this edge. Each state is related to a CG with a similar decomposition, but with different values for the local functions. To update the local action-value function for a specific state, we use the contribution of the involved agents to the maximal global action value, which is computed using either the max-plus or VE algorithm. We perform different experiments on problems involving a large group of agents with many dependencies and show that all variants outperform existing temporal-difference based learning techniques in terms of the quality of the extracted policy. Note that in our work we only consider temporal-difference methods; other multiagent reinforcement-learning methods exist that are based, for example, on policy search (Peshkin et al., 2000; Moallemi and Van Roy, 2004) or Bayesian approaches (Chalkiadakis and Boutilier, 2003).

The remainder of this article is structured as follows. We first review the notion of a CG and the VE algorithm in Section 2. Next, in Section 3, we discuss our approximate alternative to VE based on the max-plus algorithm and perform experiments on randomly generated graphs. Then, we switch to sequential decision-making problems. First, we review several existing multiagent learning methods in Section 4. In Section 5, we introduce the different variants of our Sparse Cooperative Q-learning method, and give experimental results on several learning problems in Section 6. We end with the conclusions in Section 7.

2. Coordination Graphs and Variable Elimination

All agents in a collaborative multiagent system can potentially influence each other. It is therefore important to ensure that the actions selected by the individual agents result in optimal decisions for the group as a whole. This is often referred to as the *coordination problem*. In this section we review the problem of computing a coordinated action for a group of n agents as described by Guestrin et al. (2002a). Each agent i selects an individual action a_i from its action set $\mathcal{A}_i$ and the resulting *joint* action $\mathbf{a} = (a_1, \ldots, a_n)$, as all other vectors of two or more variables in this article emphasized using a bold notation, generates a payoff $u(\mathbf{a})$ for the team. The coordination problem is to find the optimal joint action $\mathbf{a}^*$ that maximizes $u(\mathbf{a})$, that is, $\mathbf{a}^* = \arg\max_{\mathbf{a}} u(\mathbf{a})$.

We can compute the optimal joint action by enumerating over all possible joint actions and select the one that maximizes $u(\mathbf{a})$. However, this approach quickly becomes impractical, as the size of the joint action space $|\mathcal{A}_1 \times \ldots \times \mathcal{A}_n|$ grows exponentially with the number of agents n. Fortunately, in many problems the action of one agent does not depend on the actions of all other agents, but only on a small subset. For example, in many real-world domains only agents which are spatially close have to coordinate their actions.

The framework of coordination graphs (CGs) (Guestrin et al., 2002a) is a recent approach to exploit these dependencies. This framework assumes the action of an agent i only depends on a subset of the other agents, $j \in \Gamma(i)$. The global payoff function $u(\mathbf{a})$ is then decomposed into a linear combination of local payoff functions, as follows,

$$u(\mathbf{a}) = \sum_{i=1}^{n} f_i(\mathbf{a}_i). \tag{1}$$

Each local payoff function f_i depends on a subset of all actions, $\mathbf{a}_i \subseteq \mathbf{a}$, where $\mathbf{a}_i = \mathcal{A}_i \times (\times_{j \in \Gamma(i)} \mathcal{A}_j)$, corresponding to the action of agent i and those of the agents on which it depends. This decomposition can be depicted using an undirected graph $G = (V, E)$ in which each node $i \in V$ represents an agent and an edge $(i, j) \in E$ indicates that the corresponding agents have to coordinate their actions, that is, $i \in \Gamma(j)$ and $j \in \Gamma(i)$. The global coordination problem is now replaced by a number of local coordination problems each involving fewer agents.

In the remainder of this article, we will focus on problems with payoff functions including at most two agents. Note that this still allows for complicated coordinated structures since every agent can have multiple pairwise dependency functions. Furthermore, it is possible to generalize the proposed techniques to payoff functions with more than two agents because any arbitrary graph can be converted to a graph with only pairwise inter-agent dependencies (Yedidia et al., 2003; Loeliger, 2004). To accomplish this, a new agent is added for each local function that involves more than two agents. This new agent contains an individual local payoff function that is defined over the combined actions of the involved agents, and returns the corresponding value of the original function. Note that the action space of this newly added agent is exponential in its neighborhood size (which can lead to intractability in the worst case). Furthermore, new pairwise payoff functions have to be defined between each involved agent and the new agent in order to ensure that the action selected by the involved agent corresponds to its part of the (combined) action selected by the new agent.

Allowing only payoff functions defined over at most two agents, the global payoff function $u(\mathbf{a})$ can be decomposed as

$$u(\mathbf{a}) = \sum_{i \in V} f_i(a_i) + \sum_{(i,j) \in E} f_{ij}(a_i, a_j). \tag{2}$$

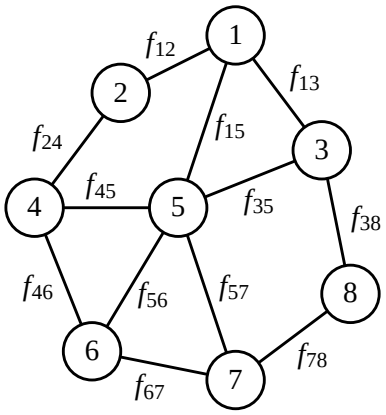
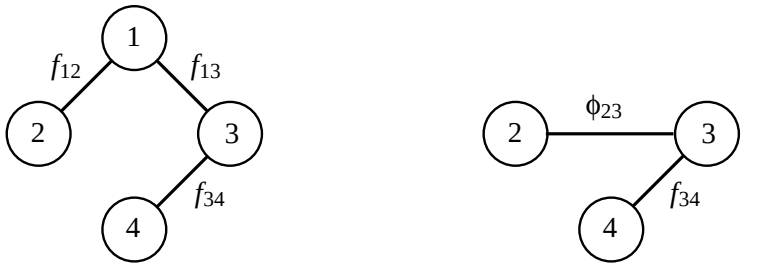


Figure 1: Example CG with eight agents; an edge represents a coordination dependency.



(a) Initial graph. (b) After elimination of agent 1.

Figure 2: CG corresponding to the decomposition (3) before and after eliminating agent 1.

A local payoff function $f_i(a_i)$ specifies the payoff contribution for the individual action a_i of agent i, and f_{ij} defines the payoff contribution for pairs of actions (a_i, a_j) of neighboring agents $(i, j) \in E$. Fig. 1 shows an example CG with 8 agents.

In order to solve the coordination problem and find $\mathbf{a}^* = \arg\max_{\mathbf{a}} u(\mathbf{a})$ we can apply the variable elimination (VE) algorithm (Guestrin et al., 2002a), which is in essence identical to variable elimination in a Bayesian network (Zhang and Poole, 1996). The algorithm eliminates the agents one by one. Before an agent (node) is eliminated, the agent first collects all payoff functions related to its edges. Next, it computes a conditional payoff function which returns the maximal value it is able to contribute to the system for every action combination of its neighbors, and a best-response function (or conditional strategy) which returns the action corresponding to the maximizing value. The conditional payoff function is communicated to one of its neighbors and the agent is eliminated from the graph. Note that when the neighboring agent receives a function including an action of an agent on which it did not depend before, a new coordination dependency is added between these agents. The agents are iteratively eliminated until one agent remains. This agent selects the action that maximizes the final conditional payoff function. This individual action is part of the optimal joint action and the corresponding value equals the desired value $\max_{\mathbf{a}} u(\mathbf{a})$. A second pass in the reverse order is then performed in which every agent computes its optimal action based on its conditional strategy and the fixed actions of its neighbors.

We illustrate VE on the decomposition graphically represented in Fig. 2(a), that is,

$$u(\mathbf{a}) = f_{12}(a_1, a_2) + f_{13}(a_1, a_3) + f_{34}(a_3, a_4), \tag{3}$$

We first eliminate agent 1. This agent does not depend on the local payoff function f_{34} and therefore the maximization of $u(\mathbf{a})$ in (3) can be written as

$$\max_{\mathbf{a}} u(\mathbf{a}) = \max_{a_2,a_3,a_4} \Big\{ f_{34}(a_3, a_4) + \max_{a_1} [f_{12}(a_1, a_2) + f_{13}(a_1, a_3)] \Big\}. \tag{4}$$

Agent 1 computes a conditional payoff function $\phi_{23}(a_2, a_3) = \max_{a_1}[f_{12}(a_1, a_2) + f_{13}(a_1, a_3)]$ and the best-response function $B_1(a_2, a_3) = \arg\max_{a_1}[f_{12}(a_1, a_2) + f_{13}(a_1, a_3)]$ which respectively return the maximal value and the associated best action agent 1 is able to perform given the actions of agent 2 and 3. Since the function $\phi_{23}(a_2, a_3)$ is independent of agent 1, it is now eliminated from the graph, simplifying (4) to $\max_{\mathbf{a}} u(\mathbf{a}) = \max_{a_2,a_3,a_4}[f_{34}(a_3, a_4) + \phi_{23}(a_2, a_3)]$. The elimination of agent 1 induces a new dependency between agent 2 and 3 and thus a change in the graph's topology. This is depicted in Fig. 2(b). We then eliminate agent 2. Only ϕ_{23} depends on agent 2, so we define $B_2(a_3) = \arg\max_{a_2} \phi_{23}(a_2, a_3)$ and replace ϕ_{23} by $\phi_3(a_3) = \max_{a_2} \phi_{23}(a_2, a_3)$ producing

$$\max_{\mathbf{a}} u(\mathbf{a}) = \max_{a_3,a_4} [f_{34}(a_3, a_4) + \phi_3(a_3)], \tag{5}$$

which is independent of a_2. Next, we eliminate agent 3 and replace the functions f_{34} and ϕ_3 resulting in $\max_{\mathbf{a}} u(\mathbf{a}) = \max_{a_4} \phi_4(a_4)$ with $\phi_4(a_4) = \max_{a_3}[f_{34}(a_3, a_4) + \phi_3(a_3)]$. Agent 4 is the last remaining agent and fixes its optimal action $a_4^* = \arg\max_{a_4} \phi_4(a_4)$. A second pass in the reverse elimination order is performed in which each agent computes its optimal (unconditional) action from its best-response function and the fixed actions from its neighbors. In our example, agent 3 first selects $a_3^* = B_3(a_4^*)$. Similarly, we get $a_2^* = B_2(a_3^*)$ and $a_1^* = B_1(a_2^*, a_3^*)$. When an agent has more than one maximizing best-response action, it selects one randomly, since it always communicates its choice to its neighbors. The described procedure holds for the case of a truly distributed implementation using communication. When communication is restricted, additional common knowledge assumptions are needed such that each agent is able to run a copy of the algorithm (Vlassis, 2003, ch. 4).

The VE algorithm always produces the optimal joint action and does not depend on the elimination order. The execution time of the algorithm, however, does. Computing the optimal order is known to be NP-complete, but good heuristics exist, for example, first eliminating the agent with the minimum number of neighbors (Bertelé and Brioschi, 1972). The execution time is exponential in the induced width of the graph (the size of the largest clique computed during node elimination). For densely connected graphs this can scale exponentially in n. Furthermore, VE will only produce its final result after the end of the second pass. This is not always appropriate for real-time multiagent systems where decision making must be done under time constraints. In these cases, an anytime algorithm that improves the quality of the solution over time is more appropriate (Vlassis et al., 2004).

3. Payoff Propagation and the Max-Plus Algorithm[1]

Although the variable elimination (VE) algorithm is exact, it does not scale well with densely connected graphs. In this section, we introduce the *max-plus algorithm* as an approximate alternative to VE and compare the two approaches on randomly generated graphs.

[1] Section 3 is largely based on (Kok and Vlassis, 2005).

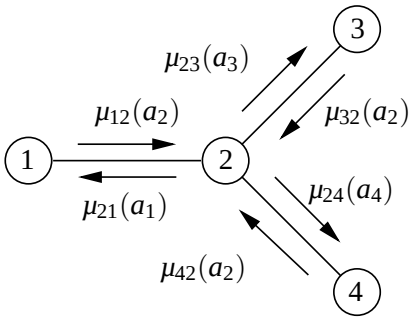


Figure 3: Graphical representation of different messages μ_{ij} in a graph with four agents.

3.1 The Max-Plus Algorithm

The max-product algorithm (Pearl, 1988; Yedidia et al., 2003; Wainwright et al., 2004) is a popular method for computing the *maximum a posteriori* (MAP) configuration in an (unnormalized) undirected graphical model. This method is analogous to the belief propagation or sum-product algorithm (Kschischang et al., 2001). It operates by iteratively sending locally optimized messages $\mu_{ij}(a_j)$ between node i and j over the corresponding edge in the graph. For tree-structured graphs, the message updates converge to a fixed point after a finite number of iterations (Pearl, 1988). After convergence, each node then computes the MAP assignment based on its local incoming messages only.

There is a direct duality between computing the MAP configuration in a probabilistic graphical model and finding the optimal joint action in a CG; in both cases we are optimizing over a function that is decomposed in local terms. This allows message-passing algorithms that have been developed for inference in probabilistic graphical models, to be directly applicable for action selection in CGs. Max-plus is a popular method of that family. In the context of CG, it can therefore be regarded as a 'payoff propagation' technique for multiagent decision making.

Suppose that we have a coordination graph $G = (V,E)$ with $|V|$ vertices and $|E|$ edges. In order to compute the optimal joint action $\mathbf{a}^*$ that maximizes (2), each agent i (node in G) repeatedly sends a message μ_{ij} to its neighbors $j \in \Gamma(i)$. The message μ_{ij} can be regarded as a local payoff function of agent j and is defined as

$$\mu_{ij}(a_j) = \max_{a_i} \left\{ f_i(a_i) + f_{ij}(a_i,a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) \right\} + c_{ij}, \tag{6}$$

where $\Gamma(i) \setminus j$ represents all neighbors of agent i except agent j, and c_{ij} is a normalization value (which can be assumed zero for now). This message is an approximation of the maximum payoff agent i is able to achieve for a given action of agent j, and is computed by maximizing (over the actions of agent i) the sum of the payoff functions f_i and f_{ij} and all incoming messages to agent i except that from agent j. Note that this message only depends on the payoff relations between agent i and agent j and the incoming message to agent i. Messages are exchanged until they converge to a fixed point, or until some external signal is received. Fig. 3 shows a CG with four agents and the corresponding messages.

A message μ_{ij} in the max-plus algorithm has three important differences with respect to the conditional payoff functions in VE. First, before convergence each message is an approximation

of the exact value (conditional team payoff) since it depends on the incoming (still not converged) messages. Second, an agent i only has to sum over the received messages from its neighbors which are defined over individual actions, instead of enumerating over all possible action combinations of its neighbors. This is the main reason for the scalability of the algorithm. Finally, in the max-plus algorithm, messages are always sent over the edges of the original graph. In the VE algorithm, the elimination of an agent often results in new dependencies between agents that did not have to coordinate initially.

For trees the messages converge to a fixed point within a finite number of steps (Pearl, 1988; Wainwright et al., 2004). Since a message $\mu_{ji}(a_i)$ equals the payoff produced by the subtree with agent j as root when agent i performs action a_i, we can at any time step define

$$g_i(a_i) = f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i), \tag{7}$$

which equals the contribution of the individual function of agent i and the different subtrees with the neighbors of agent i as root. Using (7), we can show that, at convergence, $g_i(a_i) = \max_{\{\mathbf{a}' | a'_i = a_i\}} u(\mathbf{a}')$ holds. Each agent i can then individually select its optimal action

$$a_i^* = \arg\max_{a_i} g_i(a_i). \tag{8}$$

If there is only one maximizing action for every agent i, the globally optimal joint action $\mathbf{a}^* = \arg\max_{\mathbf{a}} u(\mathbf{a})$ is unique and has elements $\mathbf{a}^* = (a_i^*)$. Note that this optimal joint action is computed by only local optimizations (each node maximizes $g_i(a_i)$ separately). In case the local maximizers are not unique, an optimal joint action can be computed by a dynamic programming technique (Wainwright et al., 2004, sec. 3.1). In this case, each agent informs its neighbors in a predefined order about its action choice such that the other agents are able to fix their actions accordingly.

```
centralized max-plus algorithm for CG = (V,E)
initialize μ_ij = μ_ji = 0 for (i,j) ∈ E, g_i = 0 for i ∈ V and m = −∞
while fixed_point = false and deadline to send action has not yet arrived do
  // run one iteration
  fixed_point = true
  for every agent i do
    for all neighbors j = Γ(i) do
      send j message μ_ij(a_j) = max_{a_i} { f_i(a_i) + f_ij(a_i,a_j) + Σ_{k∈Γ(i)\j} μ_ki(a_i) } + c_ij
      if μ_ij(a_j) differs from previous message by a small threshold then
        fixed_point = false
    determine g_i(a_i) = f_i(a_i) + Σ_{j∈Γ(i)} μ_ji(a_i) and a'_i = arg max_{a_i} g_i(a_i)
  if use anytime extension then
    if u((a'_i)) > m then
      (a*_i) = (a'_i) and m = u((a'_i))
  else
    (a*_i) = (a'_i)
return (a*_i)
```

Algorithm 1: Pseudo-code of the centralized max-plus algorithm.

Unfortunately there are no guarantees that max-plus converges in graphs with cycles and therefore no assurances can be given about the quality of the corresponding joint action $\mathbf{a}^* = (a_i^*)$ with a_i from (8) in such settings. Nevertheless, it has been shown that a fixed point of message passing exists (Wainwright et al., 2004), but there is no algorithm yet that provably converges to such a solution. However, bounds are available that characterize the quality of the solution if the algorithm converges (Wainwright et al., 2004). Regardless of these results, the algorithm has been successfully applied in practice in graphs with cycles (Murphy et al., 1999; Crick and Pfeffer, 2003; Yedidia et al., 2003). One of the main problems is that an outgoing message from agent i which is part of a cycle eventually becomes part of its incoming messages. As a result the values of the messages grow extremely large. Therefore, as in (Wainwright et al., 2004), we normalize each sent message by subtracting the average of all values in μ_{ik} using $c_{ij} = \frac{1}{|A_k|} \sum_k \mu_{ik}(a_k)$ in (6). Still, the joint action might change constantly when the messages keep fluctuating. This necessitates the development of an extension of the algorithm in which each (local) action is only updated when the corresponding global payoff improves. Therefore, we extend the max-plus algorithm by occasionally computing the global payoff and only update the joint action when it improves upon the best value found so far. The best joint action then equals the last updated joint action. We refer to this approach as the *anytime* max-plus algorithm. [2]

The max-plus algorithm can be implemented in either a centralized or a distributed version. The centralized version operates using iterations. In one iteration each agent i computes and sends a message μ_{ij} to all its neighbors $j \in \Gamma(i)$ in a predefined order. This process continues until all messages are converged, or a 'deadline' signal (either from an external source or from an internal timing signal) is received and the current joint action is reported. For the anytime extension, we insert the current computed joint action into (2) after every iteration and only update the joint action when it improves upon the best value found so far. A pseudo-code implementation of the centralized max-plus algorithm, including the anytime extension, is given in Alg. 1.

The same functionality can also be implemented using a distributed implementation. Now, each agent computes and communicates an updated message directly after it has received a new (and different) message from one of its neighbors. This results in a computational advantage over the sequential execution of the centralized algorithm since messages are now sent in parallel. We additionally assume that after a finite number of steps, the agents receive a 'deadline' signal after which they report their individual actions.

For the distributed case, the implementation of the anytime extension is much more complex since the agents do not have direct access to the actions of the other agents or the global payoff function (2). Therefore, the evaluation of the (distributed) joint action is only initiated by an agent when it believes it is worthwhile to do so, for example, after a big increase in the values of the received messages. This agent starts the propagation of an 'evaluation' message over a spanning tree ST. A spanning tree is a tree-structured subgraph of G that includes all nodes. This tree is fixed beforehand and is common knowledge among all agents. An agent receiving an evaluation message fixes its individual action until after the evaluation. When an agent is a leaf of ST it also computes its local contribution to the global payoff and sends this value to its parent in ST. Each parent accumulates all payoffs of its children and after adding its own contribution sends the result to its parent. Finally, when the root of ST has received all accumulated payoffs from its children, the sum of these payoffs (global payoff) is distributed to all nodes in ST. The agents only update their best

[2] An alternative, and perhaps more accurate, term is 'max-plus with memory'. However, we decided on the term 'anytime' for reasons of consistency with other publications (Kok and Vlassis, 2005; Kok, 2006).

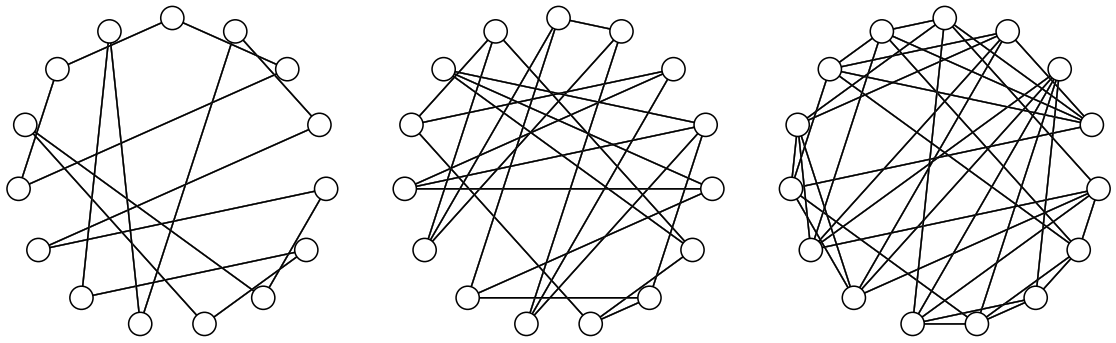
`distributed max-plus for agent` i, $CG = (V,E)$, `spanning tree` $ST = (V,S)$
initialize $\mu_{ij} = \mu_{ji} = 0$ for $j \in \Gamma(i)$, $g_i = 0$, $p_i = 0$ and $m = -\infty$
while deadline to send action has not yet arrived **do**
  wait for message *msg*
  if $msg = \mu_{ji}(a_i)$ // max-plus message **then**
    for all neighbors $j \in \Gamma(i)$ **do**
      compute $\mu_{ij}(a_j) = \max_{a_i} \left\{ f_i(a_i) + f_{ij}(a_i,a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) \right\} + c_{ij}$
      send message $\mu_{ij}(a_j)$ to agent j if it differs from last sent message
    if use anytime extension **then**
      if heuristic indicates global payoff should be evaluated **then**
        send `evaluate(` i `)` to agent i // initiate computation global payoff
    else
      $a_i^* = \arg\max_{a_i} [f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)]$
  if $msg =$ `evaluate(` j `)` // receive request for evaluation from agent j **then**
    if a_i' not locked, lock $a_i' = \arg\max_{a_i} [f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)]$ and set $p_i = 0$
    send `evaluate(` i `)` to all neighbors (parent and children) in $ST \neq j$
    if $i =$ leaf in ST **then**
      send `accumulate_payoff(` 0 `)` to agent i // initiate accumulation payoffs
  if $msg =$ `accumulate_payoff(` p_j `)` from agent j **then**
    $p_i = p_i + p_j$ // add payoff child j
    if received accumulated payoff from all children in ST **then**
      get actions a_j' from $j \in \Gamma(i)$ in CG and set $g_i = f_i(a_i') + \frac{1}{2} \sum_{j \in \Gamma(i)} f_{ij}(a_i', a_j')$
      if $i =$ root of ST **then**
        send `global_payoff(` $g_i + p_i$ `)` to agent i
      else
        send `accumulate_payoff(` $g_i + p_i$ `)` to parent in ST
  if $msg =$ `global_payoff(` g `)` **then**
    if $g > m$ **then**
      $a_i^* = a_i'$ and $m = g$
    send `global_payoff(` g `)` to all children in ST and unlock action a_i'
return a_i^*

Algorithm 2: Pseudo-code of a distributed max-plus implementation.

individual action a_i^* when this payoff improves upon the best one found so far. When the 'deadline' signal arrives, each agent reports the action related to the highest found global payoff, which might not correspond to the current messages. Alg. 2 shows a distributed version in pseudo-code.

3.2 Experiments

In this section, we describe our experiments with the max-plus algorithm on differently shaped graphs. For cycle-free graphs max-plus is equivalent to VE when the messages in the first iteration are sent in the same sequence as the elimination order of VE and in the reverse order for the second iteration (comparable to the reversed pass in VE). Therefore, we only test max-plus on graphs with cycles.



(a) Graph with 15 edges (average degree of 2).

(b) Graph with 23 edges (average degree of 3.07).

(c) Graph with 37 edges (average degree of 4.93).

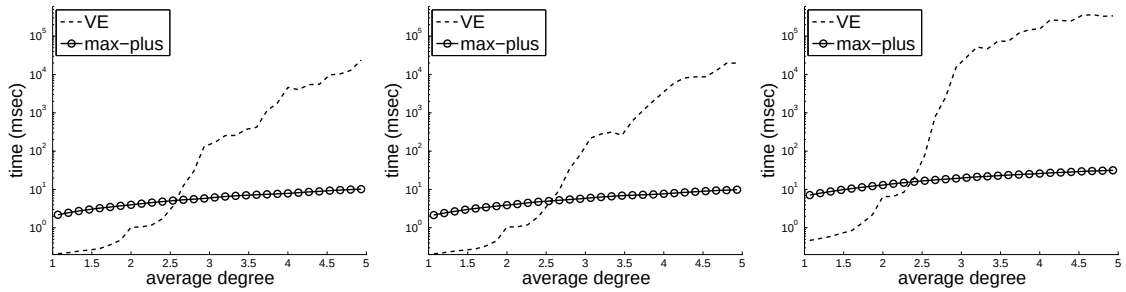
Figure 4: Example graphs with 15 agents and cycles.

We ran the algorithms on differently shaped graphs with 15 agents and a varying number of edges. In order to generate balanced graphs in which each agent approximately has the same degree, we start with a graph without edges and iteratively connect the two agents with the minimum number of neighbors. In case multiple agents satisfy this condition, an agent is picked at random from the possibilities. We apply this procedure to create 100 graphs for each $|E| \in \{8,9,\ldots,37\}$, resulting in a set of $3,000$ graphs. The set thus contains graphs in the range of on average 1.067 neighbors per agent (8 edges) to 4.93 neighbors per agent (37 edges). Fig. 10 depicts example graphs with respectively 15, 23 and 37 edges (on average 2, 3.07 and 4.93 neighbors per node). We create three copies of this set, each having a different payoff function related to the edges in the graph. In the first set, each edge $(i, j) \in E$ is associated with a payoff function f_{ij} defined over five actions per agent and each action combination is assigned a random payoff from a standard normal distribution, that is, $f_{ij}(a_i, a_j) \sim \mathcal{N}(0,1)$. This results in a total of 5^{15}, around 3 billion, different possible joint actions. In the second set, we add one outlier to each of the local payoff functions: for a randomly picked joint action, the corresponding payoff value is set to $10 \cdot \mathcal{N}(0,1)$. For the third test set, we specify a payoff function based on 10 actions per agent resulting in 10^{15} different joint actions. The values of the different payoff functions are again generated using a standard normal distribution.

For all graphs we compute the joint action using the VE algorithm, the standard max-plus algorithm, and the max-plus algorithm with the anytime extension. Irrespectively of convergence, all max-plus methods perform 100 iterations. As we will see later in Fig. 6 the policy has stabilized at this point. Furthermore, a random ordering is used in each iteration to determine which agents sends its messages.

The timing results for the three different test sets are plotted in Fig. 5.[3] The *x*-axis shows the average degree of the graph, and the *y*-axis shows, using a logarithmic scale, the average timing results, in milliseconds, to compute the joint action for the corresponding graphs. Remember from Section 2 that the computation time of the VE algorithm depends on the induced width of the graph. The induced width depends both on the average degree and the actual structure of the graph. The latter is generated at random, and therefore the complexity of graphs with the same average degree differ. Table 1 shows the induced width for the graphs used in the experiments based on the elimination order of the VE algorithm, that is, iteratively remove a node with the minimum number

[3] All results are generated on an Intel Xeon 3.4GHz / 2GB machine using a C++ implementation.



(a) 5 actions per agent. (b) 5 actions and outliers. (c) 10 actions per agent.

Figure 5: Timing results for VE and max-plus for different graphs with 15 agents and cycles.

average degree	$(1,2]$	$(2,3]$	$(3,4]$	$(4,5]$
induced width	1.23 ($\pm$0.44)	2.99 ($\pm$0.81)	4.94 ($\pm$0.77)	6.37 ($\pm$0.68)

Table 1: Average induced width and corresponding standard deviation for graphs with an average degree in $(x-1,x]$.

of neighbors. The results are averaged over graphs with a similar average degree. For a specific graph, the induced width equals the maximal number of neighbors that have to be considered in a local maximization.

In Fig. 5, we show the timing results for the standard max-plus algorithm; the results for the anytime extension are identical since they only involve an additional check of the global payoff value after every iteration. The plots indicate that the time for the max-plus algorithm grows linearly as the complexity of the graphs increases. This is a result of the relation between the number of messages and the (linearly increasing) number of edges in the graph. The graphs with 10 actions per agent require more time compared to the two other sets because the computation of every message involves a maximization over 100 instead of 25 joint actions. Note that all timing results are generated with a fixed number of 100 iterations. As we will see later, the max-plus algorithm can be stopped earlier without much loss in performance, resulting in even quicker timing results.

For the graphs with a small, less than 2.5, average degree, VE outperforms the max-plus algorithm. In this case, each local maximization only involves a few agents, and VE is able to finish its two passes through the graph quickly. However, the time for the VE algorithm grows exponentially for graphs with a higher average degree because for these graphs it has to enumerate over an increasing number of neighboring agents in each local maximization step. Furthermore, the elimination of an agent often causes a neighboring agent to receive a conditional strategy involving agents it did not have to coordinate with before, changing the graph topology to an even denser graph. This effect becomes more apparent as the graphs become more dense. More specifically, for graphs with 5 actions per agent and an average degree of 5, it takes VE on average 23.8 seconds to generate the joint action. The max-plus algorithm, on the other hand, only requires 10.18 milliseconds for such graphs. There are no clear differences between the two sets with 5 actions per agent since they both require the same number of local maximizations, and the actual values do not influence the algorithm. However, as is seen in Fig. 5(c), the increase of the number of actions per agent slows the

VE algorithm down even more. This is a result of the larger number of joint actions which has to be processed during the local maximizations. For example, during a local maximization of an agent with five neighbors $5^5 = 3,125$ actions have to be enumerated in the case of 5 actions per agent. With 10 actions per agent, this number increases to $10^5 = 100,000$ actions. During elimination the topology of the graph can change to very dense graphs resulting in even larger maximizations. This is also evident from the experiments. For some graphs with ten actions per agent and an average degree higher than 3.2, the size of the intermediate tables grows too large for the available memory, and VE is not able to produce a result. These graphs are removed from the set. For the graphs with an average degree between 3 and 4, this results in the removal of 81 graphs. With an increase of the average degree, this effect becomes more apparent: VE is not able to produce a result for 466 out of the 700 graphs with an average degree higher than 4; all these graphs are removed from the set. This also explains why the increase in the curve of VE in Fig. 5(c) decreases: the more difficult graphs, which take longer to complete, are not taken into account. Even without these graphs, it takes VE on average 339.76 seconds, almost 6 minutes, to produce a joint action for the graphs with an average degree of 5. The max-plus algorithm, on the other hand, needs on average 31.61 milliseconds.

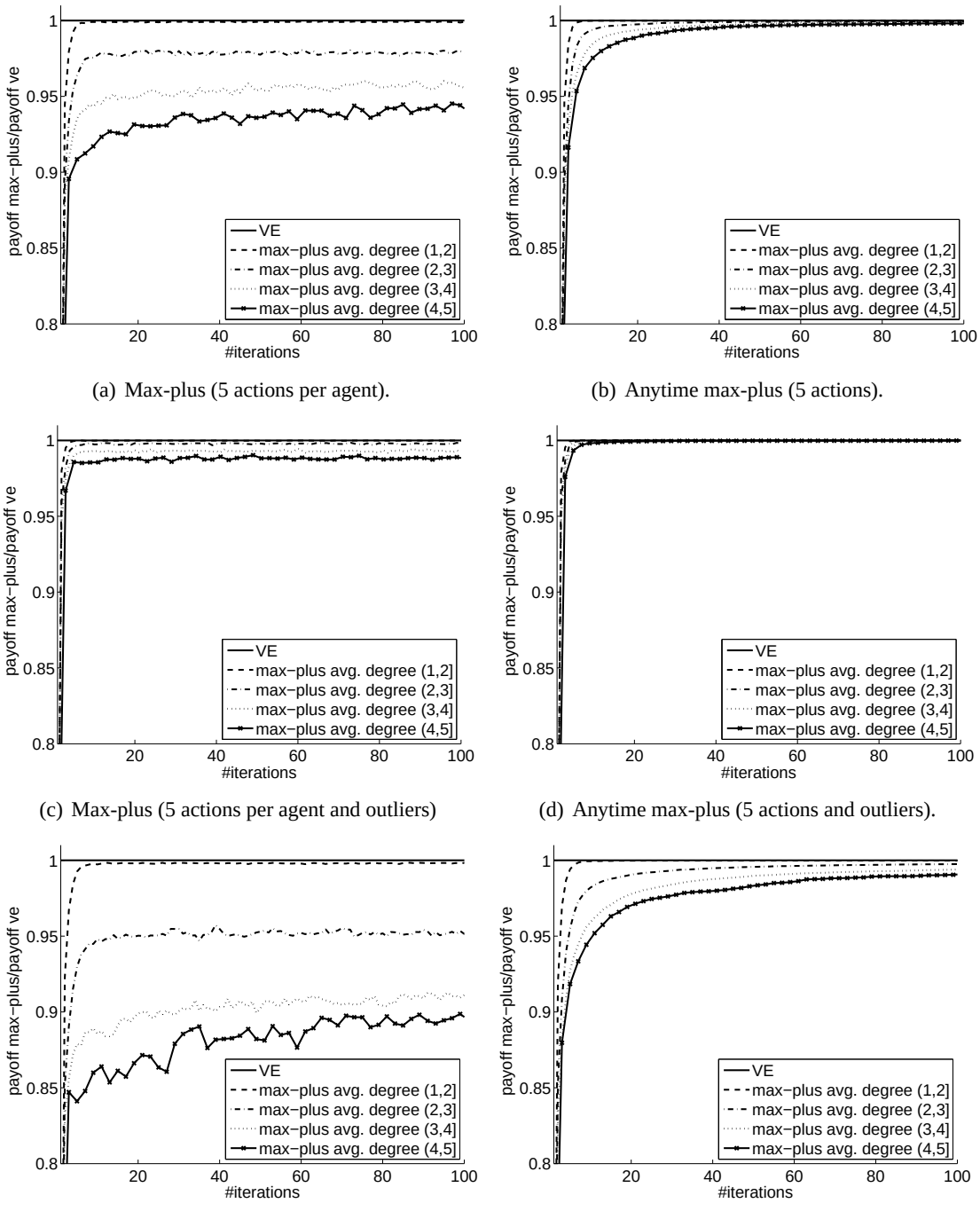
The max-plus algorithm thus outperforms VE with respect to the computation time for densely connected graphs. But how do the resulting joint actions of the max-plus algorithm compare to the optimal solutions of the VE algorithm? Fig. 6 shows the payoff found with the max-plus algorithm relative to the optimal payoff, after each iteration. A relative payoff of 1 indicates that the found joint action corresponds to the optimal joint action, while a relative payoff of 0 indicates that it corresponds to the joint action with the minimal possible payoff. Each of the four displayed curves corresponds to the average result of a subset with a similar average degree. Specifically, each subset contains all graphs with an average degree in $(x-1,x]$, with $x \in \{2,3,4,5\}$.

We first discuss the result of the standard max-plus algorithm in the graphs on the left. For all three sets, the loosely connected graphs with an average degree less than two converge to a similar policy as the optimal joint action in a few iterations only. As the average degree increases, the resulting policy declines. As seen in Fig. 6(c), this effect is less evident in the graphs with outliers; the action combinations related to the positive outliers are clearly preferred, and lowers the number of oscillations. Increasing the number of actions per agent has a negative influence on the result, as is evident from Fig. 6(e), because the total number of action combinations increases. The displayed results are an average of a large set of problems, and an individual run typically contains large oscillations between good and bad solutions.

When using the anytime version, which returns the best joint action found so far, the obtained payoff improves for all graphs. This indicates that the failing convergence of the messages causes the standard max-plus algorithm to oscillate between different joint actions and 'forget' good joint actions. Fig. 6 shows that for all sets near-optimal policies are found, although more complex graphs need more iterations to find them.

4. Collaborative Multiagent Reinforcement Learning

Until now, we have been discussing the problem of selecting an optimal joint action in a group of agents for a given payoff structure and a single state only. Next, we consider *sequential decision-making problems.* In such problems, the agents select a joint action which provides them a reward and causes a transition to a new state. The goal of the agents is to select actions that optimize a performance measure based on the received rewards. This might involve a *sequence* of decisions.



(a) Max-plus (5 actions per agent).

(b) Anytime max-plus (5 actions).

(c) Max-plus (5 actions per agent and outliers)

(d) Anytime max-plus (5 actions and outliers).

(e) Max-plus (10 actions per agent).

(f) Anytime max-plus (10 actions).

Figure 6: Relative payoff compared to VE for both standard max-plus (graphs on the left) and anytime max-plus (graphs on the right) for graphs with 15 agents and cycles.

An important aspect of this problem is that the agents have no prior knowledge about the effect of their actions, but that this information has to be *learned* based on the, possibly delayed, rewards. Next, we review a model to represent such a problem and describe several solution techniques.

4.1 Collaborative Multiagent MDP and Q-Learning

Different models exist to describe a group of agents interacting with their environment. We will use the collaborative multiagent MDP framework (Guestrin, 2003) which is an extension of the single-agent Markov decision process (MDP) framework (Puterman, 1994). It consists of the following model parameters:

- A time step $t = 0, 1, 2, 3, \ldots$.
- A group of n agents $A = \{A_1, A_2, ..., A_n\}$.
- A set of discrete state variables S_i. The global state is the cross-product of all m variables: $S = S_1 \times ... \times S_m$. A state $\mathbf{s}^t \in S$ describes the state of the world at time t.
- A finite set of actions $\mathcal{A}_i$ for every agent i. The action selected by agent i at time step t is denoted by $a_i^t \in \mathcal{A}_i$. The joint action $\mathbf{a}^t \in \mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ is the combination of all individual actions of the n agents.
- A state transition function $T : S \times \mathcal{A} \times S \to [0, 1]$ which gives transition probability $p(\mathbf{s}^{t+1}|\mathbf{s}^t, \mathbf{a}^t)$ that the system will move to state $\mathbf{s}^{t+1}$ when the joint action $\mathbf{a}^t$ is performed in state $\mathbf{s}^t$.
- A reward function $R_i : S \times \mathcal{A} \to \mathbb{R}$ which provides agent i with an individual reward $r_i^t \in R_i(\mathbf{s}^t, \mathbf{a}^t)$ based on the joint action $\mathbf{a}^t$ taken in state $\mathbf{s}^t$. The global reward is the sum of all local rewards: $R(\mathbf{s}^t, \mathbf{a}^t) = \sum_{i=1}^{n} R_i(\mathbf{s}^t, \mathbf{a}^t)$.

This model assumes that the Markov property holds which denotes that the state description at time t provides a complete description of the history before time t. This is apparent in both the transition and reward function in which all information before time t is ignored. Furthermore, it also assumes that the environment is stationary, that is, the reward and transition probabilities are independent of the time step t. Since the transition function is stationary, we will in most cases omit the time step t superscript when referring to a state $\mathbf{s}^t$, and use the shorthand $\mathbf{s}'$ for the next state $\mathbf{s}^{t+1}$.

A policy $\pi : \mathbf{s} \to \mathbf{a}$ is a function which returns an action $\mathbf{a}$ for any given state $\mathbf{s}$. The objective is to find an optimal policy π^* that maximizes the expected discounted future return $V^*(\mathbf{s}) = \max_\pi E\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}^t, \pi(\mathbf{s}^t))|\pi, \mathbf{s}^0 = \mathbf{s}\right]$ for each state $\mathbf{s}$. The expectation operator $E[\cdot]$ averages over stochastic transitions, and $\gamma \in [0, 1)$ is the discount factor. Rewards in the near future are thus preferred over rewards in the distant future. The return is defined in terms of the sum of individual rewards, and the agents thus have to cooperate in order to achieve their common goal. This differs from self-interested approaches (Shapley, 1953; Hansen et al., 2004) in which each agent tries to maximize its own payoff.

Q-functions, or action-value functions, represent the expected future discounted reward for a state $\mathbf{s}$ when selecting a specific action $\mathbf{a}$ and behaving optimally from then on. The optimal Q-function Q^* satisfies the Bellman equation:

$$Q^*(\mathbf{s}, \mathbf{a}) = R(\mathbf{s}, \mathbf{a}) + \gamma \sum_{\mathbf{s}'} p(\mathbf{s}'|\mathbf{s}, \mathbf{a}) \max_{\mathbf{a}'} Q^*(\mathbf{s}', \mathbf{a}'). \tag{9}$$

Given Q^*, the optimal policy for the agents in state $\mathbf{s}$ is to jointly select the action $\arg\max_{\mathbf{a}} Q^*(\mathbf{s},\mathbf{a})$ that maximizes the expected future discounted return.

Reinforcement learning (RL) (Sutton and Barto, 1998; Bertsekas and Tsitsiklis, 1996) can be applied to estimate $Q^*(\mathbf{s},\mathbf{a})$. Q-learning is a widely used learning method for single-agent systems when the agent does not have access to the transition and reward model. The agent interacts with the environment by selecting actions and receives (s,a,r,s') samples based on the experienced state transitions. Q-learning starts with an initial estimate $Q(s,a)$ for each state-action pair. At each time step the agent selects an action based on an exploration strategy. A commonly used strategy is ε-greedy which selects the greedy action, $\arg\max_a Q(s,a)$, with high probability, and, occasionally, with a small probability ε selects an action uniformly at random. This ensures that all actions, and their effects, are experienced. Each time an action a is taken in state s, reward $R(s,a)$ is received, and next state s' is observed, the corresponding Q-value is updated with a combination of its current value and the temporal-difference error, the difference between its current estimate $Q(s,a)$ and the experienced sample $R(s,a)+\gamma\max_{a'} Q(s',a')$, using

$$Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{10}$$

where $\alpha \in (0,1)$ is an appropriate learning rate which controls the contribution of the new experience to the current estimate. When every state-action pair is associated with a unique Q-value and every action is sampled infinitely often (as with the ε-greedy action selection method), iteratively applying (10) is known to converge to the optimal $Q^*(s,a)$ values (Watkins and Dayan, 1992).

Next, we describe four multiagent variants of tabular Q-learning to multiagent environments, and discuss their advantages and disadvantages. We do not consider any function-approximation algorithms. Although they have been been successfully applied in several domains with large state sets, they are less applicable for large action sets since it is more difficult to generalize over nearby (joint) actions. Furthermore, we only consider model-free methods in which the agents do not have access to the transition and reward function. The agents do observe the current state and also receive an individual reward depending on the performed joint action and the unknown reward function. Finally, we assume the agents are allowed to communicate in order to coordinate their actions.

4.2 MDP Learners

In principle, a collaborative multiagent MDP can be regarded as one large single agent in which each joint action is represented as a single action. It is then possible to learn the optimal Q-values for the joint actions using standard single-agent Q-learning, that is, by iteratively applying (10). In this *MDP learners* approach either a central controller models the complete MDP and communicates to each agent its individual action, or each agent models the complete MDP separately and selects the individual action that corresponds to its own identity. In the latter case, the agents do not need to communicate but they have to be able to observe the executed joint action and the received individual rewards. The problem of exploration is solved by using the same random number generator (and the same seed) for all agents (Vlassis, 2003).

Although this approach leads to the optimal solution, it is infeasible for problems with many agents. In the first place, it is intractable to model the complete joint action space, which is exponential in the number of agents. For example, a problem with 7 agents, each able to perform 6 actions, results in almost $280,000$ Q-values per state. Secondly, the agents might not have access to the needed information for the update because they are not able to observe the state, action, and

reward of all other agents. Finally, it will take many time steps to explore all joint actions resulting in slow convergence.

4.3 Independent Learners

At the other extreme, we have the *independent learners* (IL) approach (Claus and Boutilier, 1998) in which the agents ignore the actions and rewards of the other agents, and learn their strategies independently. Each agent stores and updates an individual table Q_i and the global Q-function is defined as a linear combination of all individual contributions, $Q(\mathbf{s},\mathbf{a}) = \sum_{i=1}^{n} Q_i(\mathbf{s},a_i)$. Each local Q-function is updated using

$$Q_i(\mathbf{s},a_i) := Q_i(\mathbf{s},a_i) + \alpha[R_i(\mathbf{s},\mathbf{a}) + \gamma \max_{a_i'} Q_i(\mathbf{s}',a_i') - Q_i(\mathbf{s},a_i)]. \tag{11}$$

Note that each Q_i is based on the global state $\mathbf{s}$. This approach results in big storage and computational savings in the action-space, for example, with 7 agents and 6 actions per agent only 42 Q-values have to be stored per state. However, the standard convergence proof for Q-learning does not hold anymore. Because the actions of the other agents are ignored in the representation of the Q-functions, and these agents also change their behavior while learning, the system becomes non-stationary from the perspective of an individual agent. This might lead to oscillations. Despite the lack of guaranteed convergence, this method has been applied successfully in multiple cases (Tan, 1993; Sen et al., 1994).

4.4 Coordinated Reinforcement Learning

In many situations an agent has to coordinate its actions with a few agents only, and acts independently with respect to the other agents. In Guestrin et al. (2002b) three different *Coordinated Reinforcement Learning* approaches are described which take advantage of the structure of the problem. The three approaches are respectively a variant of Q-learning, policy iteration, and direct policy search. We will concentrate on the Q-learning variant which decomposes the global Q-function into a linear combination of local agent-dependent Q-functions. Each local Q_i is based on a subset of all state and action variables,

$$Q(\mathbf{s},\mathbf{a}) = \sum_{i=1}^{n} Q_i(\mathbf{s}_i,\mathbf{a}_i), \tag{12}$$

where $\mathbf{s}_i$ and $\mathbf{a}_i$ are respectively the subset of state and action variables related to agent i. These dependencies are established beforehand and differ per problem. Note that in this representation, each agent only needs to observe the state variables $\mathbf{s}_i$ which are part of its local Q_i-function. The corresponding CG is constructed by adding an edge between agent i and j when the action of agent j is included in the action variables of agent i, that is, $a_j \in \mathbf{a}_i$. As an example, imagine a computer network in which each machine is modeled as an agent and only depends on the state and action variables of itself and the machines it is connected to. The coordination graph would in this case equal the network topology.

A local Q_i is updated using the global temporal-difference error, the difference between the current global Q-value and the expected future discounted return for the experienced state transition, using

$$Q_i(\mathbf{s}_i,\mathbf{a}_i) := Q_i(\mathbf{s}_i,\mathbf{a}_i) + \alpha[R(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}') - Q(\mathbf{s},\mathbf{a})]. \tag{13}$$

The global reward $R(\mathbf{s},\mathbf{a})$ is given. The maximizing action in $\mathbf{s}'$ and the associated maximal expected future return, $\max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')$, are computed in a distributed manner by applying the VE algorithm discussed in Section 2 on the CG. The estimate of the global Q-value in $\mathbf{s}$, $Q(\mathbf{s},\mathbf{a})$ in (13), is computed by fixing the action of every agent to the one assigned in $\mathbf{a}$ and applying a message passing scheme similar to the one used in the VE algorithm. We use a table-based representation for the Q-functions in our notation. However, since each individual Q-function is entirely local, each agent is allowed to choose its own representation, for example, using a function approximator as in Guestrin et al. (2002b).

The advantage of this method is that it is completely distributed. Each agent keeps a local Q-function and only has to exchange messages with its neighbors in the graph in order to compute the global Q-values. In sparsely connected graphs, this results in large computational savings since it is not necessary to consider the complete joint action-space. However, the algorithm is still slow for densely connected graphs because of two main reasons. First, the size of each local Q-function grows exponentially with the number of neighbors of the corresponding agent. Secondly, the computational complexity of the VE algorithm is exponential in the induced width of the graph, as shown in Section 3.2.

4.5 Distributed Value Functions

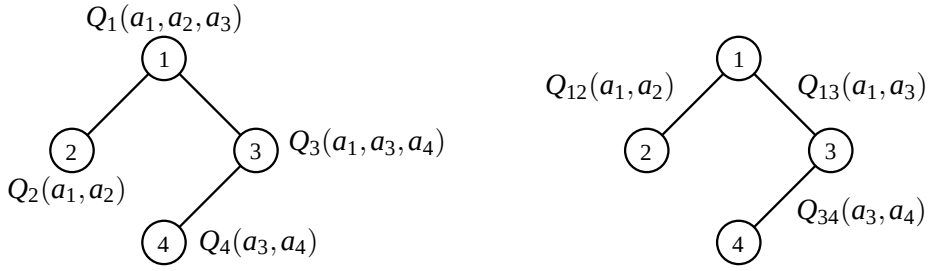
Another method to decompose a large action space is the distributed value functions (DVF) approach (Schneider et al., 1999). Each agent maintains an individual local Q-function, $Q_i(\mathbf{s}_i,a_i)$, based on its individual action and updates it by incorporating the Q-functions of its neighboring agents. A weight function $f(i,j)$ determines how much the Q-value of an agent j contributes to the update of the Q-value of agent i. This function defines a graph structure of agent dependencies, in which an edge is added between agents i and j if the corresponding function $f(i,j)$ is non-zero. The update looks as follows:

$$Q_i(\mathbf{s}_i,a_i) := (1-\alpha)Q_i(\mathbf{s}_i,a_i) + \alpha[R_i(\mathbf{s},\mathbf{a}) + \gamma \sum_{j\in\{i\cup\Gamma(i)\}} f(i,j) \max_{a'_j} Q_j(\mathbf{s}',a'_j)]. \tag{14}$$

Note that $f(i,i)$ also has to be defined and specifies the agent's contribution to the current estimate. A common approach is to weigh each neighboring function equally, $f(i,j) = 1/|i\cup\Gamma(j)|$. Each Q-function of an agent i is thus divided proportionally over its neighbors and itself. This method scales linearly in the number of agents.

5. Sparse Cooperative Q-Learning

In this section, we describe our Sparse Cooperative Q-learning, or SparseQ, methods which also approximate the global Q-function into a linear combination of local Q-functions. The decomposition is based on the structure of a CG which is chosen beforehand. In principle we can select any arbitrary CG, but in general a CG based on the problem under study is used. For a given CG, we investigate both a decomposition in terms of the nodes (or agents), as well as the edges. In the agent-based decomposition the local function of an agent is based on its own action and those of its neighboring agents. In the edge-based decomposition each local function is based on the actions of the two agents it is connected to. In order to update a local function, the key idea is to base the update not on the difference between the current global Q-value and the experienced global discounted



(a) Agent-based decomposition. (b) Edge-based decomposition.

Figure 7: An agent-based and edge-based decomposition of the global Q-function for a 4-agent problem.

return, but rather on the current local Q-value and the *local contribution* of this agent to the global return.

Next, we describe an agent-based decomposition of the global Q-function and explain how the local contribution of an agent is used in the update step. Thereafter, we describe an edge-based decomposition, and a related edge-based and agent-based update method.

5.1 Agent-Based Decomposition[4]

As in Guestrin et al. (2002b) we decompose the global Q-function over the different agents. Every agent i is associated with a local Q-function $Q_i(\mathbf{s}_i,\mathbf{a}_i)$ which only depends on a subset of all possible state and action variables. These dependencies are specified beforehand and depend on the problem. The Q_i-functions correspond to a CG which is constructed by connecting each agent with all agents in which its action variable is involved. See Fig. 7(a) for an example of an agent-based decomposition for a 4-agent problem.

Since the global Q-function equals the sum of the local Q-functions of all n agents, $Q(\mathbf{s},\mathbf{a}) = \sum_{i=1}^{n} Q_i(\mathbf{s}_i,\mathbf{a}_i)$, it is possible to rewrite the Q-learning update rule in (10) as

$$\sum_{i=1}^{n} Q_i(\mathbf{s}_i,\mathbf{a}_i) := \sum_{i=1}^{n} Q_i(\mathbf{s}_i,\mathbf{a}_i) + \alpha\big[\sum_{i=1}^{n} R_i(\mathbf{s},\mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}') - \sum_{i=1}^{n} Q_i(\mathbf{s}_i,\mathbf{a}_i)\big]. \tag{15}$$

Only the expected discounted return, $\max_{a'} Q(\mathbf{s}',\mathbf{a}')$, cannot be directly written as the sum of local terms since it depends on the *globally* maximizing joint action. However, we can use the VE algorithm to compute, in a distributed manner, the maximizing joint action $\mathbf{a}^* = \arg\max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')$ in state $\mathbf{s}'$, and from this compute the local contribution $Q_i(\mathbf{s}'_i,\mathbf{a}^*_i)$ of each agent to the total action value $Q(\mathbf{s}',\mathbf{a}^*)$. Note that the local contribution of an agent to the global action value might be lower than the maximizing value of its local Q-function because it is unaware of the dependencies of its neighboring agents with the other agents in the CG. Since we can substitute $\max_{a'} Q(\mathbf{s}',\mathbf{a}')$ with $\sum_{i=1}^{n} Q_i(\mathbf{s}',\mathbf{a}^*_i)$, we are able to decompose all terms in (15) and rewrite the update for each agent i separately:

$$Q_i(\mathbf{s}_i,\mathbf{a}_i) := Q_i(\mathbf{s}_i,\mathbf{a}_i) + \alpha[R_i(\mathbf{s},\mathbf{a}) + \gamma Q_i(\mathbf{s}'_i,\mathbf{a}^*_i) - Q_i(\mathbf{s}_i,\mathbf{a}_i)]. \tag{16}$$

[4] Subsection 5.1 is based on (Kok and Vlassis, 2004).

This update is completely based on local terms and only requires the distributed VE algorithm to compute the maximizing joint action $\mathbf{a}^*$. In contrast to CoordRL, we directly take advantage of the local rewards received by the different agents. Especially for larger problems with many agents, this allows us to propagate back the reward to the local functions related to the agents responsible for the generated rewards. This is not possible in CoordRL which uses the global reward to update the different local functions. As a consequence, the agents are not able to distinguish which agents are responsible for the received reward, and all functions, including the ones which are not related to the received reward, are updated equally. It might even be the case that the high reward generated by one agent, or a group of agents, is counterbalanced by the negative reward of another agent. In this case, the combined global reward equals zero and no functions are updated.

Just as the coordinated RL approach, both the representation of the local Q_i-functions and the VE algorithm grow exponentially with the number of neighbors. This becomes problematic for densely connected graphs, and therefore we also investigate an edge-based decomposition of the Q-function which does not suffer from this problem in the next section.

5.2 Edge-Based Decomposition

A different method to decompose the global Q-function is to define it in terms of the edges of the corresponding CG. Contrary to the agent-based decomposition, which scales exponentially with the number of neighbors in the graph, the edge-based decomposition scales linearly in the number of neighbors. For a coordination graph $G = (V, E)$ with $|V|$ vertices and $|E|$ edges, each edge $(i, j) \in E$ corresponds to a local Q-function Q_{ij}, and the sum of all local Q-functions defines the global Q-function:

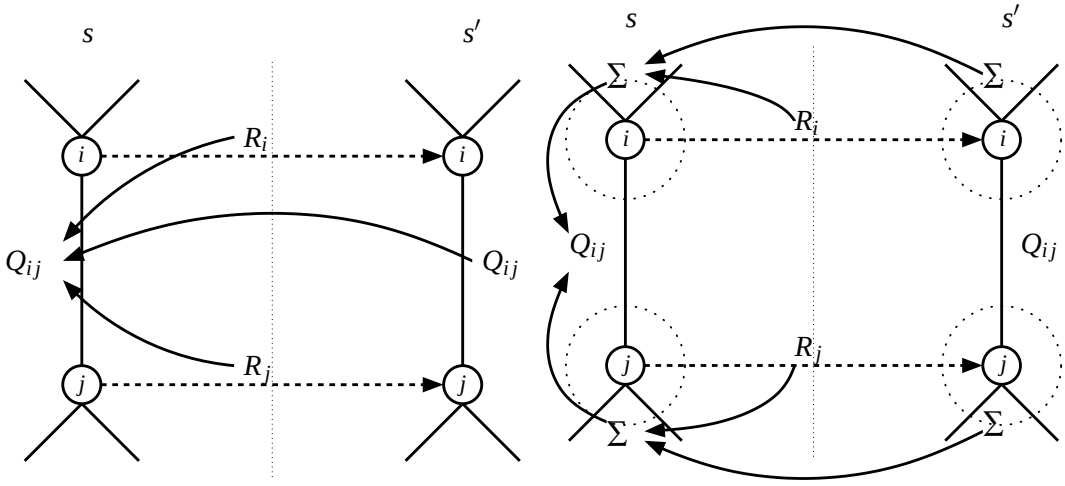
$$Q(\mathbf{s}, \mathbf{a}) = \sum_{(i,j)\in E} Q_{ij}(\mathbf{s}_{ij}, a_i, a_j), \tag{17}$$

where $\mathbf{s}_{ij} \subseteq \mathbf{s}_i \cup \mathbf{s}_j$ is the subset of the state variables related to agent i and agent j which are relevant for their dependency. Note that each local Q-function Q_{ij} always depends on the actions of two agents, a_i and a_j, only. Fig. 7(b) shows an example of an edge-based decomposition for a 4-agent problem.

An important consequence of this decomposition is that it only depends on pairwise functions. This allows us to directly apply the max-plus algorithm from Section 3 to compute the maximizing joint action. Now, both the decomposition of the action-value function and the method for action selection scale linearly in the number of dependencies, resulting in an approach that can be applied to large agent networks with many dependencies.

In order to update a local Q-function, we have to propagate back the reward received by the individual agents. This is complicated by the fact that the rewards are received per agent, while the local Q-functions are defined over the edges. For an agent with multiple neighbors it is therefore not possible to derive which dependency generated (parts of) the reward. Our approach is to associate each agent with a local Q-function Q_i that is directly computed from the edge-based Q-functions Q_{ij}. This allows us to relate the received reward of an agent directly to its agent-based Q-function Q_i. In order to compute Q_i, we assume that each edge-based Q-function contributes equally to the two agents that form the edge. Then, the local Q-function Q_i of agent i is defined as the summation of half the value of all local Q-functions Q_{ij} of agent i and its neighbors $j \in \Gamma(i)$, that is,

$$Q_i(\mathbf{s}_i, \mathbf{a}_i) = \frac{1}{2} \sum_{j\in\Gamma(i)} Q_{ij}(\mathbf{s}_{ij}, a_i, a_j). \tag{18}$$



(a) Edge-based update. (b) Agent-based update.

Figure 8: A graphical representation of the edge-based and agent-based update method after the transition from state $\mathbf{s}$ to $\mathbf{s}'$. See the text for a detailed description.

The sum of all local Q-functions Q_i equals Q in (17). Next, we describe two update methods for the edge-based decomposition defined in terms of these local agent-based Q-functions.

5.2.1 EDGE-BASED UPDATE

The first update method we consider updates each local Q-function Q_{ij} based on its current estimate and its contribution to the maximal return in the next state. For this, we rewrite (16) by replacing every instance of Q_i with its definition in (18) to

$$\frac{1}{2}\sum_{j\in\Gamma(i)} Q_{ij}(\mathbf{s}_{ij},a_i,a_j) := \frac{1}{2}\sum_{j\in\Gamma(i)} Q_{ij}(\mathbf{s}_{ij},a_i,a_j) + \alpha\left[\sum_{j\in\Gamma(i)}\frac{R_i(\mathbf{s},\mathbf{a})}{|\Gamma(i)|} + \gamma\frac{1}{2}\sum_{j\in\Gamma(i)} Q_{ij}(\mathbf{s}'_{ij},a_i^*,a_j^*) - \frac{1}{2}\sum_{j\in\Gamma(i)} Q_{ij}(\mathbf{s}_{ij},a_i,a_j)\right]. \quad (19)$$

Note that in this decomposition for agent i we made the assumption that the reward R_i is divided proportionally over its neighbors $\Gamma(i)$. In order to get an update equation for an individual local Q-function Q_{ij}, we remove the sums. Because, one half of every local Q-function Q_{ij} is updated by agent i and the other half by agent j, agent j updates the local Q-function Q_{ij} using a similar decomposition as (19). Adding the two gives the following update equation for a single local Q-function Q_{ij}:

$$Q_{ij}(\mathbf{s}_{ij},a_i,a_j) := Q_{ij}(\mathbf{s}_{ij},a_i,a_j) + \alpha\left[\frac{R_i(\mathbf{s},\mathbf{a})}{|\Gamma(i)|} + \frac{R_j(\mathbf{s},\mathbf{a})}{|\Gamma(j)|} + \gamma Q_{ij}(\mathbf{s}'_{ij},a_i^*,a_j^*) - Q_{ij}(\mathbf{s}_{ij},a_i,a_j)\right]. \quad (20)$$

Each local Q-function Q_{ij} is updated with a proportional part of the received reward of the two agents it is related to and with the contribution of this edge to the maximizing joint action $\mathbf{a}^* =$

$(a_i^*) = \arg\max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')$ in the state $\mathbf{s}'$. The latter is computed by either applying the exact VE algorithm or the approximate max-plus algorithm. We can also derive (20) from (10) directly using (17). However, we want to emphasize that it is possible to derive this update rule from the agent-based decomposition discussed in Section 5.1.

Fig. 8(a) shows a graphical representation of the update. The left part of the figure shows a partial view of a CG in state $\mathbf{s}$. Only the agents i and j, their connecting edge, which is related to a local edge-based Q-function Q_{ij}, and some outgoing edges are depicted. The right part of the figure shows the same structure for state s'. Following (20), a local Q-function Q_{ij} is directly updated based on the received reward of the involved agents and the maximizing local Q-function Q_{ij} in the next state.

5.2.2 AGENT-BASED UPDATE

In the edge-based update method the reward is divided proportionally over the different edges of an agent. All other terms are completely local and only correspond to the local Q-function Q_{ij} of the edge that is updated. A different approach is to first compute the temporal-difference error *per agent* and divide this value over the edges. For this, we first rewrite (16) for agent i using (18) to

$$\frac{1}{2}\sum_{j\in\Gamma(i)} Q_{ij}(\mathbf{s}_{ij}, a_i, a_j) := \frac{1}{2}\sum_{j\in\Gamma(i)} [Q_{ij}(\mathbf{s}_{ij}, a_i, a_j)] + \alpha[R_i(\mathbf{s},\mathbf{a}) + \gamma Q_i(\mathbf{s}_i', \mathbf{a}_i^*) - Q_i(\mathbf{s}_i, \mathbf{a}_i)]. \quad (21)$$

In order to transfer (21) into a local update function, we first rewrite the temporal-difference error as a summation of the neighbors of agent i, by

$$R_i(\mathbf{s},\mathbf{a}) + \gamma Q_i(\mathbf{s}_i', \mathbf{a}_i^*) - Q_i(\mathbf{s}_i, \mathbf{a}_i) = \sum_{j\in\Gamma(i)} \frac{R_i(\mathbf{s},\mathbf{a}) + \gamma Q_i(\mathbf{s}_i', \mathbf{a}_i^*) - Q_i(\mathbf{s}_i, \mathbf{a}_i)}{|\Gamma(i)|}. \quad (22)$$

Note that this summation only decomposes the temporal-difference error into j equal parts, and thus does not use j explicitly. Because now all summations are identical, we can decompose (21) by removing the sums. Just as in the edge-based update, there are two agents which update the same local Q-function Q_{ij}. When we add the contributions of the two involved agents i and j, we get the local update equation

$$Q_{ij}(\mathbf{s}_{ij}, a_i, a_j) := Q_{ij}(\mathbf{s}_{ij}, a_i, a_j) + \alpha \sum_{k\in\{i,j\}} \frac{R_k(\mathbf{s},\mathbf{a}) + \gamma Q_k(\mathbf{s}_k', \mathbf{a}_k^*) - Q_k(\mathbf{s}_k, \mathbf{a}_k)}{|\Gamma(k)|}. \quad (23)$$

This agent-based update rule propagates back the temporal-difference error from the two agents which are related to the local Q-function of the edge that is updated, and incorporates the information of *all* edges of these agents. This is different from the edge-based update method which directly propagates back the temporal-difference error related to the edge that is updated. This is depicted in Fig. 8(b). Again, the left part of the figure represents the situation in state $\mathbf{s}$, and the right part the situation in the next state $\mathbf{s}'$. The edge-based Q-function Q_{ij} is updated based on the local agent-based Q-functions of the two agents that form the edge. These functions are computed by summing over the local edge-based Q-functions of all neighboring edges.

Next, we will describe several experiments and solve them using both the agent-based and the edge-based decomposition. For the latter, we apply both the agent-based and edge-based update method, and show the consequences, both in speed and solution quality, of using the max-plus algorithm as an alternative to the VE algorithm.

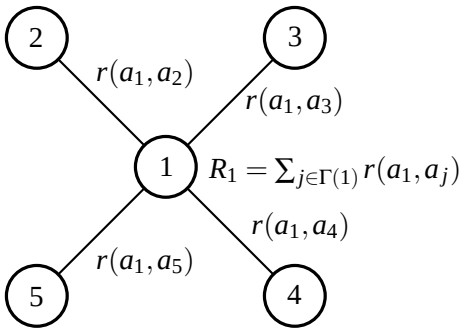
6. Experiments

In this section, we describe experiments using the methods discussed in Section 4 and Section 5. We give results on a large single-state problem and on a distributed sensor network problem, which was part of the NIPS 2005 benchmarking workshop. We selected these problems because they are both fully specified and, more importantly, require the selection of a specific combination of actions at every time step. This is in contrast with other experiments in which coordination can be modeled through the state variables, that is, each agent is able to select its optimal action based on only the state variables (for example, its own and other agents' positions) and does not have to model the actions of the other agents (Tan, 1993; Guestrin et al., 2002b; Becker et al., 2003).

6.1 Experiments on Single-State Problems

Now, we describe several experiments in which a group of n agents have to learn to take the optimal joint action in a single-state problem. The agents repeatedly interact with their environment by selecting a joint action. After the execution of a joint action $\mathbf{a}$, the episode is immediately ended and the system provides each agent an individual reward $R_i(\mathbf{a})$. The goal of the agents is to select the joint action $\mathbf{a}$ which maximizes $R(\mathbf{a}) = \sum_{i=1}^{n} R_i(\mathbf{a})$. The local reward R_i received by an agent i only depends on a subset of the actions of the other agents. These dependencies are modeled using a graph in which each edge corresponds to a local reward function that assigns a value $r(a_i, a_j)$ to each possible action combination of the actions of agent i and agent j. Each local reward function is fixed beforehand and contains one specific pair of actions, $(\tilde{a}_i, \tilde{a}_j)$ that results in a high random reward, uniformly distributed in the range $[5, 15]$, that is, $5 + \mathcal{U}([0, 10])$. However, failure of coordination, that is, selecting an action $r(\tilde{a}_i, a_j)$ with $a_j \neq \tilde{a}_j$ or $r(a_i, \tilde{a}_j)$ with $a_i \neq \tilde{a}_i$, will always result in a reward of 0. All remaining joint actions, $r(a_i, a_j)$ with $a_i \neq \tilde{a}_i$ and $a_j \neq \tilde{a}_j$, give a default reward from the uniform distribution $\mathcal{U}([0, 10])$. The individual reward R_i for an agent i equals the sum of the local rewards resulting from the interactions with its neighbors, $R_i(\mathbf{a}) = \sum_{j \in \Gamma(i)} r(a_i, a_j)$. Fig. 9 shows an example of the construction of the individual reward received by an agent based on its interaction with its four neighbors, together with an example reward function $r(a_i, a_j)$ corresponding to an edge between agent i and agent j.

The goal of the agents is to learn, based on the received individual rewards, to select a joint action that maximizes the global reward. Although we assume that the agents know on which other agents it depends, this goal is complicated by two factors. First, the outcome of a selected action of an agent also depends on the actions of its neighbors. For example, the agents must coordinate in order to select the joint action $(\tilde{a}_i, \tilde{a}_j)$ which, in most cases, returns the highest reward. Failure of coordination, however, results in a low reward. Secondly, because each agent only receives an individual reward, they are not able to derive which neighbor interaction caused which part of the reward. An important difference with the problem specified in Section 3.2, in which the the agents have to select a joint action that maximizes predefined payoff functions, is that in this case the payoff relations themselves have to be learned based on the received rewards.



(a) Construction of reward $R_1(\mathbf{a})$.

action agent i \ action agent j	1	2	3	4
1	9.155	0	9.293	0.196
2	7.651	0	2.872	1.287
3	0	12.020	0	0
4	4.536	0	1.581	8.138

(b) Example $r(a_i,a_j)$ function.

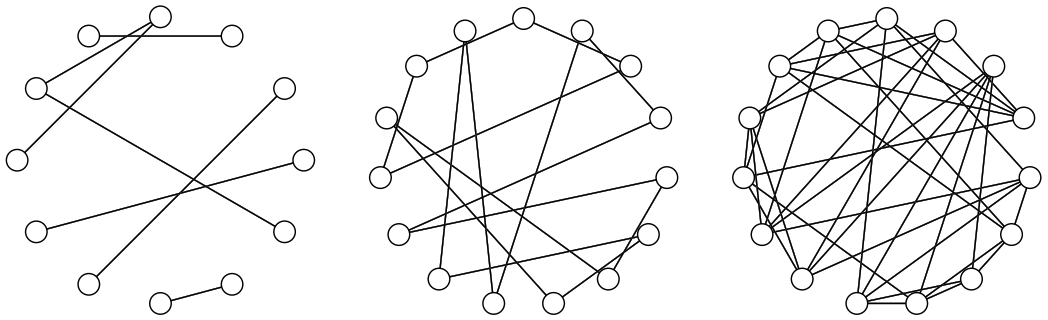
Figure 9: Construction of the reward for agent 1 in the single-state problem. (a) The individual reward R_1 is the sum of the rewards $r(a_1,a_j)$ generated by the interactions with its neighbors $j \in \Gamma(1) = \{2,3,4,5\}$. (b) Example $r(a_i,a_j)$ function.

We perform experiments with 12 agents, each able to perform 4 actions. The group as a whole thus has $4^{12} \approx 1.7 \cdot 10^7$, or 17 million, different joint actions to choose from. We investigate reward functions with different complexities, and apply the method described in Section 3.2 to randomly generate 20 graphs $G = (V,E)$ with $|V| = 12$ for each $|E| \in \{7,8,\ldots,30\}$. This results in 480 graphs, 20 graphs in each of the 24 groups. The agents of the simplest graphs (7 edges) have an average degree of 1.16, while the most complex graphs (30 edges) have an average degree of 5. Fig. 10 shows three different example graphs with different average degrees. Fig. 10(a) and (c) depict respectively the minimum and maximal considered average degree, while Fig. 10(b) shows a graph with an average degree of 2.

We apply the different variants of our sparse cooperative Q-learning method described in Section 5 and different existing multiagent Q-learning methods, discussed in Section 4, to this problem. Since the problem consists of only a single state, all Q-learning methods store Q-functions based on actions only. Furthermore, we assume that the agents have access to a CG which for each agent specifies on which other agents it depends. This CG is identical to the topology of the graph that is used to generate the reward function. Apart from the different Q-learning methods, we also apply an approach that selects a joint action uniformly at random and keeps track of the best joint action found so far, and a method that enumerates all possible joint actions and stores the one with the highest reward. To summarize, we now briefly review the main characteristics of all applied methods:

Independent learners (IL) Each agent i stores a local Q-function $Q_i(a_i)$ only depending on its own action. Each update is performed using the private reward R_i according to (11). An agent selects an action that maximizes its own local Q-function Q_i.

Distributed value functions (DVF) Each agent i stores a local Q-function based on its own action, and an update incorporates the Q-functions of its neighbors following (14). For stateless problems, as the ones in this section, the Q-value of the next state is not used and this method is identical to IL.



(a) A graph with 7 edges (average degree of 1.16).

(b) A graph with 12 edges (average degree of 2).

(c) A graph with 30 edges (average degree of 5.00).

Figure 10: Example graphs with 12 agents and different average degrees.

Coordinated reinforcement learning (CoordRL) Each agent i stores an individual Q-function based on its own action and the actions of its neighbors $j \in \Gamma(i)$. Each function is updated based on the *global* temporal-difference error using the update equation in (13). This representation scales exponentially with the number of neighbors. VE is used to determine the optimal joint action which scales exponentially with the induced width of the graph.

Sparse cooperative Q-learning, agent-based (SparseQ agent) Each agent stores a Q-function that is based on its own action and the actions of its neighbors $j \in \Gamma(i)$. A function is updated based on the *local* temporal-difference error following (16). The representation and computational complexity are similar to the CoordRL approach.

Sparse cooperative Q-learning, edge-based (SparseQ edge) Each edge in the used CG is associated with a Q-function based on the actions of the two connected agents. We apply both the *edge-based* update method (SparseQ edge, edge) from (20) which updates a Q-function based on the value of the edge that is updated, and the *agent-based* update method (SparseQ edge, agent) from (23), which updates a Q-function based on the local Q-functions of the agents forming the edge.

The two update methods are both executed with the VE algorithm and the anytime max-plus algorithm in order to determine the optimal joint action, resulting in four different methods in total. The max-plus algorithm generates a result when either the messages converge, the best joint action has not improved for 5 iterations, or more than 20 iterations are performed. The latter number of iterations is obtained by comparing the problem under study with the coordination problem addressed in Section 3.2. Both problem sizes are similar, and as is visible in Fig. 6 the coordination problem reaches a good performance after 20 iterations.

Random method with memory Each iteration, each agent selects an action uniformly at random. The resulting joint action is evaluated and compared to the best joint action found so far. The best one is stored and selected.

Enumeration In order to compare the quality of the different methods, we compute the optimal value by trying every possible joint action and store the one which results in the highest reward. This requires an enumeration over all possible joint actions. Note that this approach

method	$(1,2]$	$(2,3]$	$(3,4]$	$(4,5]$
IL/DVF	48	48	48	48
edge-based	152	248	344	440
agent-based	528	2,112	8,448	33,792

Table 2: Average number of Q-values needed for the different decompositions for graphs with an average degree in $(x-1,x]$.

does not perform any updates, and quickly becomes intractable for problems larger than the one addressed here.

We do not apply the MDP learners approach since it would take too long to find a solution. First, it requires an enumeration over 4^{12} (≈ 17 million) actions at every time step. Secondly, assuming there is only one optimal joint action, the probability to actually find the optimal joint action is negligible. An exploration action should be made (probability ε), and this exploration action should equal the optimal joint action (probability of $\frac{1}{4^{12}}$).

Table 2 shows the average number of Q-values required by each of the three types of decompositions. The numbers are based on the generated graphs and averaged over similarly shaped graphs. Note the exponential growth in the agent-based decomposition that is used in both the CoordRL and agent-based SparseQ approach.

We run each method on this problem for $15,000$ learning cycles. Each learning cycle is directly followed by a test cycle in which the reward related to the current greedy joint action is computed. The values from the test cycles, thus without exploration, are used to compare the performance between the different methods. For all Q-learning variants, the Q-values are initialized to zero and the parameters are set to $\alpha = 0.2$, $\varepsilon = 0.2$, and $\gamma = 0.9$.

Fig. 11 shows the timing results for all methods.[5] The x-axis depicts the average degree of the graph. The y-axis, shown in logarithmic scale, depicts the average number of seconds spent in the $15,000$ learning cycles on graphs with a similar average degree. For the enumeration method it represents the time for computing the reward of all joint actions.

The results show that the random and IL/DVF approach are the quickest and take less than a second to complete. In the IL/DVF method each agent only stores functions based on its individual action and is thus constant in the number of dependencies in the graph. Note that the time increase in the random approach for graphs with a higher average degree is caused by the fact that more local reward functions have to be enumerated in order to compute the reward. This occurs in all methods, but is especially visible in the curve of the random approach since for this method the small absolute increase is relatively large with respect to its computation time.

The CoordRL and the agent-based SparseQ method scale exponentially with the increase of the average degree, both in their representation of the local Q-functions and the computation of the optimal joint action using the VE algorithm. The curves of these methods overlap in Fig. 11. Because these methods need a very long time, more than a day, to process graphs with a higher average degree than 3, the results for graphs with more than 18 edges are not computed. The edge-based decompositions do not suffer from the exponential growth in the representation of the

[5] All results are generated on an Intel Xeon 3.4GHz / 2GB machine using a C++ implementation.

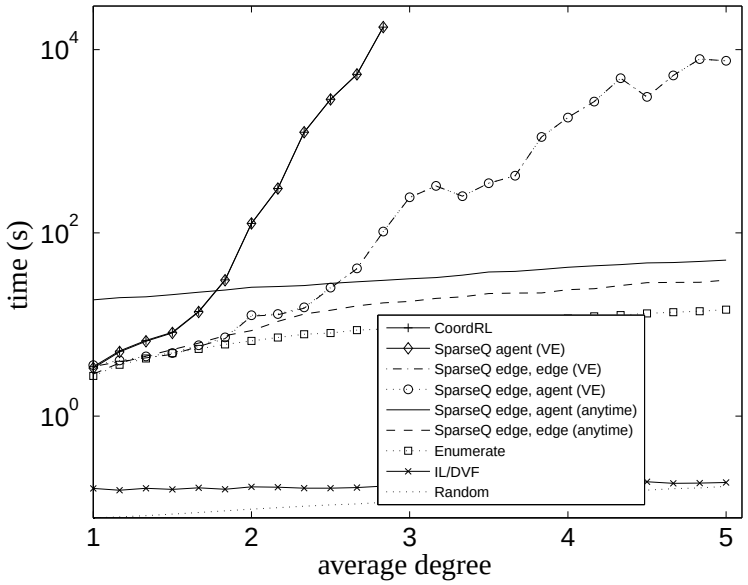


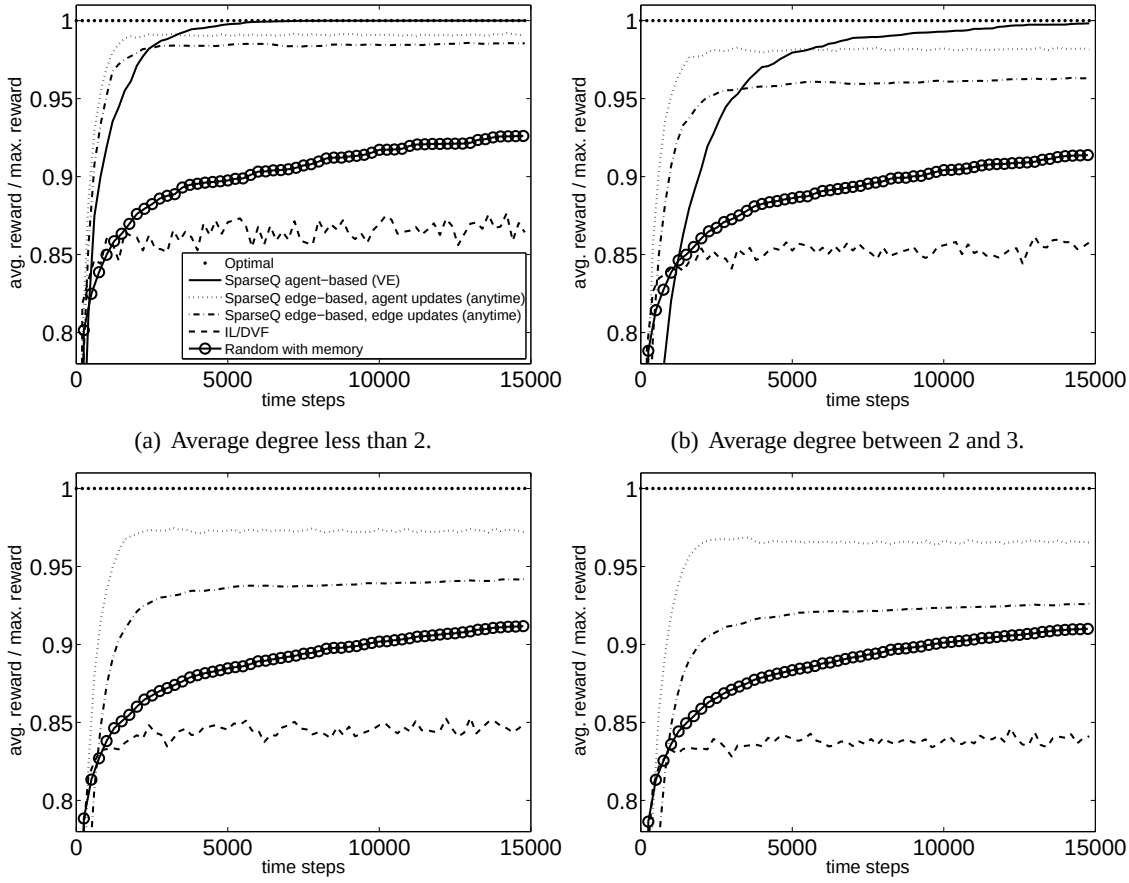
Figure 11: Timing results for the different methods applied to the single-state problems with 12 agents and an increasing number of edges. The results overlap for the CoordRL and the agent-based SparseQ decomposition, and the two edge-based decompositions using the VE algorithm.

local Q-functions. However, this approach still grows exponentially with an increase of the average degree when the VE algorithm is used to compute the maximizing joint action. This holds for both the agent-based and edge-based update method, which overlap in the graph. When the anytime max-plus algorithm is applied to compute the joint action, both the representation of the Q-function and the computation of the joint action scale linearly with an increasing average degree. The agent-based update method is slightly slower than the edge-based update method because the first incorporates the neighboring Q-functions in its update (23), and therefore the values in the Q-functions are less distinct. As a consequence, the max-plus algorithm needs more iterations in an update step to find the maximizing joint action.

Finally, the enumeration method shows a slight increase in the computation time with an increase of the average degree because it has to sum over more local functions for the denser graphs when computing the associated value. Note that the problem size was chosen such that the enumeration method was able to produce a result for all different graphs.

Fig. 12 shows the corresponding performance for the most relevant methods. Each figure depicts the running average, of the last 10 cycles, of the obtained reward relative to the optimal reward for the first $15,000$ cycles. The optimal reward is determined using the enumeration method. Results are grouped for graphs with a similar complexity, that is, having about the same number of edges per graph.

Fig. 12(a) depicts the results for the simplest graphs with an average degree less than or equal to 2. We do not show the results for the CoordRL approach since it is not able to learn a good policy and quickly stabilize around 41% of the optimal value. This corresponds to a method in which



(a) Average degree less than 2.

(b) Average degree between 2 and 3.

(c) Average degree between 3 and 4.

(d) Average degree between 4 and 5.

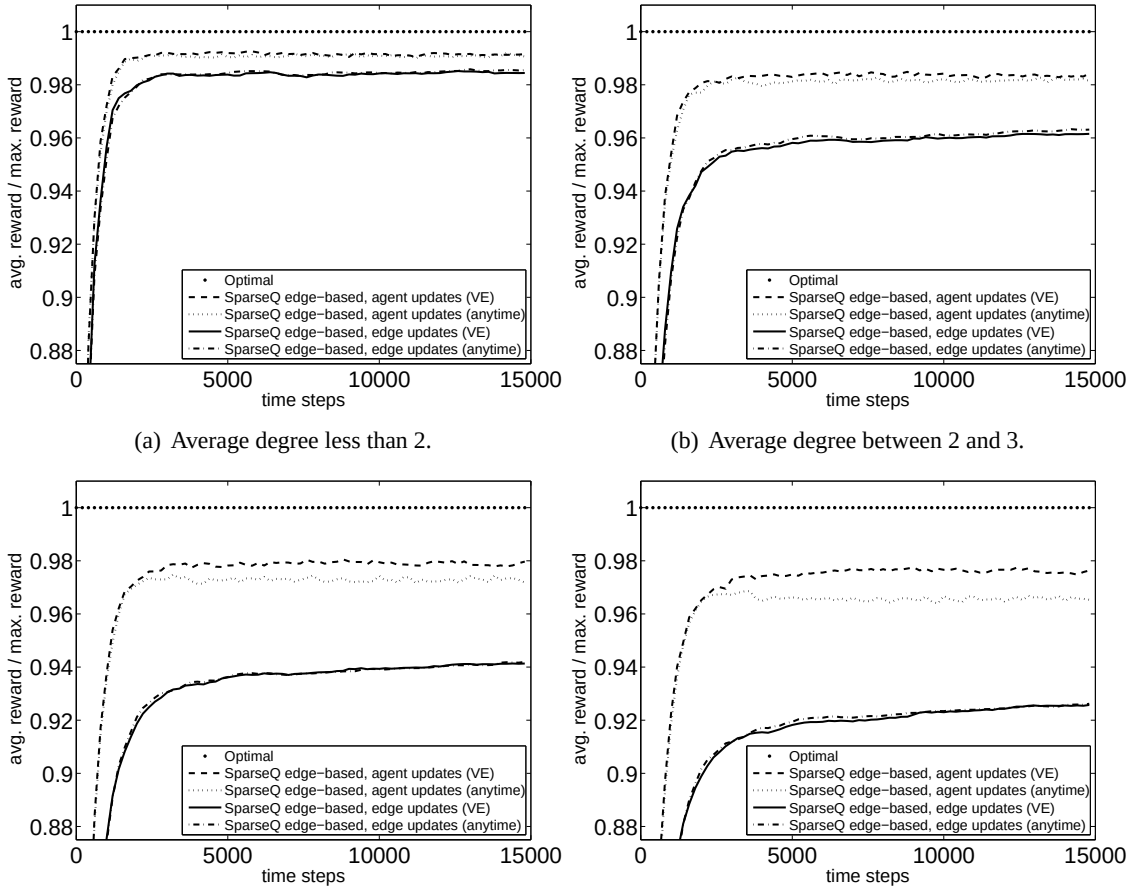
Figure 12: Running average, of the last 10 cycles, of the received reward relative to the optimal reward for different methods on the single-state, 12-agent problems. The legend of Fig. 12(a) holds for all figures. See text for the problem description.

each agent selects a value uniformly at random each iteration. The CoordRL approach updates each local Q-function with the global temporal-difference error. Therefore, the same global reward is propagated to each of the individual Q-functions and the expected future discounted return, that is, the sum of the local Q-functions, is overestimated. As a result the Q-values blow up, resulting in random behavior.

The IL/DVF approach learns a reasonable solution, but it suffers from the fact that each agent individually updates its Q-value irrespective of the actions performed by its neighbors. Therefore, the agents do not learn to coordinate and the policy keeps oscillating.

The random method keeps track of the best joint action found so far and slowly learns a better policy. However, it learns slower than the different SparseQ methods. Note that this method does not scale well to larger problems with more joint actions.

The agent-based SparseQ decomposition converges to an optimal policy since it stores a Q-value for every action combination of its neighbors, and is able to detect the best performing action



(a) Average degree less than 2.

(b) Average degree between 2 and 3.

(c) Average degree between 3 and 4.

(d) Average degree between 4 and 5.

Figure 13: Running average of the received reward relative to the optimal reward for the different edge-based methods, using either the VE or anytime max-plus algorithm, on the single-state, 12-agent problem.

combination. However, this approach learns slower than the different edge-based decompositions since it requires, as listed in Table 2, more samples to update the large number of Q-values. The two edge-based decompositions using the anytime extension both learn a near-optimal solution. The agent-based update method performs slightly better since it, indirectly, includes the neighboring Q-values in its update.

As is seen in Fig. 12(b), the results are similar for the more complicated graphs with an average degree between 2 and 3. Although not shown, the CoordRL learners are not able to learn a good policy and quickly stabilizes around 44% of the optimal value. On the other hand, the agent-based decomposition converges to the optimal policy. Although the final result is slightly worse compared to the simpler graphs, the edge-based decompositions still learn near-optimal policies. The result of the agent-based update method is better than the edge-based update method since the first includes the neighboring Q-values in its update.

method	$(1,2]$	$(2,3]$	$(3,4]$	$(4,5]$
Random with memory	0.9271	0.9144	0.9122	0.9104
IL	0.8696	0.8571	0.8474	0.8372
CoordRL	0.4113	0.4423	-	-
SparseQ agent (VE)	1.0000	0.9983	-	-
SparseQ edge, agent (VE)	0.9917	0.9841	0.9797	0.9765
SparseQ edge, edge (VE)	0.9843	0.9614	0.9416	0.9264
SparseQ edge, agent (anytime)	0.9906	0.9815	0.9722	0.9648
SparseQ edge, edge (anytime)	0.9856	0.9631	0.9419	0.9263

Table 3: Relative reward with respect to the optimal reward after $15,000$ cycles for the different methods and differently shaped graphs. Results are averaged over graphs with an average degree in $(x-1,x]$, as indicated by the column headers.

Similar results are also visible in Fig. 12(c) and Fig. 12(d). The agent-based decompositions are not applied to these graphs. As was already visible in Fig. 11, the algorithm needs too much time to process graphs of this complexity.

Fig. 13 compares the difference between using either the VE or the anytime max-plus algorithm to compute the joint action for the SparseQ methods using an edge-based decomposition. Fig. 13(a) and Fig. 13(b) show that the difference between the two approaches is negligible for the graphs with an average degree less than 3. However, for the more complex graphs (Fig. 13(c) and Fig. 13(d)) there is a small performance gain when the VE algorithm is used for the agent-based update method. The agent-based update method incorporates the neighboring Q-functions, and therefore the values of the Q-functions are less distinct. As a result, the max-plus algorithm has more difficulty in finding the optimal joint action. But note that, as was shown in Fig. 11, the VE algorithm requires substantially more computation time for graphs of this complexity than the anytime max-plus algorithm.

Although all results seem to converge, it is difficult to specify in which cases the proposed algorithms converge, and if so, whether they converge to an optimal solution. The difficulties arise from the fact that the reinforcement-learning algorithms deal with a double optimization: the computation of the optimal joint action with the maximal Q-value, and the global (long-term) optimization of the average discounted rewards. In this article we focus on the empirical results.

Table 3 gives an overview of all results and compares the value of the joint action corresponding to the learned strategy in cycle $15,000$ for the different methods. Although the results slowly decrease for the more complex reward functions, all SparseQ methods learn near-optimal policies. Furthermore, there is only a minimal difference between the methods that use the VE and the anytime max-plus algorithm to compute the joint action. For the densely connected graphs, the edge-based decompositions in combination with the max-plus algorithm are the only methods that are able to compute a good solution. The algorithms using VE fail to produce a result because of their inability to cope with the complexity of the underlying graph structure (see Section 3.2).

6.2 Experiments on a Distributed Sensor Network

We also perform experiments on a distributed sensor network (DSN) problem. This problem is a sequential decision-making variant of the distributed constraint optimization problem described by Ali et al. (2005), and was part of the NIPS 2005 benchmarking workshop.[6]

The DSN problem consists of two parallel chains of an arbitrary, but equal, number of sensors. The area between the sensors is divided into cells. Each cell is surrounded by exactly four sensors and can be occupied by a target. See Fig. 14(a) for a configuration with eight sensors and two targets. With equal probability a target moves to the cell on its left, to the cell on its right, or remains on its current position. Actions that move a target to an illegal position, that is, an occupied cell or a cell outside the grid, are not executed.

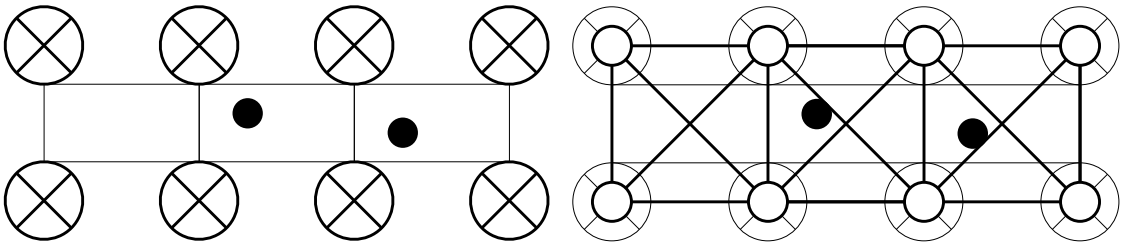
Each sensor is able to perform three actions: focus on a target in the cell to its immediate left, to its immediate right, or don't focus at all. Every focus action has a small cost modeled as a reward of -1. When in one time step at least three of the four surrounding sensors focus on a target, it is 'hit'. Each target starts with a default energy level of three. Each time a target is hit its energy level is decreased by one. When it reaches zero the target is captured and removed, and the three sensors involved in the capture each receive a reward of $+10$. In case four sensors are involved in a capture, only the three sensors with the highest index receive the reward. An episode finishes when all targets are captured.

As in the NIPS-05 benchmarking event, we will concentrate on a problem with eight sensors and two targets. This configuration results in $3^8 = 6,561$ joint actions and 37 distinct states, that is, 9 states for each of the 3 configurations with 2 targets, 9 for those with one target, and 1 for those without any targets. This problem thus has a large action space compared to its state space. When acting optimally, the sensors are able to capture both targets in three steps, resulting in a cumulative reward of 42. However, in order to learn this policy based on the received rewards, the agents have to cope with the delayed reward and learn how to coordinate their actions such that multiple targets are hit simultaneously.

In our experiments we generate all statistics using the benchmark implementation, with the following two differences. First, because the NIPS-05 implementation of the DSN problem only returns the global reward, we change the environment to return the individual rewards in order to comply to our model specification. Second, we set the fixed seed of the random number generator to a variable seed base on the current time in order to be able to perform varying runs.

We apply the different techniques described in Section 4 and Section 5 to the DSN problem. We do not apply the CoordRL approach, since, just as in the experiments in Section 6.1, it propagates back too much reward causing the individual Q-functions to blow up. However, we do apply the MDP learners approach which updates a Q-function based on the full joint action space. All applied methods learn for $10,000$ episodes which are divided into 200 episode blocks, each consisting of 50 episodes. The following statistics are computed at the end of each episode block: the average reward, that is, the undiscounted sum of rewards divided by the number of episodes in an episode block, the cumulative average reward of all previous episode blocks, and the wall-clock time. There is no distinction between learning and testing cycles, and the received reward thus includes exploration actions. The Q-learning methods all use the following parameters: $\alpha = 0.2$, $\varepsilon = 0.2$, and $\gamma = 0.9$, and start with zero-valued Q-values. We assume that both the DVF and the

[6] See `http://www.cs.rutgers.edu/~mlittman/topics/nips05-mdp/` for a detailed description of the benchmarking event and `http://rlai.cs.ualberta.ca/RLBB/` for the used RL-framework.



(a) Network configuration. (b) Network with corresponding CG.

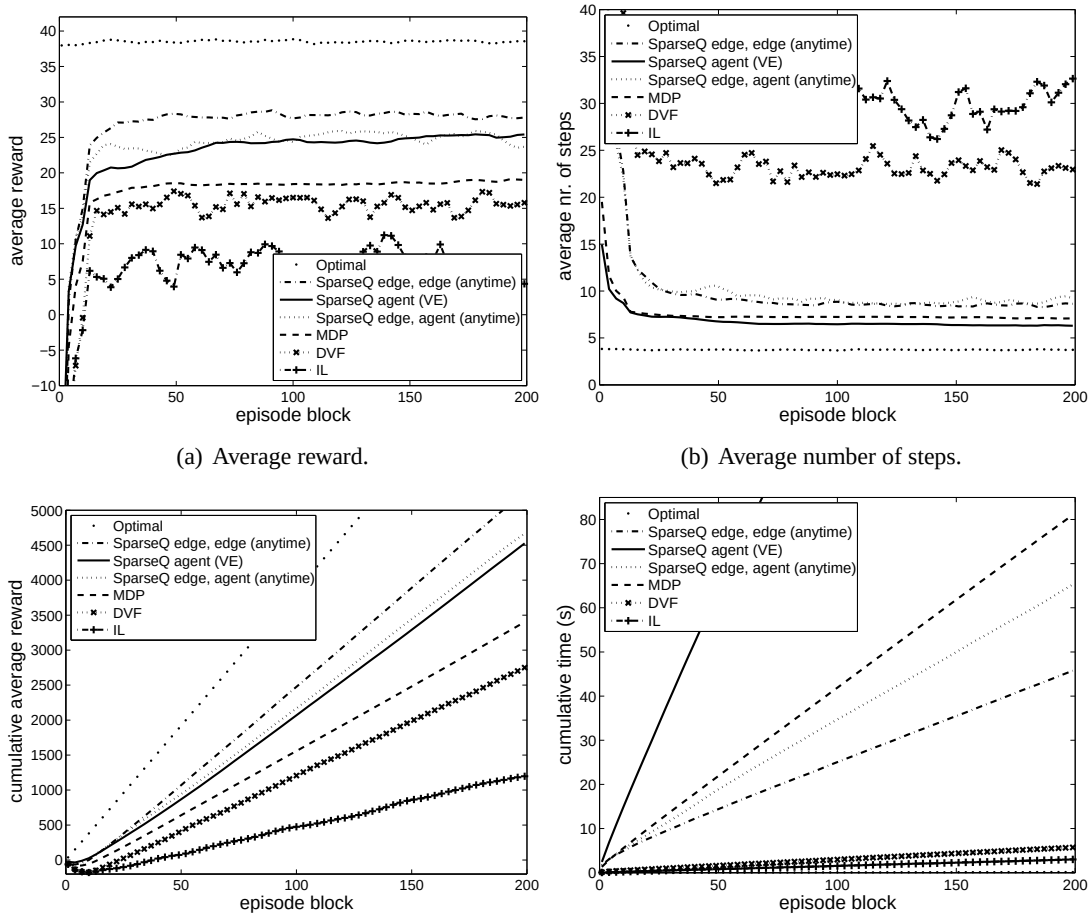
Figure 14: Fig. 14(a) shows an example sensor network with eight sensors ($\otimes$) and two targets ($\bullet$). Fig. 14(b) shows the corresponding CG representing the agent dependencies. The graph has an average degree of 4, and an induced width of 3.

different SparseQ variants have access to a CG which specifies for each agent on which other agents it depends. This CG is shown in Fig. 14(b), and has an average degree of 4.

The results, averaged over 10 runs with different random seeds, for the different techniques are shown in Fig. 15. The results contain exploration actions and are therefore not completely stable. For this reason, we show the running average over the last 10 episode blocks. Fig. 15(a) shows the average reward for the different approaches. The optimal policy is manually implemented and, in order to have a fair comparison with the other approaches, also includes random exploration actions with probability ε. It results in an average reward just below 40. The MDP approach settles to an average reward around 17 after a few episodes. Although this value is low compared to the result of the optimal policy, the MDP approach, as seen in Fig. 15(b), does learn to capture the targets in a small number of steps. From this we conclude that the low reward is mainly a result of unnecessary focus actions performed by the agents that are not involved in the actual capture. The MDP approach thus discovers one of the many possible joint actions that results in a capture of the target and the generation of a positive reward, and then exploits this strategy. However, the found joint action is non-optimal since one or more agents do not have to focus in order to capture the target. Because of the large action space and the delayed reward, it takes the MDP approach much more than $10,000$ episodes to learn that other joint actions result in a higher reward.

Although the DVF approach performs better than IL, both methods do not converge to a stable policy and keep oscillating. This is caused by the fact that both approaches store action values based on individual actions and therefore fail to select coordinated joint actions which are needed to capture the targets.

In the different SparseQ variants each agent stores and updates local Q-values. Since these are also based on the agent's neighbors in the graph, the agents are able to learn coordinated actions. Furthermore, the explicit coordination results in much more stable policies than the IL and DVF approach. The agent-based decomposition produces a slightly lower average reward than the edge-based decompositions, but, as shown in Fig. 15(b), it needs less steps to capture the targets. Identical to the MDP approach, the lower reward obtained by the agent-based decomposition is a consequence of the large action space involved in each local term. As a result the agents are able to quickly learn a good policy that captures the targets in a few steps, but it takes a long time to converge to a joint action that does not involve the unnecessary focus actions of some of the agents. For example, each of the four agents in the middle of the DSN coordinates with 5 other agents, and



(a) Average reward.

(b) Average number of steps.

(c) Cumulative average reward.

(d) Cumulative time.

Figure 15: Different results on the DSN problem, averaged over 10 runs. One run consists of 200 episode blocks, each corresponding to 50 learning episodes.

each of them thus stores a Q-function defined over $3^6 = 729$ actions per state. Because in the agent-based decomposition the full action space is decomposed into different independent local action values, it does result in a better performance than the MDP learners, both in the obtained average reward and the number of steps needed to capture the targets. With respect to the two edge-based decompositions, the edge-based update method generates a slightly higher reward, and a more stable behavior than the agent-based update method. Although in both cases the difference between the two methods is minimal, this result is different compared to the stateless problems described in Section 6.1 in which the agent-based update method performed better. The effectiveness of each approach thus depends on the type of problem. We believe that the agent-based update method has its advantages for problems with fine-grained agent interactions since it combines all neighbors in the update of the Q-value.

Fig. 15(c) shows the cumulative average reward of the different methods. Ignoring the manual policy, the edge-based update methods result in the highest cumulative average reward. This is also

method	reward	steps	method	reward	steps
Optimal	38.454	3.752	SparseQ edge, edge (anytime)	27.692	8.795
MDP	19.061	7.071	SparseQ edge, edge (VE)	28.880	8.113
DVF	16.962	22.437	SparseQ agent (VE)	24.844	6.378
IL	6.025	31.131	SparseQ edge, agent (VE)	25.767	8.413
			SparseQ edge, agent (anytime)	23.738	8.930

Table 4: Average reward and average number of steps per episode over the last 2 episode blocks (100 episodes) for the DSN problem. Results are averaged over 10 runs.

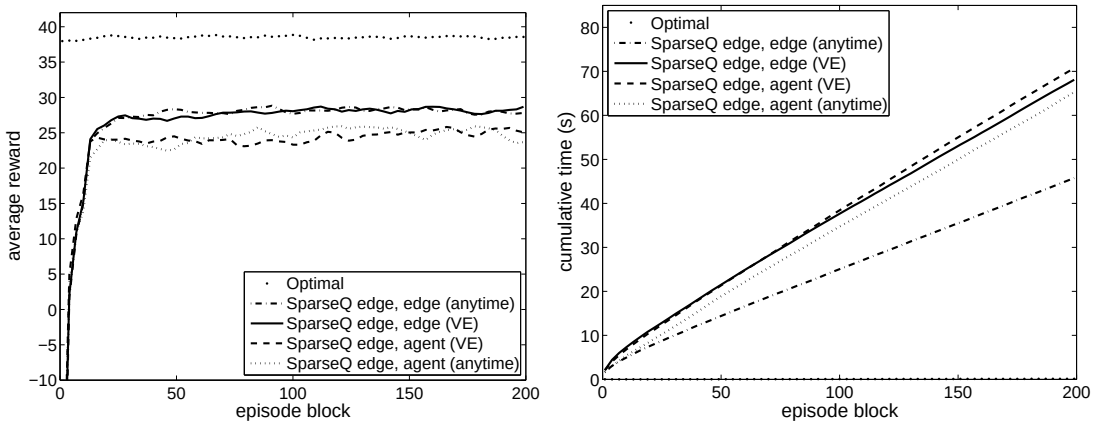
seen in Table 4 which shows the reward and the number of steps per episode averaged over the last 2 episode blocks, that is, 100 episodes, for the different methods. Since the goal of the agents is to optimize the received average reward, the SparseQ methods outperform the other learning methods. However, none of the variants converge to the optimal policy. One of the main reasons is the large number of dependencies between the agents. This requires a choice between an approach that models many of the dependencies but learns slowly because of the exploration of a large action space, for example, the agent-based SparseQ or the MDP learners, or an approach that ignores some of the dependencies but is able to learn an approximate solution quickly. The latter is the approach taken by the edge-based SparseQ variants: it models pairwise dependencies even though it requires three agents to capture a target.

Fig. 15(d) gives the timing results for the different methods. The IL and DVF methods are the fastest methods since they only store and update individual Q-values. The agent-based SparseQ method is by far the slowest. This method stores a Q-function based on all action combinations of an agent and its neighbors in the CG. This slows down the VE algorithm considerably since it has to maximize over a large number of possible joint action combinations in every local maximization step.

Finally, Fig. 16 compares the difference between using the VE or the anytime max-plus algorithm to compute the joint action for the SparseQ methods using an edge-based decomposition. Fig. 16(a) shows that there is no significant difference in the obtained reward for these two methods. Fig. 16(b) shows that the edge-based SparseQ variants that use the anytime max-plus algorithm need less computation time than those using the VE algorithm. However, the differences are not that evident as in the experiments from Section 6.1 because the used CG has a relative simple structure (it has an induced width of 3), and VE is able to quickly find a solution when iteratively eliminating the nodes with the smallest degree.

7. Conclusion and Future Directions

In this article we addressed the problem of learning how to coordinate the behavior of a large group of agents. First, we described a payoff propagation algorithm (max-plus) that can be used as an alternative to variable elimination (VE) for finding the optimal joint action in a coordination graph (CG) with predefined payoff functions. VE is an exact method that will always report the joint action that maximizes the global payoff, but it is slow for densely connected graphs with cycles because its worst-case complexity is exponential in the number of agents. Furthermore, this method



(a) Average reward. (b) Cumulative time.

Figure 16: Results of the edge-based decomposition methods on the DSN problem, averaged over 10 runs. One run consists of 200 episode blocks, each corresponding to 50 learning episodes.

is only able to report a solution after the complete algorithm has ended. The max-plus algorithm, analogous to the belief propagation algorithm in Bayesian networks, operates by repeatedly sending local payoff messages over the edges in the CG. By performing a local computation based on its incoming messages, each agent is able to select its individual action. For tree-structured graphs, this approach results in the optimal joint action. For large, highly connected graphs with cycles, we provided empirical evidence that this method can find near-optimal solutions exponentially faster than VE. Another advantage of the max-plus algorithm is that it can be implemented fully distributed using asynchronous and parallel message passing.

Second, we concentrated on model-free reinforcement-learning approaches to learn the coordinated behavior of the agents in a collaborative multiagent system. In our Sparse Cooperative Q-learning (SparseQ) methods, we approximate the global Q-function using a CG representing the coordination requirements of the system. We analyzed two possible decompositions, one in terms of the nodes and one in terms of the edges of the graph. During learning, each local Q-function is updated based on its contribution to the maximal global payoff found with either the VE or max-plus algorithm. Effectively, each agent learns its part of the global solution by only coordinating with the agents on which it depends. Results on both a single-state problem with 12 agents and more than 17 million actions, and a distributed sensor network problem show that our SparseQ variants outperform other existing multiagent Q-learning methods. The combination of the edge-based decomposition and the max-plus algorithm results in a method which scales only linearly in the number of dependencies of the problem. Furthermore, it can be implemented fully distributed and only requires that each agent is able to communicate with its neighbors in the graph. When communication is restricted, it is still possible to run the algorithm when additional common knowledge assumptions are made.

There are several directions for future work. First of all, we are interested in comparing different approximation alternatives from the Bayesian networks or constraint processing literature

to our max-plus algorithm. A natural extension is to consider factor graph representations of the problem structure (Kschischang et al., 2001), allowing more prior knowledge about the problem to be introduced beforehand. Another possible direction involves the 'mini-bucket' approach, an approximation in which the VE algorithm is simplified by changing the full maximization for each elimination of an agent to the summation of simpler local maximizations (Dechter and Rish, 1997). A different alternative for the VE algorithm is the usage of constraint propagation algorithms for finding the optimal joint action (Modi et al., 2005). Another interesting issue is related to the Q-updates of the edge-based decomposition of the SparseQ reinforcement-learning method. Now we assume that the received reward of an agent is divided proportionally over its edges (see (20) and (23)), but other schemes may also be possible. Furthermore, we like to apply our method to problems in which the topology of the CG differs per state, for example, when agents are dynamically added or removed from the system, or dependencies between the agents change based on the current situation (Guestrin et al., 2002c). Since all Q-functions and updates are defined locally, it is possible to compensate the addition or removal of an agent by redefining only the Q-functions in which this agent is involved. The max-plus algorithm and the local updates do not have to be changed as long as the neighboring agents are aware of the new topology of the CG.

Acknowledgments

We would like to thank Carlos Guestrin and Ron Parr for providing ample feedback to this work. Furthermore, we like to thank all three reviewers for their detailed and constructive comments. This research is supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414.

References

S. Muhammad Ali, S. Koenig, and M. Tambe. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1041–1048, Utrecht, The Netherlands, 2005.

T. Arai, E. Pagello, and L. E. Parker. Editorial: Advances in multi-robot systems. *IEEE Transactions on Robotics and Automation*, 18(5):665–661, 2002.

R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-independent decentralized Markov decision processes. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Melbourne, Australia, 2003.

D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov decision processes. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, Stanford, CA, 2000.

U. Bertelé and F. Brioschi. *Nonserial dynamic programming*. Academic Press, 1972.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the Conference on Theoretical Aspects of Rationality and Knowledge*, 1996.

J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems (NIPS) 6*, pages 671–678. Morgan Kaufmann Publishers, Inc., 1994.

W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A Bayesian approach. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 709–716, Melbourne, Australia, 2003. ACM Press.

C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Madison, WI, 1998.

C. Crick and A. Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2003.

R. Crites and A. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS) 8*, pages 1017–1023. MIT Press, 1996.

R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

R. Dechter and I. Rish. A scheme for approximating probabilistic inference. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, pages 132–141, 1997.

E. H. Durfee. Scaling up agent coordination strategies. *IEEE Computer*, 34(7):39–46, July 2001.

P. S. Dutta, N. R. Jennings, and L. Moreau. Cooperative information sharing to improve distributed learning in multi-agent systems. *Journal of Artificial Intelligence Research*, 24:407–463, 2005.

C. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, November 2004.

C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 137–144, New York, NY, USA, 2003. ACM Press.

C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems (NIPS) 14*. The MIT Press, 2002a.

C. Guestrin. *Planning under uncertainty in complex structured environments*. PhD thesis, Computer Science Department, Stanford University, August 2003.

C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *International Conference on Machine Learning (ICML)*, Sydney, Australia, July 2002b.

C. Guestrin, S. Venkataraman, and D. Koller. Context-specific multiagent coordination and planning with factored MDPs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Edmonton, Canada, July 2002c.

E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Jose, CA, 2004.

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The robot world cup initiative. In *Proceedings of the IJCAI-95 Workshop on Entertainment and AI/Alife*, 1995.

J. R. Kok and N. Vlassis. Sparse cooperative Q-learning. In Russ Greiner and Dale Schuurmans, editors, *Proceedings of the International Conference on Machine Learning*, pages 481–488, Banff, Canada, July 2004. ACM.

J. R. Kok and N. Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *RoboCup-2005: Robot Soccer World Cup IX*, Osaka, Japan, July 2005.

J. R. Kok, M. T. J. Spaan, and N. Vlassis. Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 50(2-3):99–114, February 2005.

J. R. Kok. *Coordination and Learning in Cooperative Multiagent Systems.* PhD thesis, Faculty of Science, University of Amsterdam, 2006.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 2001.

V. Lesser, C. Ortiz, and M. Tambe. *Distributed sensor nets: A multiagent perspective.* Kluwer academic publishers, 2003.

H.-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, pages 28–41, January 2004.

C. C. Moallemi and B. Van Roy. Distributed optimization in adaptive networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS) 16.* MIT Press, Cambridge, MA, 2004.

P. Jay Modi, W-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, Stockholm, Sweden, 1999.

A. Y. Ng, H. Jin Kim, M. Jordan, and S. Sastry. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS) 16*, 2004.

L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3):231–255, 2002.

J. Pearl. *Probabilistic reasoning in intelligent systems.* Morgan Kaufman, San Mateo, 1988.

L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, pages 489–496. Morgan Kaufmann Publishers, 2000.

M. L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley, New York, 1994.

D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller. Distributed value functions. In *International Conference on Machine Learning (ICML)*, Bled, Slovenia, 1999.

S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Seattle, WA, 1994.

L. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39:1095–1100, 1953.

P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, 1998.

K. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning (ICML)*, Amherst, MA, 1993.

G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3), March 1995.

N. Vlassis. A concise introduction to multiagent systems and distributed AI. Informatics Institute, University of Amsterdam, September 2003.

N, Vlassis, R. Elhorst, and J. R. Kok. Anytime algorithms for multiagent decision making using coordination graphs. In *Proceedings of the International Conference on Systems, Man, and Cybernetics (SMC)*, The Hague, The Netherlands, October 2004.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14: 143–166, April 2004.

C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

G. Weiss, editor. *Multiagent systems: A modern approach to distributed artificial intelligence*. MIT Press, 1999.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Exploring Artificial Intelligence in the New Millennium*, chapter 8, pages 239–269. Morgan Kaufmann Publishers Inc., January 2003.

M. Yokoo and E. H. Durfee. Distributed constraint optimization as a formal model of partially adversarial cooperation. Technical Report CSE-TR-101-91, University of Michigan, Ann Arbor, MI 48109, 1991.

N. Lianwen Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

Journal of Machine Learning Research 7 (2006) 1829-1859 Submitted 5/06; Revised 8/06; Published 9/06

Estimating the "Wrong" Graphical Model: Benefits in the Computation-Limited Setting

Martin J. Wainwright WAINWRIG@STAT.BERKELEY.EDU
Department of Statistics
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720, USA

Editor: Max Chickering

Abstract

Consider the problem of joint parameter estimation and prediction in a Markov random field: that is, the model parameters are estimated on the basis of an initial set of data, and then the fitted model is used to perform prediction (e.g., smoothing, denoising, interpolation) on a new noisy observation. Working under the restriction of limited computation, we analyze a joint method in which the *same convex variational relaxation* is used to construct an M-estimator for fitting parameters, and to perform approximate marginalization for the prediction step. The key result of this paper is that in the computation-limited setting, using an inconsistent parameter estimator (i.e., an estimator that returns the "wrong" model even in the infinite data limit) is provably beneficial, since the resulting errors can partially compensate for errors made by using an approximate prediction technique. En route to this result, we analyze the asymptotic properties of M-estimators based on convex variational relaxations, and establish a Lipschitz stability property that holds for a broad class of convex variational methods. This stability result provides additional incentive, apart from the obvious benefit of unique global optima, for using message-passing methods based on convex variational relaxations. We show that joint estimation/prediction based on the reweighted sum-product algorithm substantially outperforms a commonly used heuristic based on ordinary sum-product.

Keywords: graphical model, Markov random field, belief propagation, variational method, parameter estimation, prediction error, algorithmic stability

1. Introduction

Graphical models such as Markov random fields (MRFs) are widely used in many application domains, including machine learning, natural language processing, statistical signal processing, and communication theory. A fundamental limitation to their practical use is the difficulty associated with computing various statistical quantities (e.g., marginals, data likelihoods etc.); such quantities are of interest both Bayesian and frequentist settings. Sampling-based methods, especially those of the Markov chain Monte Carlo (MCMC) variety (Liu, 2001; Robert and Casella, 1999), represent one approach to obtaining stochastic approximations to marginals and likelihoods. A possible disadvantage of sampling methods is their relatively high computational cost. It is thus of considerable interest for various application domains to consider less computationally intensive methods for generating approximations to marginals, log likelihoods, and other relevant statistical quantities.

Variational methods are one class of techniques that can be used to At the foundation of these methods is the fact that for a broad class of MRFs, the computation of the log likelihood and

marginal probabilities can be reformulated as a convex optimization problem; see Yedidia (2001) or Wainwright and Jordan (2003) for overviews. Although this optimization problem is intractable to solve exactly for general MRFs, it suggests a principled route to obtaining approximations—namely, by relaxing the original optimization problem, and taking the optimal solutions to the relaxed problem as approximations to the exact values. In many cases, optimization of the relaxed problem can be carried out by "message-passing" algorithms, in which neighboring nodes in the Markov random field convey statistical information (e.g., likelihoods) by passing functions or vectors (referred to as messages). Well-known examples of such variational methods include mean field algorithms, the belief propagation or sum-product algorithm, as well as various extensions including generalized belief propagation and expectation propagation.

Estimating the parameters of a Markov random field from data poses another significant challenge. A direct approach—for instance, via (regularized) maximum likelihood estimation—entails evaluating the cumulant generating (or log partition) function, which is computationally intractable for general Markov random fields. One viable option is the pseudolikelihood method (Besag, 1975, 1977), which can be shown to produce consistent parameter estimates under suitable assumptions, though with an associated loss of statistical efficiency. Other researchers have studied algorithms for ML estimation based on stochastic approximation (Younes, 1988; Benveniste et al., 1990), which again are consistent under appropriate assumptions, but can be slow to converge.

1.1 Overview

As illustrated in Figure 1, the problem domain of interest in this paper is that of joint estimation and prediction in a Markov random field. More precisely, given samples $\{X^1, \ldots, X^n\}$ from some unknown underlying model $p(\cdot\,; \theta^*)$, the first step is to form an estimate of the model parameters. Now suppose that we are given a noisy observation of a new sample path $Z \sim p(\cdot\,; \theta^*)$, and that we wish to form a (near)-optimal estimate of Z using the fitted model, and the noisy observation (denoted Y). Examples of such prediction problems include signal denoising, interpolation of missing data, and sentence parsing. Disregarding any issues of computational cost and speed, one could proceed via Route A in Figure 1—that is, one could envisage first using a standard technique (e.g., regularized maximum likelihood) for parameter estimation, and then carrying out the prediction step (which might, for instance, involve computing certain marginal probabilities) by Monte Carlo methods.

This paper, in contrast, is concerned with the *computation-limited* setting, in which both sampling or brute force methods are overly intensive. With this motivation, a number of researchers have studied the use of approximate message-passing techniques, both for problems of prediction (Heskes et al., 2003; Ihler et al., 2005; Minka, 2001; Mooij and Kappen, 2005b; Tatikonda, 2003; Wainwright et al., 2003a; Wiegerinck, 2005; Yedidia et al., 2005) as well as for parameter estimation (Leisink and Kappen, 2000; Sutton and McCallum, 2005; Teh and Welling, 2003; Wainwright et al., 2003b). However, despite their wide-spread use, the theoretical understanding of such message-passing techniques remains limited[1], especially for parameter estimation. Consequently, it is of considerable interest to characterize and quantify the loss in performance incurred by using computationally tractable methods versus exact methods (i.e., Route B versus A in Figure 1). More

1. The behavior of sum-product is relatively well understood in certain settings, including graphs with single cycles (Weiss, 2000), Gaussian models (Freeman and Weiss, 2001; Rusmevichientong and Roy, 2000) and Ising models (Tatikonda and Jordan, 2002; Ihler et al., 2005; Mooij and Kappen, 2005a). Similarly, there has been substantial progress for graphs with high girth (Richardson and Urbanke, 2001), but much of this analysis breaks down in application to graphs with short cycles.

specifically, our analysis applies to variational methods that are based on *convex relaxations*. This class includes a number of existing methods—among them the tree-reweighted sum-product algorithm (Wainwright et al., 2005), reweighted forms of generalized belief propagation (Wiegerinck, 2005), and semidefinite relaxations (Wainwright and Jordan, 2005). Moreover, it is possible to modify other variational methods—for instance, expectation propagation (Minka, 2001)—so as to "convexify" them.

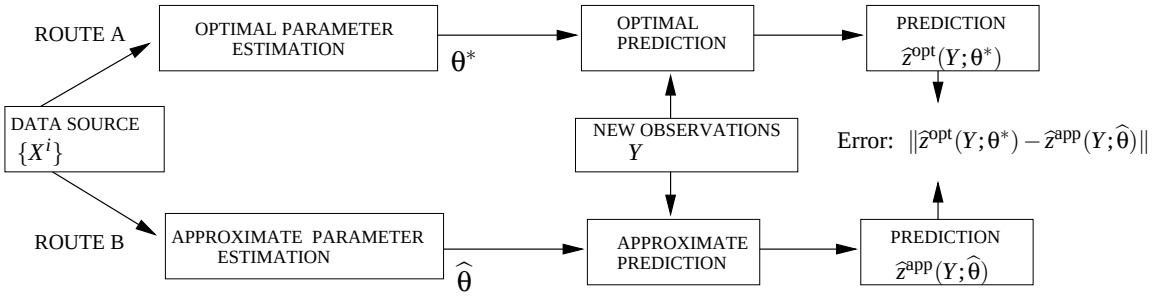


Figure 1: Route A: computationally intractable combination of parameter estimation and prediction. Route B: computationally efficient combination of approximate parameter estimation and prediction.

1.2 Our Contributions

At a high level, the key idea of this paper is the following: given that approximate methods can lead to errors at both the estimation and prediction phases, it is natural to speculate that these sources of error might be arranged to partially cancel one another. The theoretical analysis of this paper confirms this intuition: we show that with respect to end-to-end performance, it is in fact beneficial, even in the infinite data limit, to learn the "wrong" the model by using *inconsistent* methods for parameter estimation. En route to this result, we analyze the asymptotic properties of M-estimators based on convex variational relaxations, and establish a Lipschitz stability property that holds for a broad class of variational methods. Such global algorithmic stability is a fundamental concern given statistical models imperfectly estimated from limited data, or for applications in which "errors" may be introduced into message-passing (e.g., due to quantization or other forms of communication constraints in sensor networks). Thus, our global stability result provides further theoretical justification—apart from the obvious benefit of unique global optima—for using message-passing methods based on convex variational relaxations. Finally, we provide some empirical results to show that joint estimation/prediction based on the reweighted sum-product algorithm substantially outperforms a commonly used heuristic based on ordinary sum-product.

The remainder of this paper is organized as follows. Section 2 provides background on Markov random fields. In Section 3, we introduce background on variational representations, including the notion of a convex surrogate to the cumulant generating function, and then illustrate this notion via the tree-reweighted Bethe approximation (Wainwright et al., 2005). In Section 4, we describe how any convex surrogate defines a particular joint scheme for parameter estimation and prediction. Section 5 provides results on the asymptotic behavior of the estimation step, as well as the stability of the prediction step. Section 6 is devoted to the derivation of performance bounds for joint estimation

and prediction methods, with particular emphasis on the mixture-of-Gaussians observation model. In Section 7, we provide experimental results on the performance of a joint estimation/prediction method based on the tree-reweighted Bethe surrogate, and compare it to a heuristic method based on the ordinary belief propagation algorithm. We conclude in Section 8 with a summary and discussion of directions for future work.

2. Background

We begin with background on Markov random fields. Consider an undirected graph $G = (V, E)$, consisting of a set of vertices $V = \{1, \ldots, N\}$ and an edge set E. We associate to each vertex $s \in V$ a multinomial random variable X_s taking values in the set $\mathcal{X}_s = \{0, 1, \ldots, m-1\}$. We use the lower case letter x_s to denote particular realizations of the random variable X_s in the set $\mathcal{X}_s$. This paper makes use of the following exponential representation of a pairwise Markov random field over the multinomial random vector $X := \{X_s,\ s \in V\}$. We begin by defining, for each $j = 1, \ldots, m-1$, the $\{0,1\}$-valued indicator function

$$\mathbb{I}_j[x_s] \quad := \quad \begin{cases} 1 & \text{if } x_s = j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

These indicator functions can be used to define a potential function $\theta_s(\cdot) : \mathcal{X}_s \to \mathbb{R}$ via

$$\theta_s(x_s) \quad := \quad \sum_{j=1}^{m-1} \theta_{s;j} \mathbb{I}_j[x_s] \tag{2}$$

where $\theta_s = \{\theta_{s;j}, j = 1, \ldots, m-1\}$ is the vector of exponential parameters associated with the potential. Our exclusion of the index $j = 0$ is deliberate, so as to ensure that the collection of indicator functions $\phi_s(x_s) := \{\mathbb{I}_j[x_s],\ j = 1, \ldots, m-1\}$ remain affinely independent. In a similar fashion, we define for any pair $(s,t) \in E$ the pairwise potential function

$$\theta_{st}(x_s, x_t) \quad := \quad \sum_{j=1}^{m-1} \sum_{k=1}^{m-1} \theta_{st;jk} \mathbb{I}_j[x_s]\, \mathbb{I}_k[x_t],$$

where we use $\theta_{st} := \{\theta_{st;jk},\ j,k = 1, 2, \ldots, m-1\}$ to denote the associated collection of exponential parameters, and $\phi_{st}(x_s, x_t) := \{\mathbb{I}_j[x_s]\mathbb{I}_k[x_s],\ j,k = 1, 2, \ldots, m-1\}$ for the associated set of sufficient statistics.

Overall, the probability mass function of the multinomial Markov random field in exponential form can be written as

$$p(x; \theta) \quad = \quad \exp\Big\{ \sum_{s \in V} \theta_s(x_s) + \sum_{(s,t) \in E} \theta_{st}(x_s, x_t) - A(\theta) \Big\}. \tag{3}$$

Here the function

$$A(\theta) \quad := \quad \log \Big[\sum_{x \in \mathcal{X}^N} \exp\Big\{ \sum_{s \in V} \theta_s(x_s) + \sum_{(s,t) \in E} \theta_{st}(x_s, x_t) \Big\} \Big] \tag{4}$$

is the logarithm of the normalizing constant associated with $p(\cdot; \theta)$.

The collection of distributions thus defined can be viewed as a regular and minimal exponential family (Brown, 1986). In particular, the exponential parameter θ and the vector of sufficient statistics ϕ are formed by concatenating the exponential parameters (respectively indicator functions) associated with each vertex and edge—viz.

$$\begin{aligned} \theta &= \{\theta_s, s \in V\} \cup \{\theta_{st},\ (s,t) \in E\} \\ \phi(x) &= \{\phi_s(x_s), s \in V\} \cup \{\phi_{st}(x_s, x_t),\ (s,t) \in E\} \end{aligned}$$

This notation allows us to write Equation (3) more compactly as $p(x;\theta) = \exp\{\langle \theta, \phi(x)\rangle - A(\theta)\}$. A quick calculation shows that $\theta \in \mathbb{R}^d$, where $d = N(m-1) + |E|\,(m-1)^2$ is the dimension of this exponential family.

The following properties of A are well-known:

Lemma 1 *(a) The function A is convex, and strictly so when the sufficient statistics are affinely independent.*
(b) It is an infinitely differentiable function, with derivatives corresponding to cumulants. In particular, for any indices $\alpha, \beta \in \{1, \ldots, d\}$, we have

$$\frac{\partial A}{\partial \theta_\alpha} = \mathbb{E}_\theta[\phi_\alpha(X)], \qquad \frac{\partial^2 A}{\partial \theta_\alpha \partial \theta_\beta} = \mathrm{cov}_\theta\{\phi_\alpha(X), \phi_\beta(X)\},$$

where $\mathbb{E}_\theta$ and cov_θ denote the expectation and covariance respectively.

We use $\mu \in \mathbb{R}^d$ to denote the vector of *mean parameters* defined element-wise by $\mu_\alpha = \mathbb{E}_\theta[\phi_\alpha(X)]$ for any $\alpha \in \{1, \ldots, d\}$. A convenient property of the sufficient statistics ϕ defined in Equations (1) and (2) is that these mean parameters correspond to marginal probabilities. For instance, when $\alpha = (s; j)$ or $\alpha = (st; jk)$, we have respectively

$$\mu_{s;j} = \mathbb{E}_\theta[\mathbb{I}_j[x_s]] = p(X_s = j; \theta), \quad \text{and} \tag{5a}$$

$$\mu_{st;jk} = \mathbb{E}_\theta\big\{\mathbb{I}_j[x_s]\,\mathbb{I}_k[x_t]\big\} = p(X_s = j, X_t = k; \theta). \tag{5b}$$

3. Construction of Convex Surrogates

This section is devoted to a systematic procedure for constructing convex functions that represent approximations to the cumulant generating function. We begin with a quick development of an exact variational principle, one which is intractable to solve in general cases; see the papers (Pietra et al., 1997; Wainwright and Jordan, 2005) for further details. Nonetheless, this exact variational principle is useful, in that various natural relaxations of the optimization problem can be used to define convex surrogates to the cumulant generating function. After a high-level description of such constructions in general, we then illustrate it more concretely with the particular case of the "convexified" Bethe entropy (Wainwright et al., 2005).

3.1 Exact Variational Representation

Since A is a convex and continuous function (see Lemma 1), the theory of convex duality (Rockafellar, 1970) guarantees that it has a variational representation, given in terms of its conjugate dual function $A^* : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, of the following form

$$A(\theta) = \sup_{\mu \in \mathbb{R}^d} \big\{\theta^T \mu - A^*(\mu)\big\}.$$

In order to make effective use of this variational representation, it remains determine the form of the dual function. A useful fact is that the exponential family (3) arises naturally as the solution of an entropy maximization problem. In particular, consider the set of linear constraints

$$\mathbb{E}_p[\phi(X)] := \sum_{x \in \mathcal{X}^N} p(x)\phi_\alpha(x) \quad = \quad \mu_\alpha \quad \text{for } \alpha = 1, \ldots, d, \tag{6}$$

where $\mu \in \mathbb{R}^d$ is a set of target mean parameters. Letting $\mathcal{P}$ denote the set of all probability distributions with support on $\mathcal{X}^N$, consider the *constrained entropy maximization problem*: maximize the entropy $H(p) := -\sum_{x \in \mathcal{X}^N} p(x) \log p(x)$ subject to the constraints (6).

A first question is when there any distributions p that satisfy the constraints (6). Accordingly, we define the set

$$\text{MARG}_\phi(G) \quad := \quad \left\{ \mu \in \mathbb{R}^d \,\middle|\, \mu = \mathbb{E}_p[\phi(X)] \quad \text{for some } p \in \mathcal{P} \right\},$$

corresponding to the set of μ for which the constraint set (6) is non-empty. For any $\mu \notin \text{MARG}_\phi(G)$, the optimal value of the constrained maximization problem is $-\infty$ (by definition, since the problem is infeasible). Otherwise, it can be shown that the optimum is attained at a unique distribution in the exponential family, which we denote by $p(\cdot\,; \theta(\mu))$. Overall, these facts allow us to specify the conjugate dual function as follows:

$$A^*(\mu) \quad = \quad \begin{cases} -H(p(\cdot\,; \theta(\mu))) & \text{if } \mu \in \text{MARG}_\phi(G) \\ +\infty & \text{otherwise.} \end{cases} \tag{7}$$

See the technical report (Wainwright and Jordan, 2003) for more details of this dual calculation. With this form of the dual function, we are guaranteed that the cumulant generating function A has the following variational representation:

$$A(\theta) \quad = \quad \max_{\mu \in \text{MARG}_\phi(G)} \left\{ \theta^T \mu - A^*(\mu) \right\}. \tag{8}$$

However, in general, solving the variational problem (8) is intractable. This intractability should not be a surprise, since the cumulant generating function is intractable to compute for a general graphical model. The difficulty arises from two sources. First, the *constraint set* $\text{MARG}_\phi(G)$ is extremely difficult to characterize exactly for a general graph with cycles. For the case of a multinomial Markov random field (3), it can be seen (using the Minkowski-Weyl theorem) that $\text{MARG}_\phi(G)$ is a polytope, meaning that it can be characterized by a finite number of linear constraints. The question, of course, is how rapidly this number of constraints grows with the number of nodes N in the graph. Unless certain fundamental conjectures in computational complexity turn out to be false, this growth must be non-polynomial; see Deza and Laurent (1997) for an in-depth discussion of the binary case. Tree-structured graphs are a notable exception, for which the junction tree theory (Lauritzen, 1996) guarantees that the growth is only linear in N.

Second, the *dual function* A^* lacks a closed-form representation for a general graph. Note in particular that the representation (7) is not explicit, since it requires solving a constrained entropy maximization problem in order to compute the value $H(p(\cdot\,; \theta(\mu)))$. Again, important exceptions to this rule are tree-structured graphs. Here a special case of the junction tree theory guarantees

that any Markov random field on a tree $T = (V, E(T))$ can be factorized in terms of its marginals as follows

$$p(x; \theta(\mu)) \quad = \quad \prod_{s \in V} \mu_s(x_s) \prod_{(s,t) \in E(T)} \frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)}. \tag{9}$$

Consequently, in this case, the negative entropy (and hence the dual function) can be computed explicitly as

$$-A^*(\mu; T) \quad = \quad \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E(T)} I_{st}(\mu_{st}) \tag{10}$$

where $H_s(\mu_s) := -\sum_{x_s} \mu_s(x_s) \log \mu_s(x_s)$ and $I_{st}(\mu_{st}) := \sum_{x_s, x_t} \mu_{st}(x_s, x_t) \log \frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)}$ are the singleton entropy and mutual information, respectively, associated with the node $s \in V$ and edge $(s,t) \in E(T)$. For a general graph with cycles, in contrast, the dual function lacks such an explicit form, and is not easy to compute.

Given these challenges, it is natural to consider approximations to A^* and $\mathrm{MARG}_\phi(G)$. As we discuss in the following section, the resulting relaxed optimization problem defines a convex surrogate to the cumulant generating function.

3.2 Convex Surrogates to the Cumulant Generating Function

We now describe a general procedure for constructing convex surrogates to the cumulant generating function, consisting of two main ingredients. Given the intractability of characterizing the marginal polytope $\mathrm{MARG}_\phi(G)$, it is natural to consider a relaxation. More specifically, let $\mathrm{REL}_\phi(G)$ be a convex and compact set that acts as an outer bound to $\mathrm{MARG}_\phi(G)$. We use τ to denote elements of $\mathrm{REL}_\phi(G)$, and refer to them as *pseudomarginals* since they represent relaxed versions of local marginals. The second ingredient is designed to sidestep the intractability of the dual function: in particular, let B^* be a strictly convex and twice continuously differentiable approximation to A^*. We require that the domain of B^* (i.e., $\mathrm{dom}(B^*) := \{\tau \in \mathbb{R}^d \,|\, B^*(\tau) < +\infty\}$) be contained within the relaxed constraint set $\mathrm{REL}_\phi(G)$.

By combining these two approximations, we obtain a convex surrogate B to the cumulant generating function, specified via the solution of the following relaxed optimization problem

$$B(\theta) \quad := \quad \max_{\tau \in \mathrm{REL}_\phi(G)} \left\{ \theta^T \tau - B^*(\tau) \right\}. \tag{11}$$

Note the parallel between this definition (11) and the variational representation of A in Equation (8).

The function B so defined has several desirable properties, as summarized in the following proposition:

Proposition 2 *Any convex surrogate B defined via* (11) *has the following properties:*

(i) For each $\theta \in \mathbb{R}^d$, the optimum defining B is attained at a unique point $\tau(\theta)$.

(ii) The function B is convex on $\mathbb{R}^d$.

(iii) It is differentiable on $\mathbb{R}^d$, and more specifically:

$$\nabla B(\theta) \quad = \quad \tau(\theta).$$

Proof (i) By construction, the constraint set $\mathrm{REL}_\phi(G)$ is compact and convex, and the function B^* is strictly convex, so that the optimum is attained at a unique point $\tau(\theta)$.
(ii) Observe that B is defined by the maximum of a collection of functions linear in θ, which ensures that it is convex (Bertsekas, 1995).
(iii) Finally, the function $\theta^T\tau - B^*(\tau)$ satisfies the hypotheses of Danskin's theorem (Bertsekas, 1995), from which we conclude that B is differentiable with $\nabla B(\theta) = \tau(\theta)$ as claimed. ■

Given the interpretation of $\tau(\theta)$ as a pseudomarginal, this last property of B is analogous to the well-known cumulant generating property of A—namely, $\nabla A(\theta) = \mu(\theta)$—as specified in Lemma 1.

3.3 Convexified Bethe Surrogate

The following example provides a more concrete illustration of this constructive procedure, using a tree-based approximation to the marginal polytope, and a convexifed Bethe entropy approximation (Wainwright et al., 2005). As with the ordinary Bethe approximation (Yedidia et al., 2005), the cost function and constraint set underlying this approximation are exact for any tree-structured Markov random field.

Relaxed polytope: We begin by describing a relaxed version $\mathrm{REL}_\phi(G)$ of the marginal polytope $\mathrm{MARG}_\phi(\phi)$. Let τ_s and τ_{st} represent a collection of singleton and pairwise pseudomarginals, respectively, associated with vertices and edges of a graph G. These quantities, as locally valid marginal distributions, must satisfy the following set of local consistency conditions:

$$\mathrm{LOCAL}_\phi(G) \quad := \quad \Big\{\tau \in \mathbb{R}_+^d \,\Big|\, \sum_{x_s} \tau_s(x_s) = 1, \;\; \sum_{x_t} \tau_{st}(x_s, x_t) = \tau_s(x_s)\Big\}.$$

By construction, we are guaranteed the inclusion $\mathrm{MARG}_\phi(G) \subset \mathrm{LOCAL}_\phi(G)$. Moreover, a special case of the junction tree theory (Lauritzen, 1996) guarantees that equality holds when the underlying graph is a tree (in particular, any $\tau \in \mathrm{LOCAL}_\phi(G)$ can be realized as the marginals of the tree-structured distribution of the form (9)). However, the inclusion is strict for any graph with cycles; see Appendix A for further discussion of this issue.

Entropy approximation: We now define an entropy approximation B_ρ^* that is finite for any pseudomarginal τ in the relaxed set $\mathrm{LOCAL}_\phi(G)$. We begin by considering a collection $\{T \in \mathfrak{T}\}$ of spanning trees associated with the original graph. Given $\tau \in \mathrm{LOCAL}_\phi(G)$, there is—for each spanning tree T—a unique tree-structured distribution that has marginals τ_s and τ_{st} on the vertex set V and edge set $E(T)$ of the tree. Using Equations (9) and (10), the entropy of this tree-structured distribution can be computed explicitly. The *convexified Bethe entropy* approximation is based on taking a convex combination of these tree entropies, where each tree is weighted by a probability $\rho(T) \in [0,1]$. Doing so and expanding the sum yields

$$B_\rho^*(\tau) \quad := \quad \sum_{T \in \mathfrak{T}} \rho(T) \Big\{ \sum_{s \in V} H_s(\tau_s) - \sum_{(s,t) \in E(T)} I_{st}(\tau_{st}) \Big\} \;=\; \sum_{s \in V} H_s(\tau_s) - \sum_{(s,t) \in E} \rho_{st} I_{st}(\tau_{st}), \quad (12)$$

where $\rho_{st} = \sum_T \rho(T) \mathbb{I}[(s,t) \in T]$ are the *edge appearance probabilities* defined by the distribution ρ over the tree collection. By construction, the function B_ρ^* is differentiable; moreover, it can be shown (Wainwright et al., 2005) that it is strictly convex for any vector $\{\rho_{st}\}$ of strictly positive edge appearance probabilities.

Bethe surrogate and reweighted sum-product: We use these two ingredients—the relaxation $\mathrm{LOCAL}_\phi(G)$ of the marginal polytope, and the convexified Bethe entropy approximation (12)—to define the following convex surrogate

$$B_\rho(\theta) \quad := \quad \max_{\tau \in \mathrm{LOCAL}_\phi(G)} \left\{ \theta^T \tau - B^*_\rho(\tau) \right\}. \tag{13}$$

Since the conditions of Proposition 2 are satisfied, we are guaranteed that B_ρ is convex and differentiable on $\mathbb{R}^d$, and moreover that $\nabla B_\rho(\theta) = \tau(\theta)$, where (for each $\theta \in \mathbb{R}^d$) the quantity $\tau(\theta)$ denotes the unique optimum of problem (13). Perhaps most importantly, the optimizing pseudo-marginals $\tau(\theta)$ can be computed efficiently using a *tree-reweighted variant* of the sum-product message-passing algorithm (Wainwright et al., 2005). This method operates by passing "messages", which in the multinomial case are simply m-vectors of non-negative numbers, along edges of the graph. We use $M_{ts} = \{M_{ts}(i), i = 0, \ldots, m-1\}$ to represent the message passed from node t to node s. In the tree-reweighted variant, these messages are updated according to the following recursion

$$M_{ts}(x_s) \quad \leftarrow \quad \sum_{x_t} \exp\left\{ \theta_t(x_t) \frac{\theta_{st}(x_s, x_t)}{\rho_{st}} \right\} \frac{\prod_{u \in \Gamma(t) \setminus s} \left[M_{ut}(x_t) \right]^{\rho_{ut}}}{\left[M_{st}(x_t) \right]^{1-\rho_{st}}}. \tag{14}$$

Here $\Gamma(t)$ denotes the set of all neighbors of node t in the graph. Upon convergence of the updates, the fixed point messages M^* yield the unique global optimum of the optimization problem (13) via the following equations

$$\tau_s(x_s; \theta) \quad \propto \quad \exp\left\{ \theta_s(x_s) \right\} \prod_{u \in \Gamma(s)} \left[M_{us}(x_s) \right]^{\rho_{us}}, \quad \text{and} \tag{15a}$$

$$\tau_{st}(x_s, x_t; \theta) \quad \propto \quad \exp\left\{ \theta_s(x_s) + \theta_t(x_t) + \frac{\theta_{st}(x_s, x_t)}{\rho_{st}} \right\} \frac{\prod_{u \in \Gamma(s)} \left[M_{us}(x_s) \right]^{\rho_{us}} \prod_{v \in \Gamma(s)} \left[M_{vs}(x_s) \right]^{\rho_{vs}}}{M_{st}(x_t)\, M_{ts}(x_s)} \tag{15b}$$

Further details on these updates and their properties can be found in Wainwright et al. (2005).

4. Joint Estimation and Prediction Using Surrogates

We now turn to consideration of how convex surrogates, as constructed by the procedure described in the previous section, are useful for both approximate parameter estimation as well as prediction.

4.1 Approximate Parameter Estimation

Suppose that we are given i.i.d. samples $\{X^1, \ldots, X^n\}$ from an MRF of the form (3), where the underlying true parameter θ^* is unknown. One standard way in which to estimate θ^* is via maximum likelihood (possibly with an additional regularization term); in this particular exponential family setting, it is straightforward to show that the (normalized) log likelihood takes the form

$$\ell(\theta) \quad = \quad \langle \widehat{\mu}^n, \theta \rangle - A(\theta) - \lambda^n R(\theta)$$

where function R is a regularization term with an associated (possibly data-dependent) weight λ^n. The quantities $\widehat{\mu}^n := \frac{1}{n} \sum_{i=1}^n \phi(X^i)$ are the empirical moments defined by the data. For the indicator-based exponential representation (5), these empirical moments correspond to a set of singleton and pairwise marginal distributions, denoted $\widehat{\mu}^n_s$ and $\widehat{\mu}^n_{st}$ respectively.

It is intractable to maximize the regularized likelihood directly, due to the presence of the cumulant generating function A. Thus, a natural thought is to use the convex surrogate B to define an alternative estimator obtained by maximizing the regularized *surrogate likelihood*:

$$\ell_B(\theta) \quad := \quad \langle \widehat{\mu}^n, \theta \rangle - B(\theta) - \lambda^n R(\theta). \tag{16}$$

By design, the surrogate B and hence the surrogate likelihood ℓ_B, as well as their derivatives, can be computed in a straightforward manner (typically by some sort of message-passing algorithm). It is thus straightforward to compute the parameter $\widehat{\theta}^n$ achieving the maximum of the regularized surrogate likelihood (for instance, gradient descent would a simple though naive method).

For the tree-reweighted Bethe surrogate (13), we have shown in previous work (Wainwright et al., 2003b) that in the absence of regularization, the optimal parameter estimates $\widehat{\theta}^n$ have a very simple closed-form solution, specified in terms of the weights ρ_{st} and the empirical marginals $\widehat{\mu}$. (We make use of this closed form in our experimental comparison in Section 7; see Equation (32).) If a regularizing term is added, these estimates no longer have a closed-form solution, but the optimization problem (16) can still be solved efficiently using the tree-reweighted sum-product algorithm (Wainwright et al., 2003b, 2005).

4.2 Joint Estimation and Prediction

Using such an estimator, we now consider a joint approach to estimation and prediction. Recalling the basic set-up, we are given an initial set of i.i.d. samples $\{x^1, \ldots, x^n\}$ from $p(\cdot\,; \theta^*)$, where the true model parameter θ^* is unknown. These samples are used to form an estimate of the Markov random field. We are then given a noisy observation y of a new sample $z \sim p(\cdot\,; \theta^*)$, and the goal is to use this observation in conjunction with the fitted model to form a near-optimal estimate of z. The key point is that the same convex surrogate B is used both in forming the surrogate likelihood (16) for approximate parameter estimation, and in the variational method (11) for performing prediction.

For a given fitted model parameter $\theta \in \mathbb{R}^d$, the central object in performing prediction is the posterior distribution $p(z \mid y; \theta) \propto p(z; \theta)\; p(y \mid z)$. In the exponential family setting, for a fixed noisy observation y, this posterior can always be written as a new exponential family member, described by parameter $\theta + \gamma(y)$. (Here the term $\gamma(y)$ serves to incorporate the effect of the noisy observation.) With this set-up, the procedure consists of the following steps:

Joint estimation and prediction:

1. Form an approximate parameter estimate $\widehat{\theta}^n$ from an initial set of i.i.d. data $\{x^1, \ldots, x^n\}$ by maximizing the (regularized) surrogate likelihood ℓ_B.

2. Given a new noisy observation y (i.e., a contaminated version of $z \sim p(\cdot\,; \theta^*)$) specified by a factorized conditional distribution of the form $p(y \mid z) = \prod_{s=1}^N p(y_s \mid z_s)$, incorporate it into the model by forming the new exponential parameter

$$\widehat{\theta}^n_s(\,\cdot\,) + \gamma_s(y)$$

where $\gamma_s(y)$ merges the new data with the fitted model $\widehat{\theta}^n$. (The specific form of γ depends on the observation model.)

3. Using the message-passing algorithm associated with the convex surrogate B, compute approximate marginals $\tau(\widehat{\theta}+\gamma)$ for the distribution that combines the fitted model with the new observation. Use these approximate marginals to construct a prediction $\widehat{z}(y;\tau)$ of z based on the observation y and pseudomarginals τ.

Examples of the prediction task in the final step include smoothing (e.g., denoising of a noisy image) and interpolation (e.g., in the presence of missing data). We provide a concrete illustration of such a prediction problem in Section 6 using a mixture-of-Gaussians observation model. The most important property of this joint scheme is that the *convex surrogate B* underlies both the parameter estimation phase (used to form the surrogate likelihood), and the prediction phase (used in the variational method for computing approximate marginals). It is this matching property that will be shown to be beneficial in terms of overall performance.

5. Analysis

In this section, we turn to the analysis of the surrogate-based method for estimation and prediction. We begin by exploring the asymptotic behavior of the parameter estimator. We then prove a Lipschitz stability result applicable to any variational method that is based on a strongly concave entropy approximation. This stability result plays a central role in our subsequent development of bounds on the performance loss in Section 6.

5.1 Estimator Asymptotics

We begin by considering the asymptotic behavior of the parameter estimator $\widehat{\theta}^n$ defined by the surrogate likelihood (16). Since this parameter estimator is a particular type of M-estimator (Serfling, 1980), its asymptotic behavior can be investigated using standard methods, as summarized in the following:

Proposition 3 *Recall the cumulant generating function A defined in Equation* (4)*. Let B be a strictly convex surrogate for A, defined via Equation* (11) *with a strictly concave entropy approximation $-B^*$. Consider the sequence of parameter estimates $\{\widehat{\theta}^n\}$ given by*

$$\widehat{\theta}^n \quad := \quad \arg\max_{\theta\in\mathbb{R}^d} \{\langle \widehat{\mu}^n, \theta\rangle - B(\theta) - \lambda^n R(\theta)\} \tag{17}$$

where R is a non-negative and convex regularizer, and the regularization parameter satisfies $\lambda^n = o(\frac{1}{\sqrt{n}})$.

Then for a general graph with cycles, the following results hold:

(a) we have $\widehat{\theta}^n \xrightarrow{p} \widehat{\theta}$, where $\widehat{\theta}$ is (in general) distinct from the true parameter θ^*.

(b) the estimator is asymptotically normal:

$$\sqrt{n}\big[\widehat{\theta}^n - \widehat{\theta}\big] \quad \xrightarrow{d} \quad N\Big(0,\, \big(\nabla^2 B(\widehat{\theta})\big)^{-1} \nabla^2 A(\theta^*) \big(\nabla^2 B(\widehat{\theta})\big)^{-1}\Big)$$

Proof By construction, the convex surrogate B and the (negative) entropy approximation B^* are a Fenchel-Legendre conjugate dual pair. From Proposition 2, the surrogate B is differentiable.

Moreover, the strict convexity of B and B^* ensure that the gradient mapping ∇B is one-to-one and onto the relative interior of the constraint set $\mathrm{REL}_\phi(G)$ (see Section 26 of Rockafellar (1970)). Moreover, the inverse mapping $(\nabla B)^{-1}$ exists, and is given by the dual gradient ∇B^*.

Let μ^* be the moment parameters associated with the true distribution θ^*—that is, $\mu^* = \mathbb{E}_{\theta^*}[\phi(X)]$. In the limit of infinite data, the asymptotic value of the parameter estimate is defined by

$$\nabla B(\widehat{\theta}) = \mu^*. \tag{18}$$

Note that μ^* belongs to the relative interior of $\mathrm{MARG}_\phi(G)$, and hence to the relative interior of $\mathrm{REL}_\phi(G)$. Therefore, Equation (18) has a unique solution $\widehat{\theta} = \nabla^{-1}B(\mu^*)$.

By strict convexity, the regularized surrogate likelihood (17) has a unique global maximum. Let us consider the optimality conditions defining this unique maximum $\widehat{\theta}^n$; they are given by $\nabla B(\widehat{\theta}^n) = \widehat{\mu}^n - \lambda^n \partial R(\widehat{\theta}^n)$, where $\partial R(\widehat{\theta}^n)$ denotes an arbitrary element of the subdifferential of the convex function R at the point $\widehat{\theta}^n$. We can now write

$$\nabla B(\widehat{\theta}^n) - \nabla B(\widehat{\theta}) \quad = \quad [\widehat{\mu}^n - \mu^*] - \lambda^n \partial R(\widehat{\theta}^n). \tag{19}$$

Taking inner products with the difference $\widehat{\theta}^n - \widehat{\theta}$ yields

$$0 \stackrel{(a)}{\leq} \left[\nabla B(\widehat{\theta}^n) - \nabla B(\widehat{\theta})\right]^T \left[\widehat{\theta}^n - \widehat{\theta}\right] \leq [\widehat{\mu}^n - \mu^*]^T \left[\widehat{\theta}^n - \widehat{\theta}\right] + \lambda^n \partial R(\widehat{\theta}^n)^T \left[\widehat{\theta} - \widehat{\theta}^n\right], \tag{20}$$

where inequality (a) follows from the convexity of B. From the convexity and non-negativity of R, we have

$$\lambda^n \partial R(\widehat{\theta}^n)^T \left[\widehat{\theta} - \widehat{\theta}^n\right] \leq \lambda^n \left[R(\widehat{\theta}) - R(\widehat{\theta}^n)\right] \leq \lambda^n R(\widehat{\theta}).$$

Applying this inequality and Cauchy-Schwartz to Equation (20) yields

$$0 \leq \left[\nabla B(\widehat{\theta}^n) - \nabla B(\widehat{\theta})\right]^T \left[\frac{\widehat{\theta}^n - \widehat{\theta}}{\|\widehat{\theta}^n - \widehat{\theta}\|}\right] \leq \|\widehat{\mu}^n - \mu^*\| + \lambda^n R(\widehat{\theta})$$

Since $\lambda^n = o(1)$ by assumption and $\|\widehat{\mu}^n - \mu^*\| = o_p(1)$ by the weak law of large numbers, the quantity $\left[\nabla B(\widehat{\theta}^n) - \nabla B(\widehat{\theta})\right]^T \left[\frac{\widehat{\theta}^n - \widehat{\theta}}{\|\widehat{\theta}^n - \widehat{\theta}\|}\right]$ converges in probability to zero. By the strict convexity of B, this fact implies that $\widehat{\theta}^n$ converges in probability to $\widehat{\theta}$, thereby completing the proof of part (a).

To establish part (b), we observe that $\sqrt{n}[\widehat{\mu}^n - \mu^*] \xrightarrow{d} N(0, \nabla^2 A(\theta^*))$ by the central limit theorem. Using this fact and applying the delta method to Equation (19) yields that

$$\sqrt{n}\nabla^2 B(\widehat{\theta}) \left[\widehat{\theta}^n - \widehat{\theta}\right] \xrightarrow{d} N\left(0, \nabla^2 A(\theta^*)\right),$$

where we have used the fact that $\sqrt{n}\lambda^n = o(1)$. The strict convexity of B guarantees that $\nabla^2 B(\widehat{\theta})$ is invertible, so that claim (b) follows. ■

A key property of the estimator is its *inconsistency*—that is, the estimated model differs from the true model θ^* even in the limit of large data. Despite this inconsistency, we will see that the approximate parameter estimates $\widehat{\theta}^n$ are nonetheless useful for performing prediction.

5.2 Global Algorithmic Stability

A desirable property of any algorithm—particularly one applied to statistical data—is that it exhibit an appropriate form of stability with respect to its inputs. Not all message-passing algorithms have such stability properties. For instance, the standard sum-product message-passing algorithm, although stable for weakly coupled MRFs (Ihler et al., 2005; Mooij and Kappen, 2005b,a; Tatikonda and Jordan, 2002; Tatikonda, 2003), can be highly unstable in other regimes due to the appearance of multiple local optima in the non-convex Bethe problem. However, previous experimental work has shown that methods based on convex relaxations, including the reweighted sum-product (or belief propagation) algorithm (Wainwright et al., 2003b), reweighted generalized BP (Wiegerinck, 2005), and log-determinant relaxations (Wainwright and Jordan, 2005) appear to be *globally stable*—that is, even for very strongly coupled problems. For instance, Figure 2 provides a simple illustration of the instability of the ordinary sum-product algorithm, contrasted with the stability of the tree-reweighted updates. Wiegerinck (2005) provides similar results for reweighted forms of the generalized belief propagation. Here we provide theoretical support for these empirical observations: in particular, we prove that, in sharp contrast to non-convex methods, any variational method based on a strongly convex entropy approximation is globally stable. This stability property plays a fundamental role in providing a performance guarantee on joint estimation/prediction methods.

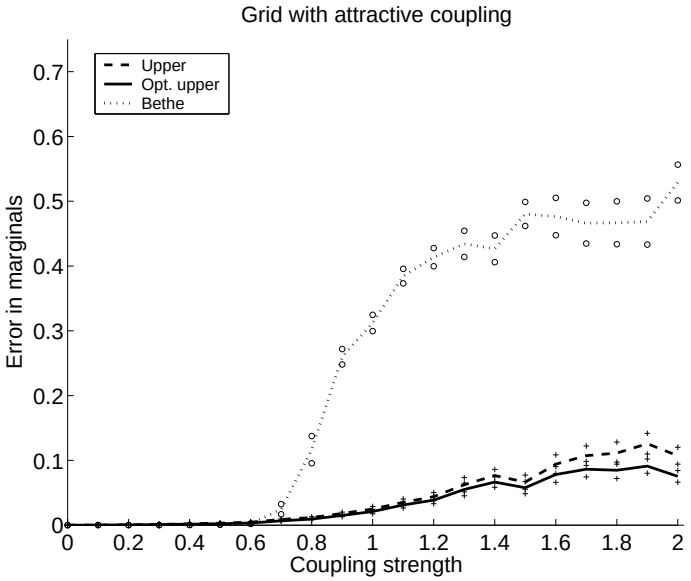


Figure 2: Contrast of the instability of the ordinary sum-product algorithm with the stability of the tree-reweighted version (Wainwright et al., 2005). Results shown with a grid with $N = 100$ nodes over a range of attractive coupling strengths. The ordinary sum-product undergoes a phase transition, after which the quality of marginal approximations degrades substantially. The tree-reweighted algorithm, shown for two different settings of the edge weights ρ_{st}, remains stable over the full range of coupling strengths. See Wainwright et al. (2005) for full details.

We begin by noting that for a multinomial Markov random field (3), the computation of the exact marginal probabilities is a globally Lipschitz operation:

Lemma 4 *For any discrete Markov random field* (3)*, there is a constant $L < +\infty$ such that*

$$\|\mu(\theta+\delta) - \mu(\theta)\| \quad \leq \quad L\|\delta\| \qquad \text{for all } \theta, \delta \in \mathbb{R}^d.$$

This lemma, which is proved in Appendix B, guarantees that small changes in the problem parameters—that is, "perturbations" δ—lead to correspondingly small changes in the computed marginals.

Our goal is to establish analogous Lipschitz properties for variational methods. In particular, it turns out that any variational method based on a suitably concave entropy approximation satisfies such a stability condition. More precisely, a function $f : \mathbb{R}^n \to \mathbb{R}$ is *strongly convex* if there exists a constant $c > 0$ such that $f(y) \geq f(x) + \nabla f(x)^T (y-x) + \frac{c}{2}\|y-x\|^2$ for all $x, y \in \mathbb{R}^n$. For a twice continuously differentiable function, this condition is equivalent to having the eigenspectrum of the Hessian $\nabla^2 f(x)$ be uniformly bounded below by c. With this definition, we have:

Proposition 5 *Consider any strictly convex surrogate B based on a strongly concave entropy approximation $-B^*$. Then there exists a constant $R < +\infty$ such that*

$$\|\tau(\theta+\delta) - \tau(\theta)\| \quad \leq \quad R\|\delta\| \qquad \text{for all } \theta, \delta \in \mathbb{R}^d.$$

Proof From Proposition 2, we have $\tau(\theta) = \nabla B(\theta)$, so that the statement is equivalent to the assertion that the gradient ∇B is a Lipschitz function. Applying the mean value theorem to ∇B, we can write $\nabla B(\theta+\delta) - \nabla B(\theta) = \nabla^2 B(\theta + t\delta)\delta$ where $t \in [0,1]$. Consequently, in order to establish the Lipschitz condition, it suffices to show that the spectral norm of $\nabla^2 B(\gamma)$ is uniformly bounded above over all $\gamma \in \mathbb{R}^d$. Since B and B^* are a strictly convex Legendre pair, we have $\nabla^2 B(\theta) = [\nabla^2 B^*(\tau(\theta))]^{-1}$. By the strong convexity of B^*, we are guaranteed that the spectral norm of $\nabla^2 B^*(\tau)$ is uniformly bounded away from zero, which yields the claim. ■

A number of existing entropy approximations can be shown to be strongly concave. In Appendix C, we provide a detailed proof of this fact for the convexified Bethe entropy (12).

Lemma 6 *For any set $\{\rho_{st}\}$ of strictly positive edge appearance probabilities, the convexified Bethe entropy* (12) *is strongly concave.*

We note that the same argument can be used to establish strong concavity for the reweighted Kikuchi approximations studied by Wiegerinck (2005). Moreover, it can be shown that the Gaussian-based log-determinant relaxation proposed by Wainwright and Jordan (2006) is also strongly concave. For all of these variational methods, then, Proposition 5 guarantees that the pseudomarginal computation is globally Lipschitz stable, thereby providing theoretical confirmation of previous experimental results (Wiegerinck, 2005; Wainwright et al., 2005; Wainwright and Jordan, 2006). The entropy approximations that underlie other variational methods (e.g., expectation-propagation Minka, 2001) can also be modified so as to be strongly concave; Proposition 5 provides further justification—in addition to the obvious benefit of unique global optima—for such "convexification" of entropy approximations.

6. Performance Bounds

In this section, we develop theoretical bounds on the performance loss of our approximate approach to joint estimation and prediction, relative to the unattainable Bayes optimum. So as not to unnecessarily complicate the result, we focus on the performance loss in the infinite data limit[2] (i.e., for which the number of samples $n = +\infty$).

In the infinite data setting, the Bayes optimum is unattainable for two reasons:

1. it is based on knowledge of the exact parameter θ^*, which is not easy to obtain.

2. it assumes (in the prediction phase) that computing exact marginal probabilities μ of the Markov random field is feasible.

Of these two difficulties, it is the latter assumption—regarding the computation of marginal probabilities—that is the most serious. As discussed earlier, there do exist computationally tractable estimators of θ^* that are consistent though not statistically efficient under appropriate conditions; one example is the pseudolikelihood method (Besag, 1975, 1977) mentioned previously. On the other hand, MCMC methods may be used to generate stochastic approximations to marginal probabilities, but may require greater than polynomial complexity.

Recall from Proposition 3 that the parameter estimator based on the surrogate likelihood ℓ_B is *inconsistent*, in the sense that the parameter vector $\widehat{\theta}$ returned in the limit of infinite data is generally not equal to the true parameter θ^*. Our analysis in this section will demonstrate that this inconsistency is beneficial.

6.1 Problem Set-up

Although the ideas and techniques described here are more generally applicable, we focus here on a special observation model so as to obtain a concrete result.

Observation model: In particular, we assume that the multinomial random vector $X = \{X_s,\ s \in V\}$ defined by the Markov random field (3) is a label vector for the components in a finite mixture of Gaussians. For each node $s \in V$, we specify a new random variable Z_s by the conditional distribution

$$p(Z_s = z_s \,|\, X_s = j) \sim N(\nu_j, \sigma_j^2) \quad \text{for } j \in \{0, 1, \ldots, m-1\},$$

so that Z_s is a mixture of m Gaussians. Such Gaussian mixture models are widely used in spatial statistics as well as statistical signal and image processing (Crouse et al., 1998; Ripley, 1981; Titterington et al., 1986).

Now suppose that we observe a noise-corrupted version of z_s—namely, a vector Y of observations with components of the form

$$Y_s = \alpha Z_s + \sqrt{1-\alpha^2} W_s, \tag{21}$$

where $W_s \sim N(0,1)$ is additive Gaussian noise, and the parameter $\alpha \in [0,1]$ specifies the signal-to-noise ratio (SNR) of the observation model. Note that $\alpha = 0$ corresponds to pure noise, whereas $\alpha = 1$ corresponds to completely uncorrupted observations.

2. Note, however, that modified forms of the results given here, modulo the usual $O(1/n)$ corrections, hold for the finite data setting.

Optimal prediction: Our goal is to compute an optimal estimate $\widehat{z}(y)$ of z as a function of the observation $Y = y$, using the mean-squared error as the risk function. The essential object in this computation is the posterior distribution $p(x \mid y; \theta^*) \propto p(x; \theta^*)\, p(y \mid x)$, where the conditional distribution $p(y \mid x)$ is defined by the observation model (21). As shown in the sequel, the posterior distribution (with y fixed) can be expressed as an exponential family member of the form $\theta^* + \gamma(y)$ (see Equation (26a)). Disregarding computational cost, it is straightforward to show that the optimal Bayes least squares estimator (BLSE) takes the form

$$\widehat{z}_s^{\text{opt}}(Y; \theta^*) \quad := \quad \sum_{j=0}^{m-1} \mu_{s;j}(\theta^* + \gamma(Y)) \Big[\omega_j(\alpha) \big(Y_s - \alpha \nu_j \big) + \nu_j \Big], \tag{22}$$

where $\mu_{s;j}(\theta^* + \gamma)$ denotes the marginal probability associated with the posterior distribution $p(x; \theta^* + \gamma)$, and

$$\omega_j(\alpha) \quad := \quad \frac{\alpha \sigma_j^2}{\alpha^2 \sigma_j^2 + (1 - \alpha^2)} \tag{23}$$

is the usual BLSE weighting for a Gaussian with variance σ_j^2.

Approximate prediction: Since the marginal distributions $\mu_{s;j}(\theta^* + \gamma)$ are intractable to compute exactly, it is natural to consider an approximate predictor, based on a set τ of pseudomarginals computed from a variational relaxation. More explicitly, we run the variational algorithm on the parameter vector $\widehat{\theta} + \gamma$ that is obtained by combining the new observation y with the fitted model $\widehat{\theta}$, and use the outputted pseudomarginals $\tau_s(\cdot; \widehat{\theta} + \gamma)$ as weights in the approximate predictor

$$\widehat{z}_s^{\text{app}}(Y; \widehat{\theta}) \quad := \quad \sum_{j=0}^{m-1} \tau_{s;j}(\widehat{\theta} + \gamma(Y)) \Big[\omega_j(\alpha) \big(Y_s - \alpha \nu_j \big) + \nu_j \Big], \tag{24}$$

where the weights ω are defined in Equation (23).

We now turn to a comparison of the Bayes least-squares estimator (BLSE) defined in Equation (22) to the surrogate-based predictor (24). Since (by definition) the BLSE is optimal for the mean-squared error (MSE), using the surrogate-based predictor will necessarily lead to a larger MSE. Our goal is to prove an upper bound on the maximal possible increase in this MSE, where the bound is specified in terms of the underlying model θ^* and the SNR parameter α. More specifically, for a given problem, we define the mean-squared errors

$$\mathrm{R}^{\text{opt}}(\alpha, \theta^*) := \frac{1}{N} \mathbb{E} \| \widehat{z}^{\text{opt}}(Y; \theta^*) - Z \|^2, \quad \text{and} \quad \mathrm{R}^{\text{app}}(\alpha, \widehat{\theta}) := \frac{1}{N} \mathbb{E} \| \widehat{z}^{\text{app}}(Y; \widehat{\theta}) - Z \|^2,$$

of the Bayes-optimal and surrogate-based predictors respectively, where the expectation is taken over the joint distribution of (Y, Z). We seek upper bounds on the increase $\Delta \mathrm{R}(\alpha, \theta^*, \widehat{\theta}) := \mathrm{R}^{\text{app}}(\alpha, \widehat{\theta}) - \mathrm{R}^{\text{opt}}(\alpha, \theta^*)$ of the approximate predictor relative to Bayes optimum.

6.2 Role of Stability

Before providing a technical statement and proof, we begin with some intuition underlying the bounds, and the role of Lipschitz stability. First, consider the low SNR regime ($\alpha \approx 0$) in which the observation Y is heavily corrupted by noise. In the limit $\alpha = 0$, the new observations are pure

noise, so that the prediction of Z should be based simply on the estimated model—namely, the true model $p(\cdot;\theta^*)$ in the Bayes optimal case, and the "incorrect" model $p(\cdot;\widehat{\theta})$ for the method based on surrogate likelihood. The key point here is the following: by properties of the MLE and surrogate-based estimator, the following equalities hold:

$$\nabla A(\theta^*) \stackrel{(a)}{=} \mu(\theta^*) \stackrel{(b)}{=} \mu^* \stackrel{(c)}{=} \tau(\widehat{\theta}) \stackrel{(d)}{=} \nabla B(\widehat{\theta}).$$

Here equality (a) follows from Lemma 1, whereas equality (b) follows from the moment-matching property of the MLE in exponential families. Equalities (c) and (d) hold from the Proposition 2 and the pseudomoment-matching property of the surrogate-based parameter estimator (see proof of Proposition 3). As a key consequence, it follows that the combination of surrogate-based estimation and prediction is *functionally indistinguishable* from the Bayes-optimal behavior in the limit of $\alpha = 0$. More specifically, in the limiting case, the errors systematically introduced by the inconsistent learning procedure are cancelled out exactly by the approximate variational method for computing marginal distributions. Of course, exactness for $\alpha = 0$ is of limited interest; however, when combined with the Lipschitz stability ensured by Proposition 5, it allows us to gain good control of the low SNR regime. At the other extreme of high SNR ($\alpha \approx 1$), the observations are nearly perfect, and hence dominate the behavior of the optimal estimator. More precisely, for α close to 1, we have $\omega_j(\alpha) \approx 1$ for all $j = 0, 1, \ldots, m-1$, so that $\widehat{z}^{\text{opt}}(Y;\theta^*) \approx Y \approx \widehat{z}^{\text{app}}(Y;\widehat{\theta})$. Consequently, in the high SNR regime, accuracy of the marginal computation has little effect on the accuracy of the predictor.

6.3 Bound on Performance Loss

Although bounds of this nature can be developed in more generality, for simplicity in notation we focus here on the case of $m = 2$ mixture components. We begin by introducing the factors that play a role in our bound on the performance loss $\Delta R(\alpha, \theta^*, \widehat{\theta})$. First, the Lipschitz stability enters in the form of the quantity:

$$L(\theta^*;\widehat{\theta}) \;:=\; \sup_{\delta \in \mathbb{R}^d} \sigma_{\max}\big(\nabla^2 A(\theta^* + \delta) - \nabla^2 B(\widehat{\theta} + \delta)\big), \tag{25}$$

where $\sigma_{\max}$ denotes the maximal singular value. Following the argument in the proof of Proposition 5, it can be seen that $L(\theta^*;\widehat{\theta})$ is finite.

Second, in order to apply the Lipschitz stability result, it is convenient to express the effect of introducing a new observation vector y, drawn from the additive noise observation model (21), as a perturbation of the exponential parameterization. In particular, for any parameter $\theta \in \mathbb{R}^d$ and observation y from the model (21), the conditional distribution $p(x|y;\theta)$ can be expressed as $p(x;\theta + \gamma(y,\alpha))$, where the exponential parameter $\gamma(y,\alpha)$ has components[3]

$$\gamma_s = \frac{1}{2}\left\{ \log \frac{\alpha^2\sigma_0^2 + (1-\alpha^2)}{\alpha^2\sigma_1^2 + (1-\alpha^2)} + \frac{(y_s - \alpha\nu_0)^2}{\alpha^2\sigma_0^2 + (1-\alpha^2)} - \frac{(y_s - \alpha\nu_1)^2}{\alpha^2\sigma_1^2 + (1-\alpha^2)} \right\} \qquad \forall\, s \in V. \tag{26a}$$

$$\gamma_{st} = 0 \qquad \forall\, (s,t) \in E. \tag{26b}$$

See Appendix D for a derivation of these relations.

3. For consistency in notation with the general $m > 2$ case, these components should be labeled as $\gamma_{s;1}$ and $\gamma_{st;11}$, but we drop the additional indices for simplicity.

Third, it is convenient to have short notation for the Gaussian estimators of each mixture component:

$$g_j(Y_s;\alpha) \quad := \quad \omega_j(\alpha)\,(Y_s - \alpha\nu_j) + \nu_j \qquad \text{for } j = 0,1,$$

With this notation, we have the following

Theorem 7 *The MSE increase* $\Delta\mathrm{R}(\alpha,\theta^*,\widehat{\theta}) := \mathrm{R}(\alpha,\widehat{\theta}) - \mathrm{R}(\alpha,\theta^*)$ *is upper bounded by*

$$\Delta\mathrm{R}(\alpha,\theta^*,\widehat{\theta}) \quad \leq \quad \mathbb{E}\left\{ \min\left(1,\, L(\theta^*;\widehat{\theta})\frac{\|\gamma(Y;\alpha)\|_2}{\sqrt{N}}\right) \sqrt{\frac{\sum_{s=1}^N |g_1(Y_s) - g_0(Y_s)|^4}{N}} \right\}. \qquad (27)$$

Before proving the bound (27), we illustrate it by considering its behavior in some special cases.

6.3.1 Superiority to True Model

Theorem 7 can be used to establish that applying an approximate message-passing algorithm to the "incorrect" model yields prediction results superior to those obtained by applying the same message-passing algorithm to the true underlying model. To see one regime in which this claim is true, consider the low SNR limit in which $\alpha \to 0^+$. In this limit, it can be seen that $\|\gamma(Y;\alpha)\| \to 0$, so that the the overall bound $\Delta\mathrm{R}(\alpha)$ tends to zero. That is, the combination of approximate estimation and approximate prediction is asymptotically optimal in the low SNR limit. In sharp contrast, this claim need not be true when approximate prediction is applied to the true underlying model. As a particular example, consider a Gaussian mixture with $m = 2$ components, with equal variances but distinct means (say $\nu_0 = -1$ and $\nu_1 = 1$). Moreover, suppose that the mixture indicator vectors $X \in \{0,1\}^N$ are sampled from an underlying distribution $p(x;\theta^*)$, and let $\mu_s = [\mu_{s;0}\ \mu_{s;1}]$ denote the marginal distributions associated with this underlying model. In the limit of zero SNR (i.e., $\alpha = 0$), it is straightforward to see that the BLSE of Z is simply its mean, given (for component $s \in V$) by

$$\mathbb{E}[Z_s \mid Y] \;=\; \mathbb{E}[Z_s] \quad = \quad \mu_{s;0}\nu_0 + \mu_{s;1}\nu_1 \;=\; \mu_{s;1} - \mu_{s;0},$$

for the two-component mixture specified above. Now suppose that *when applied to this true model, the approximate message-passing algorithm yields an incorrect set of singleton pseudomarginals*—say $\tau_s \neq \mu_s$. Since standard message-passing algorithms are rarely (if ever) exactly correct on non-trivial models with cycles, this assumption is more than reasonable. Consequently applying the approximate predictor to the true model will yield an estimate of Z which is incorrect, even in the zero SNR limit; in particular, the approximate estimate is given by

$$\widehat{Z}(Y;\theta^*) \quad = \quad \tau_{s;0}\nu_0 + \tau_{s;1}\nu_1 \;=\; \tau_{s;1} - \tau_{s;0} \;\neq\; \mathbb{E}[Z].$$

Thus, in contrast to the combination of approximate estimation with approximate estimation, applying the approximate message-passing algorithm to the true model fails to be exact even in the limit of zero SNR. In fact, our later experimental results show that the superiority of using the "wrong" model holds for a broader range of SNRs as well (see Figure 4).

We conclude by turning to the high SNR limit as $\alpha \to 1^-$, in which we see that $\omega_j(\alpha) \to 1$ for $j = 0,1$, which drives the differences $|g_1(Y_s) - g_0(Y_s)|$, and in turn the overall bound $\Delta\mathrm{R}(\alpha)$ to zero. Thus, the surrogate-based method is optimal in both the low and high SNR regimes; its behavior in the intermediate regime is governed by the balance between these two terms.

6.3.2 EFFECT OF EQUAL VARIANCES

Now consider the special case of equal variances $\sigma^2 \equiv \sigma_0^2 = \sigma_1^2$, in which case $\omega(\alpha) \equiv \omega_0(\alpha) = \omega_1(\alpha)$. Thus, the difference $g_1(Y_s,\alpha) - g_0(Y_s,\alpha)$ simplifies to $(1-\alpha\omega(\alpha))(\nu_1-\nu_0)$, so that the bound (27) reduces to

$$\Delta R(\alpha,\theta^*,\widehat{\theta}) \;\leq\; (1-\alpha\omega(\alpha))^2(\nu_1-\nu_0)^2\,\mathbb{E}\left\{\min\left(1,\, L(\theta^*;\widehat{\theta})\frac{\|\gamma(Y;\alpha)\|_2}{\sqrt{N}}\right)\right\}. \tag{28}$$

As shown by the simpler expression (28), for $\nu_1 \approx \nu_0$, the MSE increase is very small, since such a two-component mixture is close to a pure Gaussian.

6.3.3 EFFECT OF MEAN DIFFERENCES

Finally consider the case of equal means $\nu \equiv \nu_0 = \nu_1$ in the two Gaussian mixture components. In this case, we have $g_1(Y_s,\alpha) - g_0(Y_s,\alpha) = [\omega_1(\alpha) - \omega_0(\alpha)]\,[Y_s - \alpha\nu]$, so that the bound (27) reduces to

$$\Delta R(\alpha,\theta^*,\widehat{\theta}) \;\leq\; [\omega_1(\alpha)-\omega_0(\alpha)]^2\;\mathbb{E}\left\{\min\left(1,\, L(\theta^*;\widehat{\theta})\frac{\|\gamma(Y;\alpha)\|_2}{\sqrt{N}}\right)\sqrt{\frac{\sum_s (Y_s-\alpha\nu)^4}{N}}\right\}.$$

Here the MSE increase depends on the SNR α and the difference

$$\omega_1(\alpha)-\omega_0(\alpha) \;=\; \frac{\alpha\sigma_1^2}{\alpha^2\sigma_1^2+(1-\alpha^2)} - \frac{\alpha\sigma_0^2}{\alpha^2\sigma_0^2+(1-\alpha^2)} \;=\; \frac{(1-\alpha^2)\,(\sigma_1^2-\sigma_0^2)}{\left[\alpha^2\sigma_0^2+(1-\alpha^2)\right]\,\left[\alpha^2\sigma_1^2+(1-\alpha^2)\right]}.$$

Observe, in particular, that the MSE increases tends to zero as the difference $\sigma_1^2-\sigma_0^2$ decreases, as should be expected intuitively.

6.4 Proof of Theorem 7

We now turn to the proof of the main bound (27). By the Pythagorean relation that characterizes the Bayes least squares estimator $\widehat{z}^{\,\mathrm{opt}}(Y;\theta^*) = \mathbb{E}_{(Z\,|\,Y,\theta^*)}[Z]$, we have

$$\begin{aligned}\Delta R(\alpha;\theta^*,\widehat{\theta}) \;&:=\; \frac{1}{N}\mathbb{E}\|\widehat{z}^{\,\mathrm{app}}(Y;\widehat{\theta}) - Z\|_2^2 - \frac{1}{N}\mathbb{E}\|\widehat{z}^{\,\mathrm{opt}}(Y;\theta^*) - Z\|_2^2 \\ &=\; \frac{1}{N}\mathbb{E}\|\widehat{z}^{\,\mathrm{app}}(Y;\widehat{\theta}) - \widehat{z}^{\,\mathrm{opt}}(Y;\theta^*)\|_2^2.\end{aligned}$$

Using the definitions of $\widehat{z}^{\,\mathrm{app}}(Y;\widehat{\theta})$ and $\widehat{z}^{\,\mathrm{opt}}(Y;\theta^*)$, some algebraic manipulation yields

$$\begin{aligned}\left[\widehat{z}_s^{\,\mathrm{app}}(Y;\widehat{\theta}) - \widehat{z}_s^{\,\mathrm{opt}}(Y;\theta^*)\right]^2 \;&=\; \left[\tau_s(\widehat{\theta}+\gamma) - \mu_s(\theta^*+\gamma)\right]^2\,[g_1(Y_s)-g_0(Y_s)]^2 \\ &\leq\; \left|\tau_s(\widehat{\theta}+\gamma) - \mu_s(\theta^*+\gamma)\right|\,[g_1(Y_s)-g_0(Y_s)]^2,\end{aligned}$$

where the second inequality uses the fact that $|\tau_s - \mu_s| \leq 1$ since τ_s and μ_s are marginal probabilities. Next we write

$$\begin{aligned}\frac{1}{N}\|\widehat{z}^{\,\mathrm{app}}(Y;\widehat{\theta}) - \widehat{z}^{\,\mathrm{opt}}(Y;\theta^*)\|_2^2 \;&\leq\; \frac{1}{N}\sum_{s=1}^N \left|\tau_s(\widehat{\theta}+\gamma) - \mu_s(\theta^*+\gamma)\right|\,[g_1(Y_s)-g_0(Y_s)]^2 \qquad (29)\\ &\leq\; \frac{1}{\sqrt{N}}\|\tau(\widehat{\theta}+\gamma) - \mu(\theta^*+\gamma)\|_2\,\sqrt{\frac{\sum_{s=1}^N |g_1(Y_s)-g_0(Y_s)|^4}{N}}\end{aligned}$$

where the last line uses the Cauchy-Schwarz inequality.

It remains to bound the 2-norm $\|\tau(\widehat{\theta}+\gamma)-\mu(\theta^*+\gamma)\|_2$. An initial naive bound follows from the fact $\tau_s, \mu_s \in [0,1]$ implies that $|\tau_s - \mu_s| \leq 1$, whence

$$\frac{1}{\sqrt{N}}\|\tau-\mu\|_2 \leq \quad 1. \tag{30}$$

An alternative bound, which will be better for small perturbations γ, can be obtained as follows. Using the relation $\tau(\widehat{\theta}) = \mu(\theta^*)$ guaranteed by the definition of the ML estimator and surrogate estimator, we have

$$\begin{aligned} \|\tau(\widehat{\theta}+\gamma)-\mu(\theta^*+\gamma)\|_2 &= \left\| \left[\tau(\widehat{\theta}+\gamma)-\tau(\widehat{\theta})\right] + [\mu(\theta^*)-\mu(\theta^*+\gamma)] \right\|_2 \\ &= \left\| \left[\nabla^2 B(\widehat{\theta}+s\gamma) - \nabla^2 A(\theta^*+t\gamma)\right]\gamma \right\|_2, \end{aligned}$$

for some $s,t \in [0,1]$, where we have used the mean value theorem. Thus, using the definition (25) of L, we have

$$\frac{1}{\sqrt{N}}\,\|\tau(\widehat{\theta}+\gamma)-\mu(\theta^*+\gamma)\|_2 \quad \leq \quad L(\theta^*;\widehat{\theta})\frac{\|\gamma(Y;\alpha)\|_2}{\sqrt{N}}. \tag{31}$$

Combining the bounds (30) and (31) and applying them to Equation (29), we obtain

$$\frac{1}{N}\|\widehat{z}^{\text{app}}(Y;\widehat{\theta}) - \widehat{z}^{\text{opt}}(Y;\theta^*)\|_2^2 \quad \leq \quad \min\left\{1,\, L(\theta^*;\widehat{\theta})\frac{\|\gamma(Y;\alpha)\|_2}{\sqrt{N}}\right\}\sqrt{\frac{\sum_{s=1}^N |g_1(Y_s)-g_0(Y_s)|^4}{N}}.$$

Taking expectations of both sides yields the result.

7. Experimental Results

In order to test our joint estimation/prediction procedure, we have applied it to coupled Gaussian mixture models on different graphs, coupling strengths, observation SNRs, and mixture distributions. Here we describe both experimental results to quantify the performance loss of the tree-reweighted sum-product algorithm (Wainwright et al., 2005), and compare it to both a baseline independence model, as well as a closely related heuristic method that uses the ordinary sum-product (or belief propagation) algorithm.

7.1 Methods

In Section 4.2, we described a generic procedure for joint estimation and prediction. Here we begin by describing the special case of this procedure when the underlying variational method is the tree-reweighted sum-product algorithm (Wainwright et al., 2005). Any instantiation of the tree-reweighted sum-product algorithm is specified by a collection of edge weights ρ_{st}, one for each edge (s,t) of the graph. The vector of edge weights must belong to the spanning tree polytope; see Wainwright et al. (2005) for further background on these weights and the reweighted algorithm. Given a fixed set of edge weights ρ, the joint procedure based on the tree-reweighted sum-product algorithm consists of the following steps:

1. Given an initial set of i.i.d. data $\{X^1, \ldots, X^n\}$, we first compute the empirical marginal distributions

$$\widehat{\mu}_{s;j} := \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[X_s^i = j], \qquad \widehat{\mu}_{st;jk} := \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[X_s^i = j]\, \mathbb{I}[X_t^i = k],$$

and use them to compute the approximate parameter estimate

$$\widehat{\theta}^n_{s;j} := \log \widehat{\mu}_{s;j}, \qquad \widehat{\theta}^n_{s;j} := \rho_{st} \log \frac{\widehat{\mu}_{st;jk}}{\widehat{\mu}_{s;j}\widehat{\mu}_{t;k}}. \tag{32}$$

As shown in our previous work (Wainwright et al., 2003b), the estimates (32) are the global maxima of the surrogate likelihood (16) based on the convexified Bethe approximation (12) without any regularization term (i.e., $R = 0$).

2. Given the new noisy observation Y of the form (21), we incorporate it by by forming the new exponential parameter

$$\widehat{\theta}^n_s + \gamma_s(Y),$$

where Equation (26a) defines γ_s for the Gaussian mixture model under consideration.

3. We then compute approximate marginals $\tau(\widehat{\theta} + \gamma)$ by running the TRW sum-product algorithm with edge appearance weights ρ_{st}, using the message updates (14), on the graphical model distribution with exponential parameter $\widehat{\theta} + \gamma$. We use the approximate marginals (see Equation (15)) to construct the prediction $\widehat{z}^{\text{app}}$ in Equation (24).

We evaluated the tree-reweighted sum-product based on its increase in mean-squared error (MSE) over the Bayes optimal predictor (22). Moreover, we compared the performance of the tree-reweighted approach to the following alternatives:

(a) As a baseline, we used the *independence model* in which the mixture distributions at each node are all assumed to be independent. In this case, ML estimates of the parameters are given by $\widehat{\theta}_{s;j} = \log \widehat{\mu}_{s;j}$, with all of the coupling terms $\widehat{\theta}_{st;jk}$ equal to zero. The prediction step reduces to computing the Bayes least squares estimate at each node independently, based only on the local data y_s.

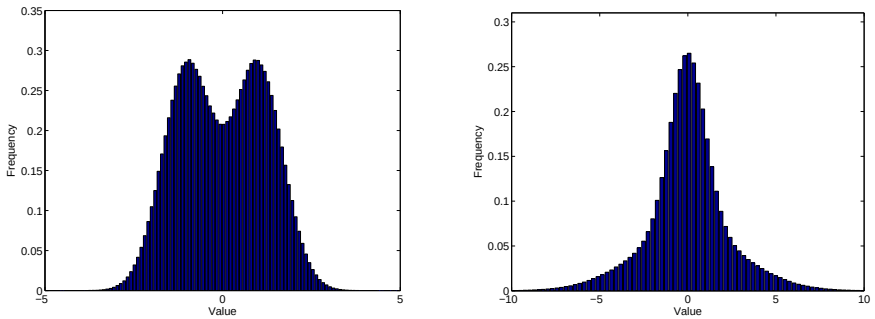
(b) The *standard sum-product or belief propagation* (BP) approach is closely related to the tree-reweighted sum-product method, but based on the edge weights $\rho_{st} = 1$ for all edges. In particular, we first form the approximate parameter estimate $\widehat{\theta}$ using Equation (32) with $\rho_{st} = 1$. As shown in our previous work (Wainwright et al., 2003b), this approximate parameter estimate uniquely defines the Markov random field for which the empirical marginals $\widehat{\mu}_s$ and $\widehat{\mu}_{st}$ are fixed points of the ordinary belief propagation algorithm. We note that a parameter estimator of this type has been used previously by other researchers (Freeman et al., 2000; Ross and Kaebling, 2005). In the prediction step, we then use the ordinary belief propagation algorithm (i.e., again with $\rho_{st} = 1$) to compute approximate marginals of the graphical model with parameter $\widehat{\theta} + \gamma$. Finally, based on these approximate BP marginals, we compute the approximate predictor using Equation (24).

Although our methods are more generally applicable, here we show representative results for $m = 2$ components, and two different types of Gaussian mixtures.

(a) Mixture ensemble A is bimodal, with components $(\nu_0, \sigma_0^2) = (-1, 0.5)$ and $(\nu_1, \sigma_1^2) = (1, 0.5)$.

(b) Mixture ensemble B was constructed with mean and variance components $(\nu_0, \sigma_0^2) = (0, 1)$ and $(\nu_1, \sigma_1^2) = (0, 9)$; these choices serve to mimic heavy-tailed behavior.

In both cases, each mixture component is equally weighted; see Figure 3 for histograms of the resulting mixture ensembles.



(a) (b)

Figure 3: Histograms of different Gaussian mixture ensembles. (a) Ensemble A: a bimodal ensemble with $(\nu_0, \sigma_0^2) = (-1, 0.5)$ and $(\nu_1, \sigma_1^2) = (1, 0.5)$. (b) Ensemble B: mimics a heavy-tailed distribution, with $(\nu_0, \sigma_0^2) = (0, 1)$ and $(\nu_1, \sigma_1^2) = (0, 9)$.

Here we show results for a 2-D grid with $N = 64$ nodes. Since the mixture variables have $m = 2$ states, the coupling distribution can be written as

$$p(x; \theta^*) \propto \exp\Big\{ \sum_{s \in V} \theta_s^* x_s + \sum_{(s,t) \in E} \theta_{st}^* x_s x_t \Big\},$$

where $x \in \{-1, +1\}^N$ are "spin" variables indexing the mixture components. In all trials (except those in Section 7.2), we chose $\theta_s^* = 0$ for all nodes $s \in V$, which ensures uniform marginal distributions $p(x_s; \theta^*) = [0.5 \;\; 0.5]^T$ at each node. We tested two types of coupling in the underlying Markov random field:

(a) In the case of *attractive coupling*, for each coupling strength $\beta \in [0, 1]$, we chose edge parameters as $\theta_{st}^* \sim \mathcal{U}[0, \beta]$.

(b) In the case of *mixed coupling*, for each coupling strength $\beta \in [0, 1]$, we chose edge parameters as $\theta_{st}^* \sim \mathcal{U}[-\beta, \beta]$.

Here $\mathcal{U}[a, b]$ denotes a uniform distribution on the interval $[a, b]$. In all cases, we varied the SNR parameter α, as specified in the observation model (21), in the interval $[0, 1]$.

7.2 Comparison between "Incorrect" and True Model

We begin with an experimental comparison to substantiate our earlier claim that applying an approximate message-passing algorithm to the "incorrect" model yields prediction results superior to

those obtained by applying the same message-passing algorithm to the true underlying model. As discussed earlier in Section 6.3.1, for any underlying model $p(x;\theta^*)$ in which approximate message-passing yields the incorrect marginals (without any additional observations), there exists a range of SNR around $\alpha \approx 0$ for which this superior performance will hold.

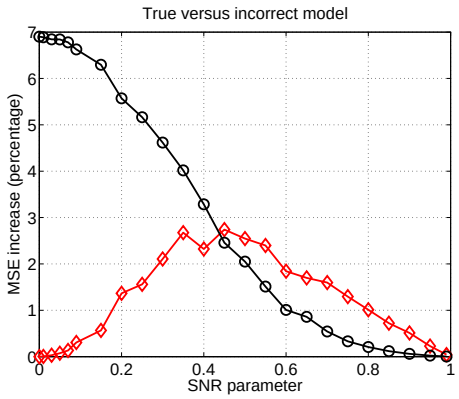


Figure 4: Line plots of percentage increase in MSE relative to Bayes optimum for the TRW method applied to the true model (black circles) versus the approximate model (red diamonds) as a function of observation SNR for grids with $N = 64$ nodes, and attractive coupling $\beta = 0.70$. As predicted by theory, using the "incorrect" model leads to superior performance, when prediction is performed using the approximate TRW method, for a range of SNR.

Figure 4 provides an empirical demonstration of this claim, when the TRW algorithm for prediction is applied to a grid with $N = 64$ nodes and attractive coupling strength $\beta = 0.70$, and the node observations chosen randomly as $\theta_s^* \sim N(0, 0.5)$. Plotted versus the SNR parameter α is the percentage increase in MSE performance relative to the Bayes optimal baseline. Note that for all SNR parameters up to $\alpha \approx 0.40$, applying the TRW algorithm to the true model yields worse performance than applying it to the "incorrect model". Beyond this point, the pattern reverses, but any differences between the two methods are rather small for $\alpha > 0.40$.

7.3 Comparison between Tree-reweighted and Ordinary Sum-product

We now compare the performance of the prediction method based on tree-reweighted sum-product (TRW) message-passing to that based on ordinary sum-product or belief propagation (BP) message-passing. Shown in Figure 5 are 2-D surface plots of the average percentage increase in MSE, taken over 100 trials, as a function of the coupling strength $\beta \in [0,1]$ and the observation SNR parameter $\alpha \in [0,1]$ for the independence model (left column), BP approach (middle column) and TRW method (right column). The top two rows show performance for attractive coupling, for mixture ensemble A ((a) through (c)) and ensemble B ((d) through (f)), whereas the bottom two row show performance for mixed coupling, for mixture ensemble A ((g) through (i)) and ensemble B ((j) through (l)).

First, observe that for weakly coupled problems ($\beta \approx 0$), whether attractive or mixed coupling, all three methods—including the independence model—perform quite well, as should be expected

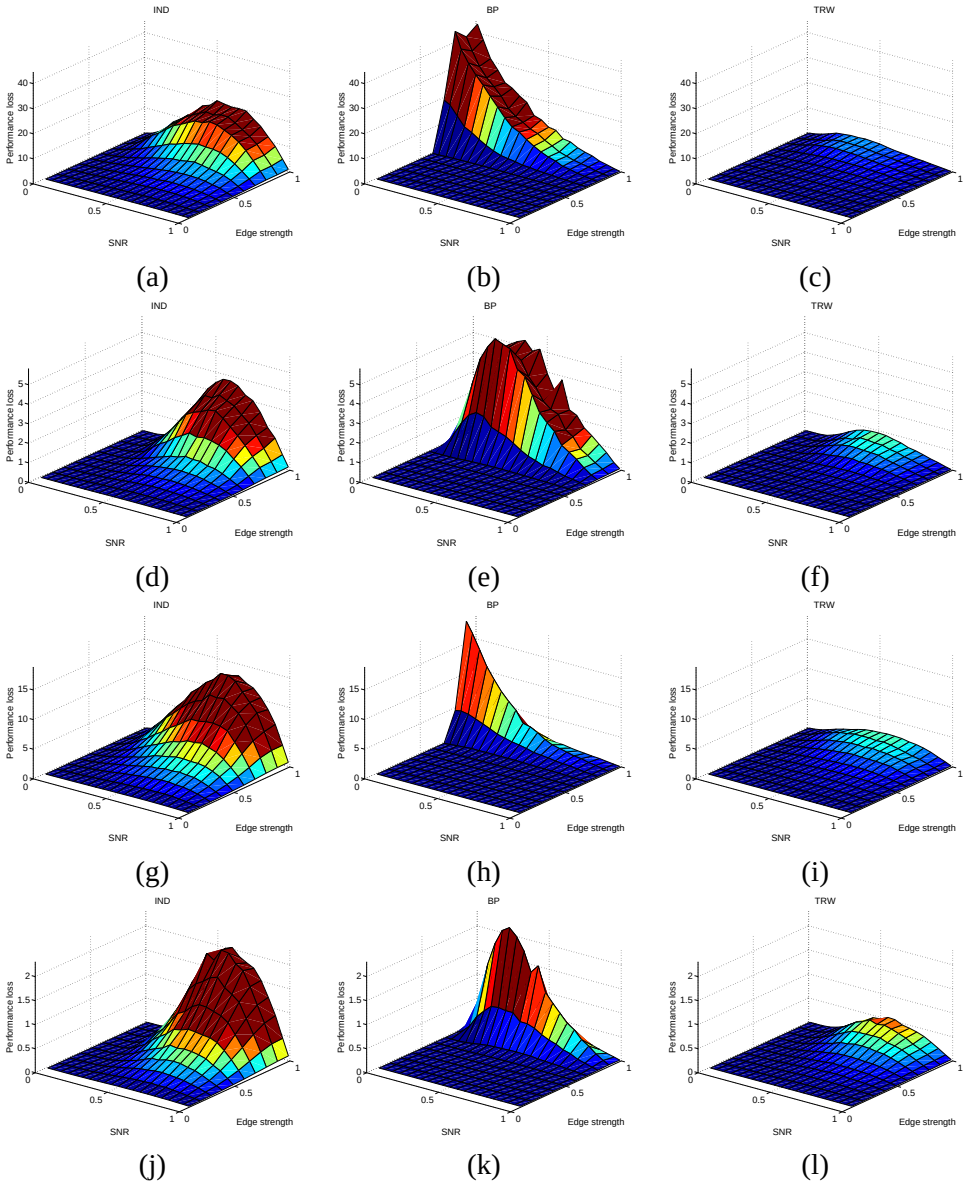


Figure 5: Surface plots of the percentage increase in MSE relative to Bayes optimum for different methods as a function of observation SNR for grids with $N = 64$ nodes. Left column: independence model (IND). Center column: ordinary belief propagation (BP). Right column: tree-reweighted algorithm (TRW). First row: Attractive coupling and a Gaussian mixture with components $(\nu_0, \sigma_0^2) = (-1, 0.5)$ and $(\nu_1, \sigma_1^2) = (1, 0.5)$. Second row: Attractive coupling and a Gaussian mixture with components $(\nu_0, \sigma_0^2) = (0, 1)$ and $(\nu_0, \sigma_1^2) = (0, 9)$. Third row: Mixed coupling and a Gaussian mixture with components $(\nu_0, \sigma_0^2) = (-1, 0.5)$ and $(\nu_1, \sigma_1^2) = (1, 0.5)$. Fourth row: Mixed coupling and a Gaussian mixture with components $(\nu_0, \sigma_0^2) = (0, 1)$ and $(\nu_0, \sigma_1^2) = (0, 9)$.

given the weak dependency between different nodes in the Markov random field. Although not clear in these plots, the standard BP method outperforms the TRW-based method for weak coupling; however, both methods lose less than 1% in this regime. As the coupling is increased, the BP method eventually deteriorates quite seriously; indeed, for large enough coupling and low/intermediate SNR, its performance can be worse than the independence (IND) model. This deterioration is particularly severe for the case of mixture ensemble A with attractive coupling, where the percentage loss in BP can be as high as 50%. Note that the degradation is *not* caused by failure of the BP algorithm to converge. Rather, by looking at alternative models (in which phase transitions are known), we have found that this type of rapid degradation coincides with the appearance of multiple fixed points for the BP algorithm. In contrast, the behavior of the TRW method is extremely stable, which is consistent with our theoretical results.

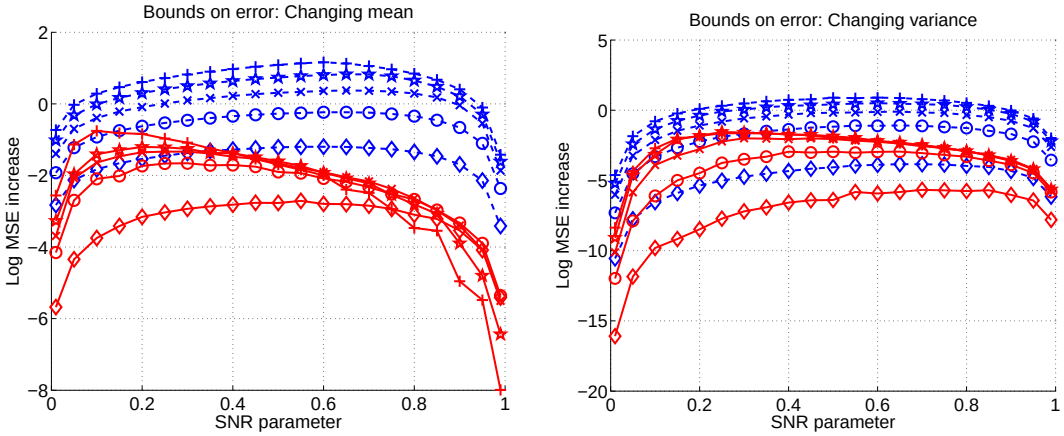
7.4 Comparison between Theory and Practice

We now compare the practical behavior of the tree-reweighted sum-product algorithm to the theoretical predictions from Theorem 7. In general, we have found that in quantitative terms, the bounds (27) are rather conservative—in particular, the TRW sum-product method performs much better than the bounds would predict. However, here we show how the bounds can capture qualitative aspects of the MSE increase in different regimes.

Figure 6 provides plots of the actual MSE increase for the TRW algorithm (solid red lines), compared to the theoretical bound (27) (dotted blue lines), for the grid with $N = 64$ nodes, and attractive coupling of strength $\beta = 0.70$. For all comparisons in both panels, we used $L = 0.10$, which numerical calculations showed to be a reasonable choice for this coupling strength. (Overall, changes in the constant L primarily cause the bounds to shift up and down on the log scale, and so do not overly affect the qualitative comparisons given here.) Panel (a) provides the comparison ensembles of type A, with fixed variances $\sigma_0^2 = \sigma_1^2 = 0.5$ and mean vectors (ν_0, ν_1) ranging from $(-0.5, 0.5)$ to $(-2.5, 2.5)$. Note how the bounds capture the qualitative behavior for low SNR, for which the difficulty of the problem increases as the mean separation is increased. In contrast, in the high SNR regime, the bounds are extremely conservative, and fail to predict that the sharp drop-off in error as the SNR parameter α approaches one. This drop-off is particularly pronounced for the ensemble with largest mean separation (marked with $+$). Panel (b) provides a similar comparison for ensembles of type B, with fixed mean vectors $\nu_0 = \nu_1 = 0$, and variances (σ_0^1, σ_1^2) ranging from $(1, 1.25)$ to $(1, 25)$. In this case, although the bounds are still very conservative in quantitative terms, they reasonably capture the qualitative behavior of the error over the full range of SNR.

8. Discussion

Key challenges in the application of Markov random fields include the estimation (learning) of model parameters, and performing prediction using noisy samples (e.g., smoothing, interpolation, denoising). Both of these problems present substantial computational challenges for general Markov random fields. In this paper, we have described and analyzed methods for joint estimation and prediction that are based on convex variational methods. Our central result is that using inconsistent parameter estimators can be beneficial in the computation-limited setting. Indeed, our results provide rigorous confirmation of the fact that using parameter estimates that are "systematically incorrect" is helpful in offsetting the error introduced by using an approximate method during the prediction step. Moreover, our analysis establishes an additional benefit—aside from the obvious one of ensuring



(a) (b)

Figure 6: Comparison of actual MSE increase and upper bounds for grid with $N = 64$ nodes with attractive coupling. (a) Equal variances $\sigma_0^2 = \sigma_1^2 = 0.5$, and mean vectors (ν_0, ν_1) ranging from $(-0.5, 0.5)$ to $(-2.5, 2.5)$. (b) Equal mean vectors $\nu_0 = \nu_1 = 0$, and variances (σ_0^2, σ_1^2) ranging from $(1, 1.25)$ to $(1, 25)$.

unique global optima—to using variational methods based on convex approximations. In particular, we established a global Lipschitz stability property that applies to any message-passing algorithm that is based on a strongly concave entropy approximation. This type of global stability is a key consideration for algorithms that are applied to models estimated from data, or in which errors might be introduced during message-passing (e.g., due to quantization or other forms of communication constraints). Our empirical results showed that a joint prediction/estimation method using the tree-reweighted sum-product algorithm yields good performance across a wide range of experimental conditions. Although our work has focused on a particular scenario, we suspect that similar ideas and techniques will be useful in related applications of approximate methods for learning combined with prediction and/or classification.

Acknowledgments

We thank the reviewers for helpful comments and suggestions to improve the initial draft of this manuscript. We would like to acknowledge support for this project from the National Science Foundation (NSF Grant DMS-0528488), an Alfred P. Sloan Foundation Fellowship, an Okawa Foundation Research Fellowship, and an Intel Corporation Equipment Grant.

Appendix A. Tree-Based Relaxation

As an illustration on the single cycle on 3 vertices, the pseudomarginal vector with elements

$$\tau_s(x_s) = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad \text{for } s = 1,2,3 \quad \text{and} \quad \tau_{st}(x_s, x_t) = \begin{bmatrix} \alpha_{st} & 0.5 - \alpha_{st} \\ 0.5 - \alpha_{st} & \alpha_{st} \end{bmatrix}$$

belongs to $\mathrm{LOCAL}_\phi(G)$ for all choices $\alpha_{st} \in [0, 0.5]$, but fails to belong to $\mathrm{MARG}_\phi(G)$, for instance, when $\alpha_{12} = \alpha_{23} = \alpha_{13} = 0$.

Appendix B. Proof of Lemma 4

Using Lemma 1 and the mean value theorem, we write

$$\begin{aligned} \mu(\theta+\delta) - \mu(\theta) &= \nabla A(\theta+\delta) - \nabla A(\theta) \\ &= \nabla^2 A(\theta + t\delta)\delta \end{aligned}$$

for some $t \in (0,1)$. Hence, it suffices to show that the eigenspectrum of the Hessian $\nabla^2 A(\theta) = \mathrm{cov}_\theta\{\phi(X)\}$ is uniformly bounded above by $L < +\infty$. The functions ϕ are all 0-1 valued indicator functions, so that the diagonal elements of $\mathrm{cov}_\theta\{\phi(X)\}$ are bounded above—in particular, $\mathrm{var}(\phi_\alpha(X)) \leq \frac{1}{4}$ for any index $\alpha \in \{1, \ldots, d\}$. Consequently, we have

$$\lambda_{\max}(\mathrm{cov}_\theta\{\phi(X)\}) \leq \sum_{\alpha=1}^{d} \lambda_\alpha(\mathrm{cov}_\theta\{\phi(X)\} = \mathrm{trace}(\mathrm{cov}_\theta\{\phi(X)\} = \frac{d}{4}$$

as required.

Appendix C. Proof of Lemma 6

Consider a spanning tree T of G with edge set $E(T)$. Given a vector $\tau \in \mathrm{LOCAL}_\phi(G)$, we associate with T a subvector $\tau(T)$ formed by those components of τ associated with vertices V and edges $E(T)$. Note that by construction $\tau(T) \in \mathrm{LOCAL}_\phi(T) = \mathrm{MARG}_\phi(T)$. The mapping $\tau \mapsto \tau(T)$ can be represented by a projection matrix $\Pi^T \in \mathbb{R}^{d(T) \times d}$ with the block structure

$$\Pi^T := \begin{bmatrix} I_{d(T)\times d(T)} & 0_{d(T)\times(d-d(T))} \end{bmatrix}.$$

In this definition, we are assuming for convenience that τ is ordered such that the $d(T)$ components corresponding to the tree T are placed first. With this notation, we have $\Pi^T \tau = \begin{bmatrix} \tau(T) & 0 \end{bmatrix}'$.

By our construction of the function B_ρ, there exists a probability distribution $\rho := \{\rho(T) \mid T \in \mathfrak{T}\}$ such that $B_\rho(\tau) = \sum_{T \in \mathfrak{T}} \rho(T) A^*(\tau(T))$, where $A^*(\tau(T))$ denotes the negative entropy of the tree-structured distribution defined by the vector of marginals $\tau(T)$. Hence, the Hessian of B_ρ has the decomposition

$$\nabla^2 B_\rho(\tau) = \sum_{T \in \mathfrak{T}} \rho(T) (\Pi^T)' \nabla^2 A^*(\tau(T)) (\Pi^T). \tag{33}$$

To check dimensions of the various quantities, note that $\nabla^2 A^*(\tau(T))$ is a $d(T) \times d(T)$ matrix, and recall that each matrix $\Pi^T \in \mathbb{R}^{d(T)\times d}$.

Now by Lemma 4, the eigenvalues of the $\nabla^2 A$ are uniformly bounded above; hence, the eigenvalues of $\nabla^2 A^*$ are uniformly bounded away from zero. Hence, for each tree T, there exists a constant C_T such that for all $z \in \mathbb{R}^d$

$$z'(\Pi^T)' \nabla^2 A^*(\tau(T))(\Pi^T) z \geq C_T \|\Pi^T z\|^2 = C_T \|z_T\|^2,$$

where we use $z_T = \Pi^T z$ as a shorthand for the projection of z onto the indices associated with T. Substituting this relation into our decomposition (33) and expanding the sum over T yields

$$\begin{aligned} z'\nabla^2 B_\rho(\tau) z &\geq \sum_{T \in \mathfrak{T}} \rho(T) C_T \|z_T\|^2 \\ &= \Big[\sum_{T \in \mathfrak{T}} \rho(T) C_T\Big] \sum_{s \in V} \|z_s\|^2 + \sum_{(s,t) \in E} \Big[\sum_{T \in \mathfrak{T}} \rho(T) C_T \mathbb{I}[(s,t) \in E(T)]\Big] \|z_{st}\|^2. \end{aligned} \quad (34)$$

Defining $C^* := \min_{T \in \mathfrak{T}} C_T$, we have the lower bounds

$$\begin{aligned} \Big[\sum_{T \in \mathfrak{T}} \rho(T) C_T\Big] &\geq C^* \sum_{T \in \mathfrak{T}} \rho(T) = C^* > 0 \\ \sum_{T \in \mathfrak{T}} \rho(T) C_T \mathbb{I}[(s,t) \in E(T)] &\geq C^* \sum_{T \in \mathfrak{T}} \rho(T) \mathbb{I}[(s,t) \in E(T)] = C^* \rho_{st} \geq C^* \rho^* > 0, \end{aligned}$$

where $\rho^* := \min_{(s,t) \in E} \rho_{st} > 0$. Applying these bounds to Equation (34) yields the final inequality

$$z'\nabla^2 B_\rho(\tau) z \geq C^* \rho^* \|z\|^2 \quad \forall z \in \mathbb{R}^d$$

with $C^* \rho^* > 0$, which establishes that the eigenvalues of $\nabla^2 B_\rho(\tau)$ are bounded away from zero.

Appendix D. Form of Exponential Parameter

Consider the observation model $y_s = \alpha z_s + \sqrt{1-\alpha^2} v_s$, where $v_s \sim N(0,1)$ and z_s is a mixture of two Gaussians (ν_0, σ_0^2) and (ν_1, σ_1^2). Conditioned on the value of the mixing indicator $X_s = j$, the distribution of y_s is Gaussian with mean $\alpha \nu_j$ and variance $\alpha^2 \sigma_j^2 + (1-\alpha^2)$.

Let us focus on one component $p(y_s \,|\, x_s)$ in the factorized conditional distribution $p(y \,|\, x) = \prod_{s=1}^n p(y_s \,|\, x_s)$. For $j = 0, 1$, it has the form

$$p(y_s \,|\, X_s = j) = \frac{1}{\sqrt{2\pi\big[\alpha^2\sigma_j^2 + (1-\alpha^2)\big]}} \exp\Big\{-\frac{1}{2\big[\alpha^2\sigma_j^2 + (1-\alpha^2)\big]}(y_s - \alpha\nu_j)^2\Big\}.$$

We wish to represent the influence of this term on x_s in the form $\exp(\gamma_s x_s)$ for some exponential parameter γ_s. We see that γ_s should have the form

$$\begin{aligned} \gamma_s &= \log p(y_s \,|\, X_s = 1) - \log p(y_s \,|\, X_s = 0) \\ &= \frac{1}{2} \log \frac{\big[\alpha^2\sigma_0^2 + (1-\alpha^2)\big]}{\big[\alpha^2\sigma_1^2 + (1-\alpha^2)\big]} + \frac{(y_s - \alpha\nu_0)^2}{2\big[\alpha^2\sigma_0^2 + (1-\alpha^2)\big]} - \frac{(y_s - \alpha\nu_1)^2}{2\big[\alpha^2\sigma_1^2 + (1-\alpha^2)\big]}. \end{aligned}$$

References

A. Benveniste, M. Metivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, New York, NY, 1990.

D.P. Bertsekas. *Nonlinear programming*. Athena Scientific, Belmont, MA, 1995.

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

J. Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, 64(3): 616–618, 1977.

L.D. Brown. *Fundamentals of statistical exponential families*. Institute of Mathematical Statistics, Hayward, CA, 1986.

M.S. Crouse, R.D. Nowak, and R.G. Baraniuk. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Trans. Signal Processing*, 46:886–902, April 1998.

M. Deza and M. Laurent. *Geometry of Cuts and Metric Embeddings*. Springer-Verlag, New York, 1997.

W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *Intl. J. Computer Vision*, 40(1):25–47, 2000.

W. T. Freeman and Y. Weiss. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Trans. Info. Theory*, 47:736–744, 2001.

T. Heskes, K. Albers, and B. Kappen. Approximate inference and constrained optimization. In *Uncertainty in Artificial Intelligence*, volume 13, pages 313–320, July 2003.

A. Ihler, J. Fisher, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, May 2005.

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

M. A. R. Leisink and H. J. Kappen. Learning in higher order Boltzmann machines using linear response. *Neural Networks*, 13:329–335, 2000.

J. S. Liu. *Monte Carlo strategies in Scientific Computing*. Springer-Verlag, New York, NY, 2001.

T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, January 2001.

J. M. Mooij and H. J. Kappen. On the properties of the Bethe approximation and loopy belief propagation on binary networks. *Journal of Statistical Mechanics: Theory and Experiment*, P11012: 407–432, 2005a.

J. M. Mooij and H. J. Kappen. Sufficient conditions for convergence of loopy belief propagation. Technical Report arxiv:cs.IT:0504030, University of Nijmegen, April 2005b. Submitted to IEEE Trans. Info. Theory.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

T. Richardson and R. Urbanke. The capacity of low-density parity check codes under message-passing decoding. *IEEE Trans. Info. Theory*, 47:599–618, February 2001.

B. D. Ripley. *Spatial statistics*. Wiley, New York, 1981.

C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer-Verlag, New York, NY, 1999.

G. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970.

M. Ross and L. P. Kaebling. Learning static object segmentation from motion segmentation. In *20th National Conference on Artificial Intelligence*, 2005.

P. Rusmevichientong and B. Van Roy. An analysis of turbo decoding with Gaussian densities. In *NIPS 12*, pages 575–581. MIT Press, 2000.

R. J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley Series in Probability and Statistics. Wiley, 1980.

C. Sutton and A. McCallum. Piecewise training of undirected models. In *Uncertainty in Artificial Intelligence*, July 2005.

S. Tatikonda. Convergence of the sum-product algorithm. In *Information Theory Workshop*, April 2003.

S. Tatikonda and M. I. Jordan. Loopy belief propagation and Gibbs measures. In *Proc. Uncertainty in Artificial Intelligence*, volume 18, pages 493–500, August 2002.

Y. W. Teh and M. Welling. On improving the efficiency of the iterative proportional fitting procedure. In *Workshop on Artificial Intelligence and Statistics*, 2003.

D. M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical analysis of finite mixture distributions*. Wiley, New York, 1986.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. Info. Theory*, 49(5):1120–1146, May 2003a.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudomoment matching. In *Workshop on Artificial Intelligence and Statistics*, January 2003b.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Trans. Info. Theory*, 51(7):2313–2335, July 2005.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical report, UC Berkeley, Department of Statistics, No. 649, September 2003.

M. J. Wainwright and M. I. Jordan. A variational principle for graphical models. In *New Directions in Statistical Signal Processing*. MIT Press, Cambridge, MA, 2005.

M. J. Wainwright and M. I. Jordan. Log-determinant relaxation for approximate inference in discrete Markov random fields. *IEEE Trans. Signal Processing*, 54(6):2099–2109, June 2006.

Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.

W. Wiegerinck. Approximations with reweighted generalized belief propagation. In *Workshop on Artificial Intelligence and Statistics*, January 2005.

J. Yedidia. An idiosyncratic journey beyond mean field theory. In M. Opper and D. Saad, editors, *Advanced mean field methods: Theory and practice*, pages 21–36. MIT Press, 2001.

J.S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Trans. Info. Theory*, 51(7):2282–2312, July 2005.

L. Younes. Estimation and annealing for Gibbsian fields. *Ann. Inst. Henri Poincare*, 24(2):269–294, 1988.

Journal of Machine Learning Research 7 (2006) 1861-1885 Submitted 8/05; Revised 6/06; Published 9/06

Streamwise Feature Selection

Jing Zhou JINGZHOU@SEAS.UPENN.EDU
Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104, USA

Dean P. Foster FOSTER@WHARTON.UPENN.EDU
Robert A. Stine STINE@WHARTON.UPENN.EDU
Statistics Department
University of Pennsylvania
Philadelphia, PA 19104, USA

Lyle H. Ungar UNGAR@CIS.UPENN.EDU
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA

Editor: Isabelle Guyon

Abstract

In *streamwise feature selection*, new features are sequentially considered for addition to a predictive model. When the space of potential features is large, streamwise feature selection offers many advantages over traditional feature selection methods, which assume that all features are known in advance. Features can be generated dynamically, focusing the search for new features on promising subspaces, and overfitting can be controlled by dynamically adjusting the threshold for adding features to the model. In contrast to traditional forward feature selection algorithms such as stepwise regression in which at each step all possible features are evaluated and the best one is selected, streamwise feature selection only evaluates each feature once when it is generated. We describe information-investing and α-investing, two adaptive complexity penalty methods for streamwise feature selection which dynamically adjust the threshold on the error reduction required for adding a new feature. These two methods give false discovery rate style guarantees against overfitting. They differ from standard penalty methods such as AIC, BIC and RIC, which always drastically over- or under-fit in the limit of infinite numbers of non-predictive features. Empirical results show that streamwise regression is competitive with (on small data sets) and superior to (on large data sets) much more compute-intensive feature selection methods such as stepwise regression, and allows feature selection on problems with millions of potential features.

Keywords: classification, stepwise regression, multiple regression, feature selection, false discovery rate

1. Introduction

In many predictive modeling tasks, one has a fixed set of observations from which a vast, or even infinite, set of potentially predictive features can be computed. Of these features, often only a small number are expected to be useful in a predictive model. Pairwise interactions and data transformations of an original set of features are frequently important in obtaining superior statistical models,

but expand the number of feature candidates while leaving the number of observations constant. For example, in a recent bankruptcy prediction study (Foster and Stine, 2004b), pairwise interactions between the 365 original candidate features led to a set of over $67,000$ resultant candidate features, of which about 40 proved to be significant. The feature selection problem is to identify and include features from a candidate set with the goal of building a statistical model with minimal out-of-sample (test) error. As the set of potentially predictive features becomes ever larger, careful feature selection to avoid overfitting and to reduce computation time becomes ever more critical.

In this paper, we describe *streamwise feature selection*, a class of feature selection methods in which features are considered sequentially for addition to a model, and either added to the model or discarded, and two simple streamwise regression algorithms[1], information-investing and α-investing, that exploit the streamwise feature setting to produce simple, accurate models. Figure 1 gives the basic framework of streamwise feature selection. One starts with a fixed set of y values (for example, labels for observations), and each potential feature is sequentially tested for addition to a model. The threshold on the required benefit (for example, error or entropy reduction, or statistical significance) for adding new features is dynamically adjusted in order to optimally control overfitting.

Streamwise regression should be contrasted with "batch" methods such as *stepwise* regression or support vector machines (SVMs). In stepwise regression, there is no order on the features; all features must be known in advance, since all features are evaluated at each iteration and the best feature is added to the model. Similarly, in SVMs or neural networks, all features must be known in advance. (Overfitting in these cases is usually avoided by regularization, which leaves all features in the model, but shrinks the weights towards zero.) In contrast, in streamwise regression, since potential features are tested one by one, they can be generated dynamically.

By modeling the candidate feature set as a dynamically generated stream, we can handle candidate feature sets of unknown, or even infinite size, since not all potential features need to be generated and tested. Enabling selection from a set of features of unknown size is useful in many settings. For example, in statistical relational learning (Jensen and Getoor, 2003; Dzeroski et al., 2003; Dzeroski and Lavrac, 2001), an agent may search over the space of SQL queries to augment the base set of candidate features found in the tables of a relational database. The number of candidate features generated by such a method is limited by the amount of CPU time available to run SQL queries. Generating 100,000 features can easily take 24 CPU hours (Popescul and Ungar, 2004), while millions of features may be irrelevant due to the large numbers of individual words in text. Another example occurs in the generation of transformations of features already included in the model (for example, pairwise or cubic interactions). When there are millions or billions of potential features, just generating the entire set of features (for example, cubic interactions or three-way table merges in SQL) is often intractable. Traditional regularization and feature selection settings assume that all features are pre-computed and presented to a learner before *any* feature selection begins. Streamwise regression does not.

Streamwise feature selection can be used with a wide variety of models where p-values or similar measures of feature significance are generated. We evaluate streamwise regression using

1. The algorithms select features and add these features into regression models. Since feature selection and regression are closely coupled here, we use "streamwise feature selection" and "streamwise regression" interchangeably. Some papers use the terms "regression" for continuous responses and "classification" for categorical responses. We use "regression" for both cases, since generalized linear regression methods such as logistic regression handle categorical responses well.

```
Input: A vector of y values (for example, labels), and a stream of features x.
  {initialize}
  model = {}                  //initially no features in model
  i = 1                       // index of features
  while CPU_time_used < max_CPU_time do
    x_i ← get_next_feature()
    {Is x_i a "good" feature?}
    if fit_of(x_i, model) > threshold then
      model ← model ∪ x_i  // add x_i to the model
      decrease threshold
    else
      increase threshold
    end if
    i ← i + 1
  end while
```

Figure 1: Algorithm: general framework of streamwise feature selection. The threshold on statistical significance of a future new feature (or the entropy reduction required for adding the future new feature) is adjusted based on whether current feature was added. fit_of(x_i, model) represents a score, indicating how much adding x_i to the model improves the model. Details are provided below.

linear and logistic regression (also known as maximum entropy modeling), where a large variety of selection criteria have been developed and tested. Although streamwise regression is designed for settings in which there is some prior knowledge about the structure of the space of potential features, and the feature set size is unknown, in order to compare it with stepwise regression, we apply streamwise regression in traditional feature selection settings, that is, those of fixed feature set size. In such settings, empirical evaluation shows that, as predicted by theory, for smaller feature sets such as occur in the UCI data sets, streamwise regression produces performance competitive to stepwise regression using traditional feature selection penalty criteria including AIC (Akaike, 1973), BIC (Schwartz, 1978), and RIC (Donoho and Johnstone, 1994; Foster and George, 1994). As feature set size becomes larger, streamwise regression offers significant computational savings and higher prediction accuracy.

The ability to do feature selection well encourages the use of different transformations of the original features. For sparse data, principal components analysis (PCA) or other feature extraction methods generate new features which are often predictive. Since the number of potentially useful principal components is low, it costs very little to generate a couple different projections of the data, and to place these at the head of the feature stream. Smaller feature sets should be put first. For example, first PCA components, then the original features, and then interaction terms. Results presented below confirm the efficiency of this approach.

Features in the feature stream can be sorted by cost. If features which are cheaper to collect are placed early in the feature stream, they will be preferentially selected over redundant expensive features later in the stream. When using the resulting model for future predictions, one needs not collect the redundant expensive features.

Alternatively, features can be sorted so as to place potentially higher signal content features earlier in the feature stream, making it easier to discover the useful features. Different applications benefit from different sorting criteria. For example, sorting gene expression data on the variance of features sometimes helps (see Section 6.2). Often features come in different types (person, place, organization; noun, verb, adjective; car, boat, plane). A combination of domain knowledge and use of the different sizes of the feature sets can be used to provide a partial order on the features, and thus to take full advantage of streamwise feature selection. As described below, one can also dynamically re-order feature streams based on which features have been selected so far.

2. Traditional Feature Selection: A Brief Review

Traditional feature selection typically assumes a setting consisting of n observations and a fixed number m of candidate features. The goal is to select the feature subset that will ultimately lead to the best performing predictive model. The size of the search space is therefore 2^m, and identifying the best subset is NP-complete. Many commercial statistical packages offer variants of a greedy method, stepwise feature selection, an iterative procedure in which at each step all features are tested at each iteration, and the single best feature is selected and added to the model. Stepwise regression thus performs hill climbing in the space of feature subsets. Stepwise selection is terminated when either all candidate features have been added, or none of the remaining features lead to increased expected benefit according to some measure, such as a p-value threshold. We show below that an even greedier search, in which each feature is considered only once (rather than at every step) gives competitive performance. Variants of stepwise selection abound, including forward (adding features deemed helpful), backward (removing features no longer deemed helpful), and mixed methods (alternating between forward and backward). Our evaluation and discussion will assume a simple forward search.

There are many methods for assessing the benefit of adding a feature. Computer scientists tend to use cross-validation, where the training set is divided into several (say k) batches with equal sizes. $k-1$ of the batches are used for training while the remainder batch is used for evaluation. The training procedure is run k times so that the model is evaluated once on each of the batches and performance is averaged. The approach is computationally expensive, requiring k separate retraining steps for each evaluation. A second disadvantage is that when observations are scarce the method does not make good use of the observations. Finally, when many different models are being considered (for example, different combinations of features), there is a serious danger of overfitting when cross-validation is used. One, in effect, is selecting the model to fit the test set.

Penalized likelihood ratio methods (Bickel and Doksum, 2001) for feature selection are preferred to cross-validation by many statisticians, as they do not require multiple re-trainings of the model and they have attractive theoretical properties. Penalized likelihood can be represented as:

$$\text{score} = -2\log(\text{likelihood}) + F \times q$$

where F is a function designed to penalize model complexity, and q represents the number of features currently included in the model at a given point. The first term in the equation represents a measure of the in-sample error given the model, while the second is a model complexity penalty. Table 1 contains the definitions which we use throughout the paper. In addition, we define *ben-*

Symbol	Meaning
n	Number of observations
m	Number of candidate features
m^*	Number of beneficial features in the candidate feature set
q	Number of features currently included in a model

Table 1: Symbols used throughout the paper and their definitions.

Name	Nickname	Penalty
Akaike information criterion	AIC	2
Bayesian information criterion	BIC	$\log(n)$
risk inflation criterion	RIC	$2\log(m)$

Table 2: Different choices for the model complexity penalty F.

eficial[2] or *spurious* features as those which, if added to the current model, would or would not reduce prediction error, respectively, on a hypothetical infinite large test data set. Note that under this definition of beneficial, if two features are perfectly correlated, the first one in the stream would be beneficial and the second one spurious, as it would not improve prediction accuracy. Also, if a prediction requires an exact XOR of two features, the raw features themselves could be spurious, even though the derived XOR-feature might be beneficial. We speak of the *set of beneficial features in a stream* as those which would have improved the prediction accuracy of the model at the time they were considered for addition if all prior beneficial features had been added.

Only features that decrease the score defined in Equation (1) are added to the model. In other words, the benefit of adding the feature to the model as measured by the likelihood ratio must surpass the penalty incurred by increasing the model complexity. We focus now on choice of F. Many different functions F have been used, defining different criteria for feature selection. The most widely used of these criteria are the Akaike information criterion (AIC), the Bayesian information criterion (BIC), and the risk inflation criterion (RIC). Table 2 summarizes the penalties F used in these methods.

For exposition we find it useful to compare the different choices of F as alternative coding schemes for use in a minimum description length (MDL) criterion framework (Rissanen, 1999). In MDL, both sender and receiver are assumed to know the feature matrix and the sender wants to send a coded version of a statistical model and the residual error given the model so that the receiver can construct the response values. Equation (1) can be viewed as the length of a message encoding a statistical model (the second term in Equation (1)) plus the residual error given that model (the first term in Equation (1)). To encode a statistical model, an encoding scheme must identify which features are selected for inclusion and encode the estimated coefficients of the included features. Using the fact that the log-likelihood of the data given a model gives the number of bits to code the model residual error leads to the criteria for feature selection: accept a new feature x_i only if the change in log-likelihood from adding the feature is greater than the penalty F, that is, if

2. Some papers use the terms “useful” or “relevant”; please see Kohavi and John (1997) and Blum and Langley (1997) for a discussion and definitions of these terms. If the features were independent (orthogonal), then we could speak of “true” features, which improve prediction accuracy for a given classification method regardless of which other features are already in the model.

$2\log(P(y|\hat{y}_{x_i})) - 2\log(P(y|\hat{y}_{-x_i})) > F$ where y is the response values, $\hat{y}_{x_i}$ is the prediction when the feature x_i is added into the model, and $\hat{y}_{-x_i}$ is the prediction when x_i is not added. Different choices for F correspond to different coding schemes for the model.

Better coding schemes encode the model more efficiently; they produce a more accurate depiction of the model using fewer bits. AIC's choice of $F = 2$ corresponds to a version of MDL which uses *universal priors* for the coefficient of a feature which is added into the model (Foster and Stine, 1999). BIC's choice of $F = \log(n)$ employs more bits to encode the coefficient as the training set size grows larger. Using BIC, each zero coefficient (feature not included in the model) is coded with one bit, and each non-zero coefficient (feature included in the model) is coded with $1 + \frac{1}{2}\log(n)$ bits (all logs are base 2). BIC is equivalent to an MDL criterion which uses *spike-and-slab priors* if the number of observations n is large enough (Stine, 2004).

However, neither AIC nor BIC are valid codes for $m \gg n$. They thus are expected to perform poorly as m grows larger than n, a situation common in streamwise regression settings. We confirm this theory through empirical investigation in Section 6.2.

RIC corresponds to a penalty of $F = 2\log(m)$ (Foster and George, 1994; George, 2000). Although the criterion is motivated by a minimax argument, following Stine (2004) we can view RIC as an encoding scheme where $\log(m)$ bits encode the index of which feature is added. Using RIC, no bits are used to code the coefficients of the features that are added. This is based on the assumption that m is large, so that the $\log(m)$ cost dominates the cost of specifying the coefficients. Such an encoding is most efficient when we expect few of the m candidate features enter the model.

RIC can be problematic for streamwise feature selection since RIC requires that we know m in advance, which is often not the case (see Section 3). We are forced to guess a m, and when our guess is inaccurate, the method may be too stringent or not stringent enough. By substituting $F = 2\log(m)$ into Equation (1) and examining the resulting chi-squared hypothesis test, it can be shown that the p-value required to reject the null hypothesis must be smaller than $\frac{0.05}{m}$. In other words, RIC may be viewed as a Bonferroni p-value thresholding method. Bonferroni methods are known to be overly stringent (Benjamini and Hochberg, 1995), a problem exacerbated in streamwise feature selection applications when m should technically be chosen to be the largest number of features that might be examined. On the other hand, if m is picked to be a lower bound of the number of predictors that might be examined, then it is too small and there is increased risk that some feature will appear by chance to give significant performance improvement.

Streamwise feature selection is closer in spirit to an alternate class of feature selection methods that control the false discovery rate (FDR), the fraction of the features that are added to the model that reduce predictive accuracy (Benjamini and Hochberg, 1995). Unlike AIC, BIC and RIC, which require each potential feature to be above the same threshold, FDR methods compute p-values (here, the probability of feature increasing test error), sort the features by p-value, and then use a threshold which depends on both the total number of features considered (like RIC) and the number of features that have been added, making use of the fact that adding some features which are almost certain to reduce prediction error allows us to add other features which are more marginal, while still meeting the FDR criterion. In this paper we propose an alternative to FDR that, among other benefits, can handle infinite feature streams, and make the above claims precise.

3. Interleaving Feature Generation and Testing

In streamwise feature selection, candidate features are sequentially presented to the modeling code for potential inclusion in the model. As each feature is presented, a decision is made using an adaptive penalty scheme as to whether or not to include the feature in the model. Each feature needs be examined at most once.

The "streamwise" view supports flexible ordering on the generation and testing of features. Features can be generated dynamically based on which features have already been added to the model.[3] Note that the theory provided below is independent of the feature generation scheme used. All that is required is a method of generating features, and an estimation package which given a proposed feature for addition to the model returns a p-value for the corresponding coefficient or, more generally, the change in likelihood of the model resulting from adding the feature. One can also test the same feature more than once (as in stepwise regression), but we have not found significant benefit from doing multiple passes through the features.

New features can be generated in many ways. For example, in addition to the m original features, m^2 pairwise interaction terms can be formed by multiplying all m^2 pairs of features together. (Almost half of these features are, of course, redundant with the other half due to symmetry, and so need not be generated and tested.) We refer to the interaction terms as *generated* features; they are examples of a more general class of features formed from transformations of the original features (square root, log, *etc.*), or combinations of them including, for instance, PCA. Such strategies are frequently successful in obtaining better predictive models.

Rather than testing all possible interactions in an arbitrary order, it is generally better to initially test interactions of the features that have already been selected with themselves, then to test interactions of the selected features with the original features, and finally (if computer power permits) to test all interactions of the original features. This requires dynamic generation of the feature stream, since the first interaction terms can not be specified in advance, as they depend on which features have already been selected. (It can be the case, as in an XOR or parity problem, that interactions are significant when none of the individual component features are, but it still makes sense as a search strategy to try the smaller parts of the feature space first.)

Statistical relational learning (SRL) methods can easily generate millions of potentially predictive features as they "crawl" through a database or other relational structure and generate features by building increasingly complex compound relations or SQL queries (Popescul and Ungar, 2004). For example, when building a model to predict the journal in which an article will be published, potentially predictive features include the words in the target article itself, the words in the articles cited by the target article, the words in articles that cite articles written by the authors of the target article, and so forth.

Both stepwise regression and standard shrinkage methods require knowing all features in advance, and are poorly suited for the feature sets generated by SRL. Since stepwise regression tests all features for inclusion at each iteration, it is computational infeasible on large data sets. Even if computer speed and memory were not an issue, control of overfitting using standard penalty methods would fail. Some other strategy such as streamwise feature selection is required. Interleaving the generation of features with the assessment of model improvement allows the search over po-

3. One cannot use the coefficients of the features that were not added to the model, since streamwise regression does not include the cost of coding these coefficients, and so this *would* lead to overfitting. One can, of course, use the rejected features themselves in interaction terms, just not their coefficients.

tential features to be pruned to promising regions. A potentially intractable search thus becomes tractable.

In SRL, one searches further in those branches of a refinement graph where more component terms have proven predictive. In searching for interaction terms, one looks first for interactions or transformations of features which have proven significant. This saves the computation, and more importantly, avoids the need to take a complexity penalty for the many interaction terms which are never examined.

There are also simple ways to dynamically interleave multiple kinds of features, each of which is in its own stream. The main feature stream used in streamwise regression is dynamically constructed by taking the next feature from the sub stream which has had the highest success in having features accepted. If a previously successful stream goes long enough without having a feature selected, then other streams will be automatically tried. To assure that all streams are eventually tried, one can use a score for each stream defined as (number of features selected + a)/(number of features tried + b). The exact values of a and b do not matter, as long as both are positive. A single feature stream is used in this paper.

4. Streamwise Regression using Information-investing

Streamwise regression can be used either in an MDL setting ("information-investing") or in a statistical setting using a t or F statistic ("α-investing"). We first present streamwise regression in an information-investing setting. Information-investing (Ungar et al., 2005) is derived using a minimum description length (MDL) approach (Stine, 2004). From a coding viewpoint, we wish to transmit a message to a receiver in order to let the receiver get the response values (y), assuming that the receiver knows x. In this sense, the score in Equation (1) is the description length required to code this message. The model is then chosen that minimizes the description length. If a feature is added to the model and reduces the description length, we call this reduction the *bits_saved*. Therefore, *bits_saved* is the decrease in the bits required to code the model error minus the increase in the bits required to code the model. The coding used to calculate *bits_saved* is described in details in Section 4.3. If *bits_saved* is larger than a threshold, we add the feature to the model. The algorithm is shown in Figure 2. We set both W_0 and W_Δ to 0.5 bit in all of the experiments presented in this paper.

Information-investing allows us, for any valid coding, to have a false discovery rate (FDR) style bound, and thus to minimize the expected test error by adding as many features as possible subject to controlling the FDR bound (Zhou et al., 2005).

Streamwise regression with information-investing consists of three components:

- *Wealth Updating*: a method for adaptively adjusting the bits available to code the features which will not be added to the model.

- *Bid Selection*: a method for determining how many bits, ε_i, one is willing to spend to code the fact of not adding a feature x_i. Asymptotically, it is also the probability of adding this feature. We show below how bid selection can be done optimally by keeping track of the bits available to cover future overfitting (that is, the wealth).

- *Feature Coding*: a coding method for determining how many bits are required to code a feature for addition. We use a two part code, coding the presence or absence of the features, and then, if the feature is present, coding the sign and size of the estimated coefficient.

Input: A vector of y values (for example, labels), a stream of features x, W_0, and W_Δ.
{initialize}
model = {} //initially no features in model
$i = 1$ // index of features
$w_1 = W_0$ // initial bits available for coding
while CPU_time_used $<$ max_CPU_time **do**
  $x_i \leftarrow$ get_next_feature()
  $\varepsilon_i \leftarrow w_i/2i$ // select bid amount
  {see Section 4.3 for the calculation of *bits_saved*}
  if $bits\_saved(x_i, \varepsilon_i, model) > W_\Delta$ **then**
    model $\leftarrow$ model $\cup\, x_i$ // add x_i to the model
    $w_{i+1} \leftarrow w_i + W_\Delta$ // increase wealth
  else
    $w_{i+1} \leftarrow w_i - \varepsilon_i$ // reduce wealth
  end if
  $i \leftarrow i + 1$
end while

Figure 2: Algorithm: streamwise regression using information-investing.

4.1 Wealth Updating

The information-investing coding scheme is adjusted using the wealth, w, which represents the number of bits currently available for future overfitting. The wealth is "invested" in testing features. Wealth starts at an initial value W_0.

Each time a feature is added, it is (in expectation) likely to be a beneficial feature and lead to a decrease in the total description length, leaving more bits available to risk future overfitting. Thus, wealth is increased by W_Δ. By increasing wealth, we gain more feature selection power under the FDR bound. Our algorithm guarantees that the sum of wealth (which is increased by W_Δ) and total description length (which is decreased by more than W_Δ) is decreased. If a feature is not added to the model, ε bits is "invested" to code this fact and subtracted from wealth.

4.2 Bid Selection

The selection of ε_i as $w_i/2i$ gives the slowest possible decrease in wealth such that all wealth is used; that is, so that as many features as possible are included in the model without systematically overfitting.[4]

Theorem 1 *Computing* ε_i *as proportional to* $w_i/2i$ *gives the slowest possible decrease in wealth such that* $\lim_{i\to\infty} w_i = 0$.

4. Slightly better and more complex bid selection methods such as $\varepsilon_i \leftarrow w_i/(i\log(i))$ could be used, but they are statistically equivalent to the simpler one in terms of rates, and more importantly they generate tests that have no more power. We will stick with the simpler one in this paper.

Proof Define $\delta_i = \varepsilon_i / w_i$ to be the fraction of wealth invested at time i. If no features are added to the model, wealth at time i is $w_i = \Pi_i(1-\delta_i)$. If we pass to the limit to generate w_∞, we have $w_\infty = \Pi_i(1-\delta_i) = e^{\sum \log(1-\delta_i)} = e^{-\sum \delta_i + O(\delta_i^2)}$. Thus, $w_\infty = 0$ iff $\sum \delta_i$ is infinite.

Thus if we let δ_i go to zero faster than $1/i$, say $i^{-1-\gamma}$ where $\gamma > 0$ then $w_\infty > 0$ and we have wealth that we never use. ■

4.3 Feature Coding

To code an added feature, we code both the fact that the feature is added and the value of its estimated coefficient. Since ε is the number of bits available to code the fact of not adding a feature, the probability of not adding a feature should be $e^{-\varepsilon}$ if the coding is optimal. Therefore, the probability of adding a feature is $1 - e^{-\varepsilon} = 1 - (1 - \varepsilon + O(\varepsilon^2)) \approx \varepsilon$, and the cost in bits of coding the fact the feature is added is roughly $-\log(\varepsilon)$ bits. Different codings can be used for the feature's estimated coefficient. For example, BIC uses $\frac{1}{2}\log(n)$ bits. In section 4.3.1, we present an optimal coding of the estimated coefficients.

For now, for simplicity assume we use b bits to code each feature x's estimated coefficient $\hat{\beta}$ when x is added to the model. Adding x to the model reduces the model entropy by $\frac{1}{2}t^2\log(e)$ bits where t is the t statistic associated with $\hat{\beta}$, as defined above. Here, and below $\log()$ is log based 2; the $\log(e)$ converts the t^2 to bits. Then,

$$bits\_saved = \frac{1}{2}t^2\log(e) - (-\log(\varepsilon) + b).$$

4.3.1 OPTIMAL CODING OF COEFFICIENTS IN INFORMATION-INVESTING

A key question is what coding scheme to use to code the coefficient of a feature which is added to the model. We describe here an "optimal" coding scheme which can be used in the information-investing criterion. The key idea is that coding an event with probability p requires $\log(p)$ bits. This equivalence allows us to think in terms of distributions and thus to compute codes which handle fractions of a bit. Our goal is to find a (legitimate) coding scheme which, given a "bid" ε, will guarantee the highest probability of adding the feature to the model. We show below that given any actual distribution $\tilde{f}_\beta$ of the coefficients, we can produce a coding corresponding to a modified distribution f_β which produces a coding which uniformly dominates it.

Assume, for simplicity, that we increase the wealth by one bit when a feature x_i with coefficient β_i is added. Thus, when x_i is added, we have

$$\log \frac{p(x_i \text{ is a beneficial feature})}{p(x_i \text{ is a spurious feature})} > 1 \text{ bit},$$

that is, the log-likelihood decreases by more than one bit.

Let f_{β_i} be the distribution implied by the coding scheme for t_{β_i} if we add x_i and $f_0(t_{\beta_i})$ be the normal distribution (the null model in which x_i should not be added). The coding saves enough bits to justify adding a feature whenever $f_{\beta_i}(t_{\beta_i}) \geq 2 f_0(t_{\beta_i})$. This happens with probability $\alpha_i \equiv p_0(\{t_{\beta_i} : f_{\beta_i}(t_{\beta_i}) \geq 2 f_0(t_{\beta_i})\})$ under the null. α_i is the area under the tails of the null distribution.

There is no reason to have $f_{\beta_i}(t_{\beta_i}) \gg 2 f_0(t_{\beta_i})$ in the tails, since this would "waste" probability or bits. Hence, the optimal coding is $f_\beta(t_{\beta_i}) = 2 f_0(t_{\beta_i})$ for all the features that are likely to be

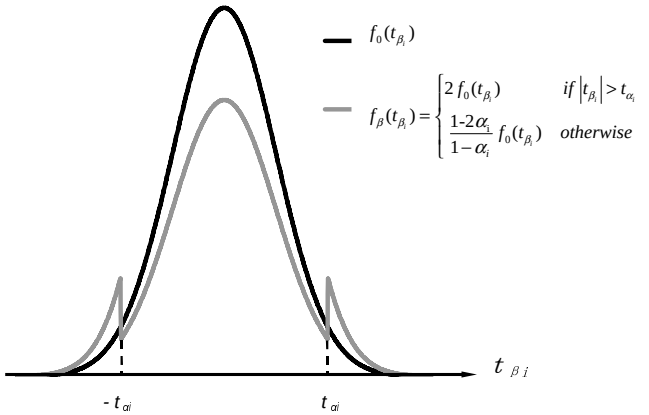


Figure 3: Optimal distribution f_β.

added. Using all of the remaining probability mass (or equivalently, making the coding "Kraft tight") dictates the coding for the case when the feature is not likely to be added. The most efficient coding to use is thus:

$$\begin{cases} f_\beta(t_{\beta_i}) = 2 f_0(t_{\beta_i}) & \text{if } |t_{\beta_i}| > t_{\alpha_i} \\ f_\beta(t_{\beta_i}) = \frac{1-2\alpha_i}{1-\alpha_i} f_0(t_{\beta_i}) & \text{otherwise} \end{cases}$$

and the corresponding cost in bits is:

$$\begin{cases} \log(f_\beta(t_{\beta_i})/f_0(t_{\beta_i})) = \log(2) = 1 \text{ bit} & \text{if } |t_{\beta_i}| > t_{\alpha_i} \\ \log(f_\beta(t_{\beta_i})/f_0(t_{\beta_i}) = \log(\frac{1-2\alpha_i}{1-\alpha_i}) \approx -\alpha_i \text{ bits} & \text{otherwise.} \end{cases}$$

Figure 3 shows the distribution $f_\beta(t(\beta_i))$, with the probability mass transferred away from the center, where features are not added, out to the tails, where features are added.

The above equations are derived assuming that 1 bit is added to the wealth. It can be generalized to add W_Δ bits to the wealth each time a feature is added to the model. Then, when a feature is added to the model the probability of it being "beneficial" should be 2^{W_Δ} times that of it being "spurious", and all of the 2's in the above equations are replaced with 2^{W_Δ}.

5. Streamwise Regression using Alpha-investing

One can define an alternate form of streamwise regression, α-investing (Zhou et al., 2005), which is phrased in terms of p-values rather than information theory. The p-value associated with a t-statistic is the probability that a coefficient of the observed size could have been estimated by chance even though the true coefficient was zero (Larsen and Marx, 2001). Of the three components of streamwise regression using information-investing, in α-investing, *wealth updating* is similar, *bid selection* is identical, and *feature coding* is not required. The two different streamwise regression algorithms are asymptotically identical (the wealth update of $\alpha_\Delta - \alpha_i$ approaches the update of W_Δ as α_i becomes small), but differ slightly when the initial features in the stream are considered. The relation between the two methods follows from the fact that coding an event with probability p requires $\log(p)$ bits. The α-investing algorithm is shown in Figure 4, and the equivalence between α-investing and information-investing is shown in Table 3. Wealth updating is now done in terms of α, the probability of adding a spurious feature.

Input: A vector of y values (for example, labels), a stream of features x, W_0, and α_Δ.

```
{initialize}
model = {}                  //initially no features in model
i = 1                       // index of features
w_1 = W_0                   // initial prob. of false positives
while CPU_time_used < max_CPU_time do
   x_i ← get_next_feature()
   α_i ← w_i/2i
   {Is p-value of the new feature below threshold?}
   if get_p-value(x_i, model) < α_i then
      model ← model ∪ x_i   // add x_i to the model
      w_{i+1} ← w_i + α_Δ − α_i // increase wealth
   else
      w_{i+1} ← w_i − α_i       // reduce wealth
   end if
   i ← i + 1
end while
```

Figure 4: Algorithm: streamwise regression with α-investing.

information-investing	**α-investing**
w_i	$\log(w_i)$
bits_saved	test statistic = Δlog-likelihood
bits_saved $> W_\Delta$	p-value $< \alpha_i$

Table 3: The equivalence of α-investing and information-investing.

α-investing controls the FDR bound by dynamically adjusting a threshold on the p-statistic for a new feature to enter the model (Zhou et al., 2005). Similarly to the information-investing, α-investing adds as many features as possible subject to the FDR bound giving the minimum out-of-sample error.

The threshold, α_i, corresponds to the probability of including a spurious feature at step i. It is adjusted using the wealth, w_i, which represents the current acceptable number of future false positives. Wealth is increased when a feature is added to the model (presumably correctly, and hence permitting more future false positives without increasing the overall FDR). Wealth is decreased when a feature is not added to the model. In order to save enough wealth to add future features, bid selection is identical to the information-investing.

More precisely, a feature is added to the model if its p-value is greater than α. The p-value is computed by using the fact that Δlog-likelihood is equivalent to a t-statistic. The idea of α-investing is to adaptively control the threshold for adding features so that when new (probably predictive) features are added to the model, one "invests" α increasing the wealth, raising the threshold, and allowing a slightly higher future chance of incorrect inclusion of features. We increase wealth by $\alpha_\Delta - \alpha_i$. Note that when α_i is very small, this increase amount is roughly equivalent to α_Δ. Each time a feature is tested and found not to be significant, wealth is "spent", reducing the threshold so as to keep the guarantee of not adding more than a target fraction of spurious features. There are

two user-adjustable parameters, α_Δ and W_0, which can be selected to control the FDR; we set both of them to 0.5 in all of the experiments presented in this paper.

6. Experimental Evaluation

We compared streamwise feature selection using α-investing against both streamwise and stepwise feature selection (see Section 2) using the AIC, BIC and RIC penalties on a battery of synthetic and real data sets. After a set of features are selected from the real data sets, we applied logistic regression on this feature set selected, calculated the probability of observation labels, provided a cutoff/threshold of 0.5 to classify the response labels if label values are binary and get the prediction accuracies or balance errors. (Actually, different cutoffs could be used for different loss functions.) Information-investing gives extremely similar results, so we do not report them. We used R to implement our evaluation.

6.1 Evaluation on Synthetic Data

The base synthetic data set contains 100 observations each of 1,000 features, of which 4 are predictive. We generated the features independently from a normal distribution, $N(0,1)$, with the true model being the sum of four of the features (their coefficients are one's)[5] plus noise, $N(0,0.1^2)$. The artificially simple structure of the data (the features are uncorrelated and have relatively strong signal) allows us to easily see which feature selection methods are adding spurious features or failing to find features that should be in the model.

The results are presented in Table 4. As expected, AIC massively overfits, always putting in as many features as there are observations. BIC overfits severely, although less badly than AIC. RIC gives performance comparable to α-investing. As one would also expect, if all of the beneficial features in the model occur at the beginning of the stream, α-investing does better, giving the same error as RIC, while if all of the beneficial features in the model are last, α-investing does (two times) worse than RIC. In practice, if one is not taking advantage of known structure of the features, one can randomize the feature order to avoid such bad performance.

Stepwise regression gave noticeably better results than streamwise regression for this problem when the penalty is AIC or BIC. Using AIC and BIC still resulted in n features being added, but at least all of the beneficial features were found. Stepwise regression with RIC gave the same error of its streamwise counterpart. However, using standard code from R, the stepwise regression was *much* slower than streamwise regression. Running stepwise regression on data sets with tens of thousands of features, such as the ones presented in Table 5, was not possible.

One might hope that adding more spurious features to the end of a feature stream would not severely harm an algorithm's performance.[6] However, AIC and BIC, since their penalty is not a function of m, will add even more spurious features (if they haven't already added a feature for every observation!). RIC (Bonferroni) produces a harsher penalty as m gets large, adding fewer and fewer features. As Table 5 and 6 show, α-investing is clearly the superior method in this case.

5. Similar results are also observed, if instead of using coefficients which are strictly 0 or 1, we use coefficients that are generated in either of the two cases: (a) most coefficients are zeros and several are from Gaussian distribution; (b) all coefficients are generated from t distribution with degree of freedom of two.

6. One would not, of course, intentionally add features known not to be predictive. However, as described above, there is often a natural ordering on features so that some classes of features, such as interactions, have a smaller fraction of predictive features, and can be put later in the feature stream.

streamwise	AIC	BIC	RIC	α-invest.	α-invest.	
					first	last
features	100	90	4.3	4.2	4.6	3.7
error	6.13	1.91	0.33	0.42	0.33	0.71
stepwise	AIC	BIC	RIC			
features	100	100	4.5	–	–	–
error	0.54	0.54	0.33	–	–	–

Table 4: AIC and BIC overfit for $m \gg n$. The number of features selected and the out-of-sample error, averaged over 20 runs. $n = 100$ observations, $m = 1,000$ features, $m^* = 4$ beneficial features in data. Synthetic data: $x \sim N(0,1)$, y is linear in x with noise $\sigma^2 = 0.1$. Beneficial features are randomly distributed in the feature set except the "first" and "last" cases. "first" and "last" denote the beneficial features being first or last in the feature stream.

m		1,000	10,000	100,000	1,000,000
RIC	features	4.3	4.0	4.0	3.4
RIC	false pos.	0.3	0.2	0.2	0.4
RIC	error	0.33	0.42	0.50	0.97
α-invest.	features	4.2	4.1	4.7	4.8
α-invest.	false pos.	0.3	0.2	0.7	0.9
α-invest.	error	0.42	0.42	0.43	0.45

Table 5: Effect of adding spurious features. The average number of features selected, false positives, and out-of-sample error (20 runs). $m^* = 4$ beneficial features, randomly distributed over the first 1,000 features. Otherwise the same model as Table 4.

Table 6 shows that when the number of potential features goes up to 1,000,000, RIC puts in one less beneficial feature, while streamwise regression puts the same four beneficial features plus a half of a spurious feature. Thus, streamwise regression is able to find the extra feature even when the feature is way out in the 1,000,000 features.

6.2 Evaluation on Real Data

Tables 7, 8, and 9 provide a summary of the characteristics of the real data sets that we used. All are for binary classification tasks. The six data sets in Table 7 were taken from the UCI repository. The seven data sets in Table 8 are bio-medical data, in which each feature represents a gene expression value for each observation (patient with cancer or healthy donor). For example, in *aml* data set, observations consist of patients with acute myeloid leukemia and patients with acute lymphoblastic leukemia. The classification task is to identify which patient has which cancer. *ha* and *hung* are private data sets and other gene expression data sets are available to the public (Li and Liu, 2002). The NIPS data sets are from the NIPS2003 workshop (Guyon, 2003).

The observations are shuffled and those observations which contain missing feature values are deleted. Since the gene expression data sets have large feature sets, we shuffled their original features five times (in addition to the cross validations), applied streamwise regression on each feature

m		1,000	10,000	100,000	1,000,000
RIC	features	4.3	4.2	3.9	3.7
RIC	false pos.	0.3	0.3	0.1	0.6
RIC	error	0.33	0.42	0.50	0.97
α invest.	features	4.2	4.2	4.5	4.9
α invest.	false pos.	0.3	0.3	0.6	0.8
α invest.	error	0.42	0.42	0.43	0.42

Table 6: Effect of adding spurious features. The average number of features selected, false positives, and out-of-sample error (20 runs). $m^* = 4$ beneficial features: when $m = 1,000$, all four beneficial features are randomly distributed; in the other three cases, there are three beneficial features randomly distributed over the first 1,000 features and another beneficial feature randomly distributed within the feature index ranges [1001, 10000], [10001, 100000], and [100001, 1000000] when $m = 10000$, 100000, and 1000000 respectively. Otherwise the same model as Table 4 and 5.

	cleve	internet	ionosphere	spect	wdbc	wpbc
features, *m*	13	1558	34	22	30	33
nominal features	7	1555	0	22	0	0
continuous features	6	3	34	0	30	33
observations, *n*	296	2359	351	267	569	194
baseline accuracy	54%	84%	64%	79%	63%	76%

Table 7: Description of the UCI data sets.

order, and averaged the five evaluation results. The baseline accuracy is the accuracy (on the whole data set) when predicting the majority class. The feature selection methods were tested on these data sets using ten-fold cross-validation.

On the UCI and gene expression data sets, experiments were done on two different feature sets. The first experiments used only the original feature set. The second interleaved feature selection and generation, initially testing PCA components and the original features, and then generating interaction terms between any of the features which had been selected and any of the original features. On the NIPS data sets, since our main concern is to compare against the challenge best models, we did only the second kind of experiment.

	aml	ha	hung	ctumor	ocancer	pcancer	lcancer
features, *m*	7,129	19,200	19,200	2,000	15,154	12,600	12,533
observations, *n*	72	83	57	62	253	136	181
baseline accuracy	65%	71%	63%	65%	64%	57%	92%

Table 8: Description of the gene expression data sets. All features are continuous.

	arcene	dexter	dorothea	gisette	madelon
features, *m*	10,000	20,000	100,000	5,000	500
observations, *n*	100	300	800	6,000	2,000
baseline accuracy	56%	50%	90%	50%	50%

Table 9: Description of the NIPS data sets. All features are nominal.

On UCI data sets (Figure 5)[7] , when only the original feature set is used, paired two-sample t-tests show that α-investing has better performance than streamwise AIC and BIC only on two of the six UCI data sets: the *internet* and *wpbc* data sets. On the other data sets, which have relatively few features, the less stringent penalties do as well as or better than streamwise regression. When interaction terms and PCA components are included, α-investing gives better performance than streamwise AIC on five data sets, than streamwise BIC on three data sets, and than streamwise RIC on two data sets. In general, when the feature set size is small, there is no significant difference in the prediction accuracies between α-investing and the other penalties. When the feature set size is larger (that is, when new features are generated) α-investing begins to show its superiority over the other penalties.

On the UCI data sets (Figure 5), we also compared streamwise regression with α-investing[8] with stepwise regression. Paired two-sample t-tests show that when the original feature set is used, α-investing does not differ significantly from stepwise regression. α-investing has better performance than stepwise regression in 5 cases, and worse performance in 3 cases. (Here a "case" is defined as a comparison of α-investing and stepwise regression under a penalty, that is, AIC or BIC or RIC, on a data set.) However, when interaction terms and PCA components are included, α-investing gives better performance than stepwise regression in 9 cases, and worse performance in none of the cases. Thus, in our tests, α-investing is comparable to stepwise regression on the smaller data sets and superior on the larger ones.

On the UCI data sets (Table 10), α-investing was also compared with support vector machines (SVM), neural networks (NNET), and decision tree models (TREE). In all cases, standard packages available with R were used[9]. No doubt these could be improved by fine tuning parameters and kernel functions, but we were interested in seeing how well "out-of-the-box" methods could do. We did not tune any parameters in streamwise regression to particular problems either. Paired two-sample t-tests show that α-investing has better performance than NNET on 3 out of 6 data sets, and than SVM and TREE on 2 out of 6 data sets. On the other data sets, streamwise regression doesn't have

7. In Figure 5, a small training set size of 50 was selected to make sure the problems were difficult enough that the methods gave clearly different results. The right columns graphs differs from the left ones in that: (1) we generated PCA components from the original data sets and put them at the front of the feature sets; (2) after the PCA component "block" and the original feature "block", there is an interaction term "block" in which the interaction terms are generated using the features selected from the first two feature blocks. This kind of feature stream was also used in the experiments on the other data sets. We were unable to compute the stepwise regression results on the internet data set using the software at hand when interaction terms and PCA components were included giving millions of potential features with thousands of observations. It is indicative of the difficulty of running stepwise regression on large data sets.
8. In later text of this section, for simplicity, we use "α-investing" to mean the "streamwise regression with α-investing".
9. Please find details at http://cran.r-project.org/doc/packages for SVM (e1071.pdf), NNET (VR.pdf), and TREE (tree.pdf). SVM uses the radial kernel. NNET uses feed-forward neural networks with a single hidden layer. TREE grows a tree by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side.

significant better or worse performance than NNET, SVM, or TREE. These tests shows that the performance of streamwise regression is at least comparable to those of SVM, NNET, and TREE.

On the gene expression data sets (Figure 6), when comparing α-investing with streamwise AIC, streamwise BIC, and streamwise RIC, paired two-sample t-tests show that when the original features are used, the performances of α-investing and streamwise RIC don't have significant difference on any of the data sets. But when interaction terms and PCA components are included , RIC is often too conservative to select even only one feature, whereas α-investing has stable performance and the t-tests show that α-investing has significant better prediction accuracies than streamwise RIC on 5 out of 7 data sets. Note that, regardless of whether or not interaction terms and PCA components are included, α-investing always has much higher accuracy than streamwise AIC and BIC.

The standard errors (SE) of prediction accuracies in shuffles gave us sense of the approach sensitivity to the feature order. When the original features are used, α-investing has a maximum SE of four percent on *pcancer* and its other SEs are less than two percents in accuracy. When PCA components and interaction terms are included, α-investing has a maximum SE of two percents on *ha* and its other SEs are around or less than one percent. Streamwise RIC has similar SEs as α-investing has, but streamwise AIC and BIC usually have one percent higher SEs than α-investing and streamwise RIC. We can see that the feature shuffles don't change the performance much on most of gene expression data sets.

When PCA components and interaction terms are included and the original feature set is sorted in advance by feature value variance (one simple way of making use of the ordering in the stream), the prediction accuracy of α-investing on *hung* is increased from 79.3% to 86.7%; for the other gene expression data, sorting gave no significant change.

Also note that, for streamwise AIC, BIC, and RIC, adding interaction terms and PCA components often hurts. In contrast, the additional features have not much effect on α-investing. With these additional features, the prediction accuracies of α-investing are improved or kept the same on 4 out of 6 UCI data sets and 5 out of 7 gene data sets.

On the gene expression data sets (Figure 6), we also compared α-investing with stepwise regression. The results show that, α-investing is competitive with stepwise regression with the RIC penalty. Stepwise regression with AIC or BIC penalties gives inferior performance.

On the NIPS data sets (Table 11), we compared α-investing against results reported on the NIPS03 competition data set using other feature selection methods (Guyon et al., 2006). Table 11 shows the results we obtained, and compares them against the two methods which did best in the competition. These methods are *BayesNN-DFT* (Neal, 1996, 2001), which combines Bayesian neural networks and Bayesian clustering with a Dirichlet diffusion tree model and *greatest-hits-one* (Gilad-Bachrach et al., 2004), which normalizes the data set, selects features using distance information, and classifies them using a perceptron or SVM.

Different feature selection approaches such as those used in *BayesNN-DFT* can be contrasted based on their different levels of greediness. *Screening* methods or filters look at the relation between y and each x_i *independently*. In a typical screen, one computes how predictive each x_i $(i = 1...m)$ is of y (or how are they correlated), or the mutual information between them, and all features above a threshold are selected. In an extension of the simple screen, FBFS (Fleuret, 2004) looks at the mutual information $I(y; x_i|x_j)$ $(i, j = 1...m)$, that is, the effect of adding a second feature

after one has been added.[10] Streamwise and stepwise feature selection are one step less greedy, sequentially adding features by computing $I(y; x_i|Model)$.

BayesNN-DFT uses a screening method to select features, followed by a sophisticated Bayesian modeling method. Features were selected using the union of three univariate significance test-based screens (Neal and Zhang, 2003): correlation of class with the ranks of feature values, correlation of class with a binary form of the feature (zero/nonzero), and a runs test on the class labels reordered by increasing feature value. The threshold was selected by comparing each to the distribution found when permuting the class labels. This richer feature set of transformed variables could, of course, be used within the streamwise feature selection setting, or streamwise regression could be used to find an initial set of features to be provided to the Bayesian model.

greatest-hits-one applied margin based feature selection on data sets *arcene* and *madelon*, and used a simple infogain ranker to select features on data sets *dexter* and *dorothea*. Assuming a fixed feature set size, a generalization error bound is proved for the margin based feature selection method (Gilad-Bachrach et al., 2004).

Table 11 shows that we mostly get comparable accuracy to the best-performing of the NIPS03 competition methods, while using a small fraction of the features. Many of the NIPS03 contestants got far worse performance, without finding small feature sets (NIPS'03, 2003). When SVM is used on the features selected by streamwise regression, the errors on *arcene*, *gisette*, and *madelon* are reduced further to 0.151, 0.021, 0.214 respectively. One could also apply more sophisticated methods, such as the Bayesian models which *BayesNN-DFT* used, to our selected features.

There is only one data set, *madelon*, where streamwise regression gives substantially higher error than the other methods. This may be partly due to the fact that madelon has substantially more observations than features, thus making streamwise regression (when not augmented with sophisticated feature generators) less competitive with more complex models. Madelon is also the only synthetic data set in the NIPS03 collection, and so its structure may benefit more from the richer models than typical real data.

7. Discussion: Statistical Feature Selection

Recent developments in statistical feature selection take into account the size of the feature space, but only allow for finite, fixed feature spaces, and do not support streamwise feature selection. The risk inflation criterion (RIC) produces a model that possesses a type of competitive predictive optimality. RIC chooses a set of features from the potential feature pool so that the loss of the resulting model is within a factor of $\log(m)$ of the loss of the best such model. In essence, RIC behaves like a Bonferroni rule (Foster and George, 1994). Each time a feature is considered, there is a chance that it will enter the model even if it is merely noise. In other words, the tested null hypothesis is that the proposed feature does not improve the prediction of the model. Doing a formal test generates a p-value for this null hypothesis. Suppose we only add this feature if its p-value is less than α_j when we consider the jth feature. Then the Bonferroni rule keeps the chance of adding even one extraneous feature to less than, say, 0.05 by constraining $\sum \alpha_j \leq 0.05$.

Bonferroni methods like RIC are conservative, limiting the ability of a model to add features that improve its predictive accuracy. The connection of RIC to α-spending rules leads to a more powerful alternative. An α-spending rule is a multiple comparison procedure that bounds its cumulative type one error rate at a small level, say 5%. For example, suppose one has to test the m hypotheses

10. FBFS has been developed only for binary features, but could be easily extended.

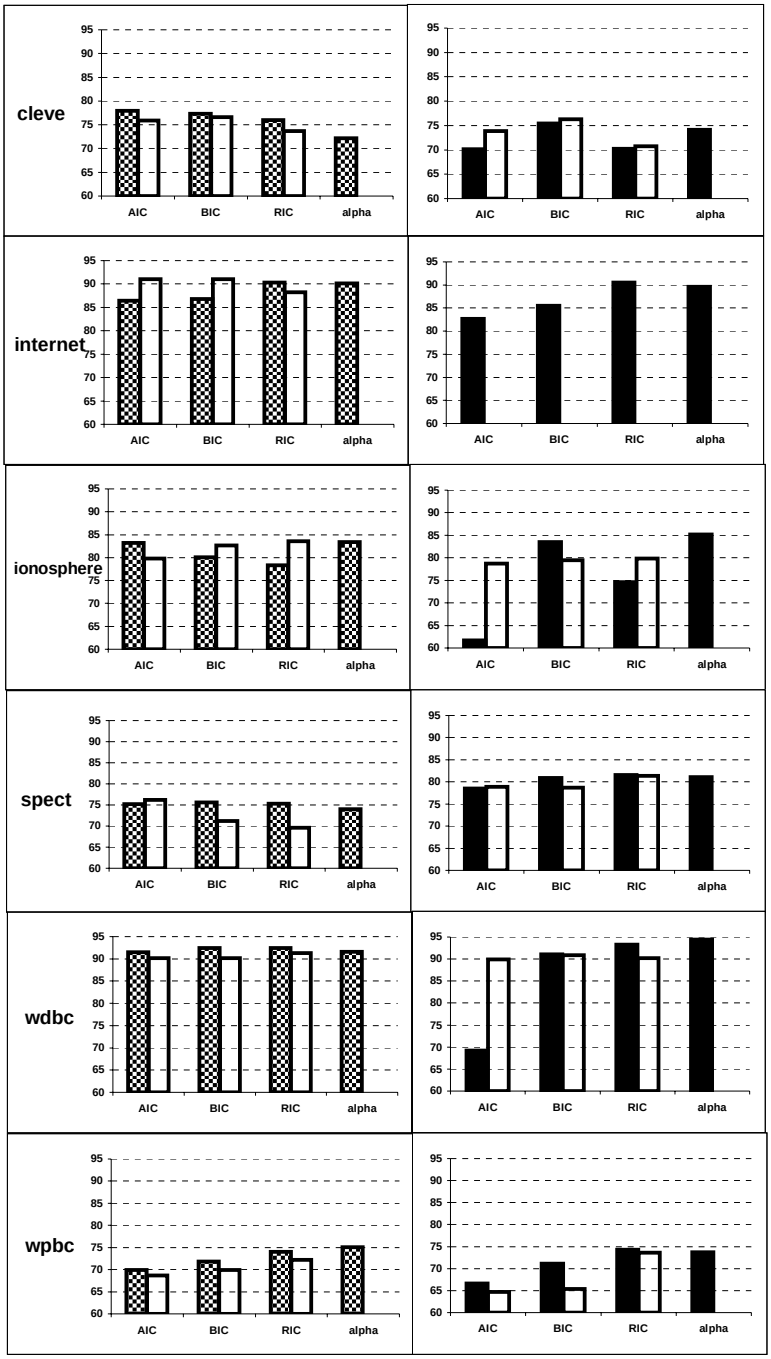


Figure 5: UCI Data Streamwise vs. Stepwise Validation Accuracy for different penalties. Training size is 50. The average accuracy is on 10 cross-validations. The black-dot and solid black bars are the average accuracies using streamwise regressions. The transparent bars are the average accuracies using stepwise regressions. Raw features are used in the left column graphs. Additional interaction terms and PCA components are used in the right column graphs. Please see Footnote 7 for additional information about this figure. Section 6.2 gives the results of paired two-sample t-tests.

	cleve	internet	ionosphere
stream. (α-invest.)	84.3±1.8 (8.5)	96.5±0.3 (166)	91.4±1.8 (23)
SVM	82.0±2.0	93.4±0.6	92.2±1.7
NNET	70.3±4.5	84.2±0.9	91.7±1.9
TREE	76.0±3.3	96.5±0.5	86.7±1.8

	spect	wdbc	wpbc
stream. (α-invest.)	82.2±2 (2)	95.1±0.6 (37)	77±3.4 (4.4)
SVM	81.5±2.4	96.3±0.8	76.5±4.9
NNET	78.9±2.2	68.8±4.5	75.0±5.0
TREE	81.1±1.9	94.2±0.9	74.0±3.0

Table 10: Comparison of streamwise regression and other methods on UCI Data. Average accuracy using 10-fold cross validation. The number before $\pm$ is the average accuracy; the number immediately after $\pm$ is the standard deviation of the average accuracy. The number in parentheses is the average number of features used by the streamwise regression, and these features includes PCA components, raw features, and interaction terms (see Footnote 7 for the details of this kind of feature stream). SVM, NNET, and TREE use the whole raw feature set. Section 6.2 gives the results of paired two-sample t-tests.

	arcene	dexter	dorothea	gisette	madelon
α-invest. error	0.176	0.067	0.090	0.037	0.295
α-invest. features	8	21	8	72	24
greatest-hits-one error	0.172	0.053	0.109	0.030	0.086
greatest-hits-one features	10,000	1,400	300	5,000	18
BayesNN-DFT error	0.133	0.039	0.085	0.013	0.072
BayesNN-DFT features	10,000	303	100,000	5,000	500

Table 11: Comparison of Streamwise regression and other methods on NIPS Data. *error* is the "balanced error"(Guyon, 2003); *features* is the number of features selected by models.

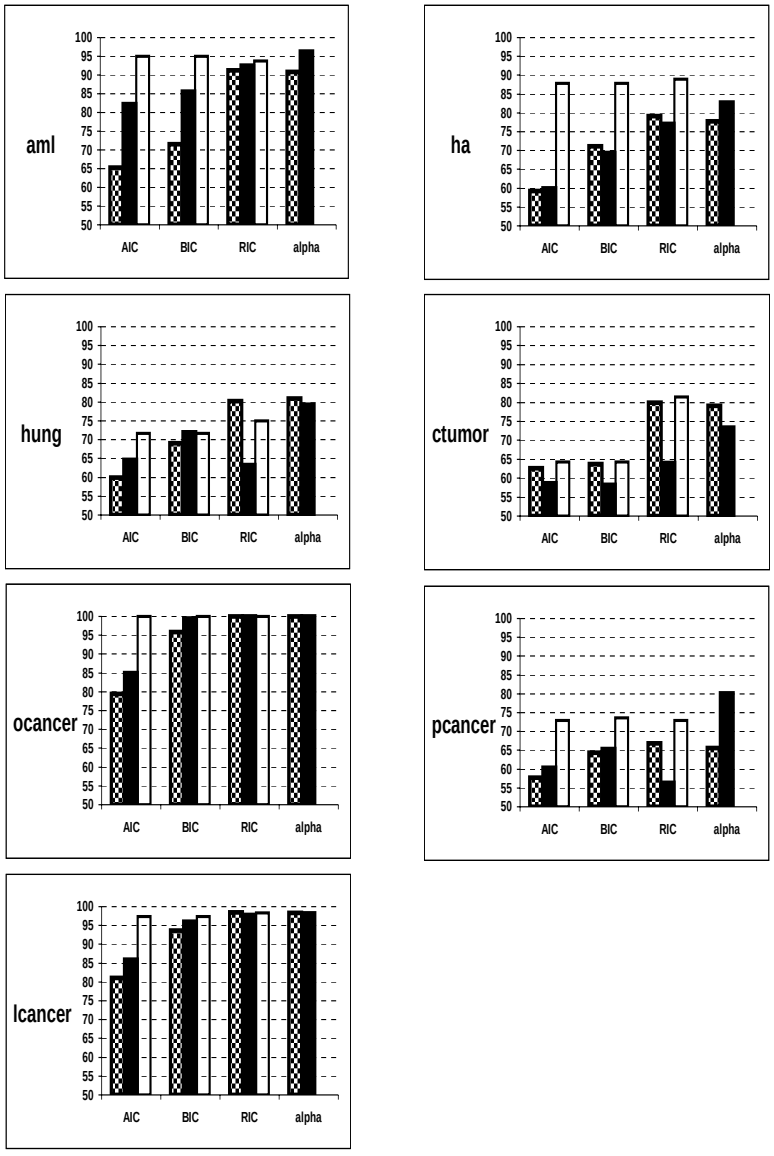


Figure 6: Gene expression data Streamwise vs. Stepwise Validation Accuracy for different penalties. Average accuracy using 10-fold cross validation. The black-dot bars are the average accuracies using streamwise regressions on raw features. The solid black bars are the average accuracy using streamwise regressions with PCA components, raw features, and interaction terms (see Footnote 7 for the details of this kind of feature stream). The transparent bars are the average accuracies using stepwise regressions on raw features. Section 6.2 gives the results of paired two-sample t-tests.

$H_1, H_2, \ldots, H_m$. If we test the first using level α_1, the second using level α_2 and so forth with $\sum_j \alpha_j = 0.05$, then we have only a 5% chance of falsely rejecting one of the m hypotheses. If we associate each hypothesis with the claim that a feature improves the predictive power of a regression, then we are back in the situation of a Bonferroni rule for feature selection. Bonferroni methods and RIC simply fix $\alpha_j = \alpha/m$ for each test.

Alternative multiple comparison procedures control a different property. Rather than controlling the cumulative α (also known as the family wide error rate), they control the so-called false discovery rate (Benjamini and Hochberg, 1995). Control of the FDR at 5% implies that at most 5% of the rejected hypotheses are false positives. In feature selection, this implies that of the included features, at most 5% degrade the accuracy of the model. The Benjamini-Hochberg method for controlling the FDR suggests the α-investing rule used in streamwise regression, which keeps the FDR below α: order the p-values of the independents tests of $H_1, H_2, \ldots, H_m$ so that $p_1 \leq p_2 \leq \cdots p_m$. Now find the largest p-value for which $p_k \leq \alpha/(m-k)$ and reject all H_i for $i \leq k$. Thus, if the smallest p-value $p_1 \leq \alpha/m$, it is rejected. Rather than compare the second largest p-value to the RIC/Bonferroni threshold α/m, reject H_2 if $p_2 \leq 2\alpha/m$. Our proposed α-investing rule adapts this approach to evaluate an infinite sequence of features. There have been many papers that looked at procedures of this sort for use in feature selection from an FDR perspective (Abramovich et al., 2000), an empirical Bayesian perspective (George and Foster, 2000; Johnstone and Silverman, 2004), an information theoretical perspective (Foster and Stine, 2004a), or simply a data mining perspective (Foster and Stine, 2004b). But all of these require knowing the entire list of possible features ahead of time. Further, most of them assume that the features are orthogonal and hence tacitly assume that $m < n$. Obviously, the Benjamini-Hochberg method fails as m gets large; it is a batch-oriented procedure.

The α-investing rule of streamwise regression controls a similar characteristic. Framed as a multiple comparison procedure, the α-investing rule implies that, with high probability, no more than α times the number of rejected tests are false positives. That is, the procedure controls a difference rather than a rate. As a streamwise feature selector, if one has added, say, 20 features to the model, then with high probability (tending to 1 as the number of accepted features grows) no more than 5% (that is, one feature in the case of 20 features) are false positives.

8. Summary

A variety of machine learning algorithms have been developed over the years for online learning where *observations* are sequentially added. Algorithms such as the streamwise regression presented in this paper, which are online in the *features* being used are much less common. For some problems, all features are known in advance, and a large fraction of them are predictive. In such cases, regularization or smoothing methods work well and streamwise feature selection does not make sense. For other problems, selecting a small number of features gives a much stronger model than trying to smooth across all potential features. (See JMLR (2003) and Guyon (2003) for a range of feature selection problems and approaches.) For example, in predicting what journal an article will be published in, we find that roughly $10-20$ of the 80,000 features we examine are selected (Popescul and Ungar, 2003). For the problems in citation prediction and bankruptcy prediction that we have looked at, generating potential features (for example, by querying a database or by computing transformations or combinations of the raw features) takes far more time than the streamwise feature selection. Thus, the flexibility that streamwise regression provides to dynamically decide which features to generate and add to the feature stream provides potentially large savings in computation.

Empirical tests show that for the smaller UCI data sets where stepwise regression can be done, streamwise regression gives comparable results to stepwise regression or techniques such as decision trees, neural networks, or SVMs. However, unlike stepwise regression, streamwise regression scales well to large feature sets, and unlike the AIC, BIC and RIC penalties or simpler variable screening methods which use univariate tests, streamwise regression with information-investing or α-investing works well for all values of number of observations and number of potential features. Key to this guarantee is controlling the FDR by adjusting the threshold on the information gain or p-value necessary for adding a feature to the model. Fortunately, given any software which incrementally considers features for addition and calculates their p-value or entropy reduction, streamwise regression using information-investing or α-investing is extremely easy to implement. For linear and logistic regression, we have found that streamwise regression can easily handle millions of features.

The results presented here show that streamwise feature selection is highly competitive even when there is no prior knowledge about the structure of the feature space. Our expectation is that in real problems where we do know more about the different kinds of features that can be generated, streamwise regression will provide even greater benefit.

Acknowledgments

We thank Andrew Schein for his help in this work and Malik Yousef for supplying the data sets *ha* and *hung*.

References

F. Abramovich, Y. Benjamini, D. Donoho, and I. Johnstone. Adapting to unknown sparsity by controlling the false discovery rate. Technical Report 2000–19, Dept. of Statistics, Stanford University, Stanford, CA, 2000.

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csàki, editors, *2nd International Symposium on Information Theory*, pages 261–281, Budapest, 1973. Akad. Kiàdo.

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

P. Bickel and K. Doksum. *Mathematical Statistics*. Prentice Hall, 2001.

A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81: 425–455, 1994.

S. Dzeroski and N. Lavrac. *Relational Data Mining*. Springer-Verlag, 2001. ISBN 3540422897.

S. Dzeroski, L. D. Raedt, and S. Wrobel. Multi-relational data mining workshop. In *KDD-2003*, 2003.

F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.

D. P. Foster and E. I. George. The risk inflation criterion for multiple regression. *Annals of Statistics*, 22:1947–1975, 1994.

D. P. Foster and R. A. Stine. Local asymptotic coding. *IEEE Trans. on Info. Theory*, 45:1289–1293, 1999.

D. P. Foster and R. A. Stine. Adaptive variable selection competes with Bayes experts. Submitted for publication, 2004a.

D. P. Foster and R. A. Stine. Variable selection in data mining: Building a predictive model for bankruptcy. *Journal of the American Statistical Association (JASA)*, 2004b. 303-313.

E. I. George. The variable selection problem. *Journal of the Amer. Statist. Assoc.*, 95:1304–1308, 2000.

E. I. George and D. P. Foster. Calibration and empirical bayes variable selection. *Biometrika*, 87: 731–747, 2000.

R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin based feature selection - theory and algorithms. In *Proc. 21'st ICML*, 2004.

I. Guyon. Nips 2003 workshop on feature extraction and feature selection. 2003. URL `http://clopinet.com/isabelle/Projects/NIPS2003`.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature Extraction, Foundations and Applications*. Springer, 2006.

D. Jensen and L. Getoor. *IJCAI Workshop on Learning Statistical Models from Relational Data*. 2003.

JMLR. Special issue on variable selection. In *Journal of Machine Learning Research (JMLR)*, 2003. URL `http://jmlr.csail.mit.edu/`.

I. M. Johnstone and B. W. Silverman. Needles and straw in haystacks: Empirical bayes estimates of possibly sparse sequences. *Annals of Statistics*, 32:1594–1649, 2004.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.

R. J. Larsen and M. L. Marx. *An Introduction to Mathematical Statistics and Its Applications*. Prentice Hall, 2001.

J. Li and H. Liu. Bio-medical data analysis. 2002. URL `http://sdmc.lit.org.sg/GEDatasets`.

R. M. Neal. *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. Springer-Verlag, 1996.

R. M. Neal. Defining priors for distributions using dirichlet diffusion trees. Technical Report 0104, Dept. of Statistics, University of Toronto, 2001.

R. M. Neal and J. Zhang. Classification for high dimensional problems using bayesian neural networks and dirichlet diffusion trees. In *NIPS 2003 workshop on feature extraction and feature selection*, 2003. URL `http://www.cs.toronto.edu/ radford/slides.html`.

NIPS'03. Challenge results. 2003. URL `http://www.nipsfsc.ecs.soton.ac.uk/results`.

A. Popescul and L. H. Ungar. Structural logistic regression for link analysis. In *KDD Workshop on Multi-Relational Data Mining*, 2003.

A. Popescul and L. H. Ungar. Cluster-based concept invention for statistical relational learning. In *Proc. Conference Knowledge Discovery and Data Mining (KDD-2004)*, 2004.

J. Rissanen. Hypothesis selection and testing by the mdl principle. *The Computer Journal*, 42: 260–269, 1999.

G. Schwartz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

R. A. Stine. Model selection using information theory and the mdl principle. *Sociological Methods Research*, 33:230–260, 2004.

L. H. Ungar, J. Zhou, D. P. Foster, and R. A. Stine. Streaming feature selection using iic. In *AI&STAT'05*, 2005.

J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar. Streaming feature selection using alpha-investing. In *ACM SIGKDD'05*, 2005.

Journal of Machine Learning Research 7 (2006) 1887–1907 Submitted 10/05; Published 9/06

Linear Programming Relaxations and Belief Propagation – An Empirical Study

Chen Yanover CHENY@CS.HUJI.AC.IL
Talya Meltzer TALYAM@CS.HUJI.AC.IL
Yair Weiss YWEISS@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

The problem of finding the most probable (MAP) configuration in graphical models comes up in a wide range of applications. In a general graphical model this problem is NP hard, but various approximate algorithms have been developed. Linear programming (LP) relaxations are a standard method in computer science for approximating combinatorial problems and have been used for finding the most probable assignment in small graphical models. However, applying this powerful method to real-world problems is extremely challenging due to the large numbers of variables and constraints in the linear program. Tree-Reweighted Belief Propagation is a promising recent algorithm for solving LP relaxations, but little is known about its running time on large problems.

In this paper we compare tree-reweighted belief propagation (TRBP) and powerful general-purpose LP solvers (CPLEX) on relaxations of real-world graphical models from the fields of computer vision and computational biology. We find that TRBP almost always finds the solution significantly faster than all the solvers in CPLEX and more importantly, TRBP can be applied to large scale problems for which the solvers in CPLEX cannot be applied. Using TRBP we can find the MAP configurations in a matter of minutes for a large range of real world problems.

1. Introduction

The task of finding the most probable assignment (or MAP) in a graphical model comes up in a wide range of applications including image understanding (Tappen and Freeman, 2003), error correcting codes (Feldman et al., 2003) and protein folding (Yanover and Weiss, 2002). For an arbitrary graph, this problem is known to be NP hard (Shimony, 1994) and various approximation algorithms have been proposed [see. e.g (Marinescu et al., 2003) for a recent review].

Linear Programming (LP) Relaxations are a standard method for approximating combinatorial optimization problems in computer science (Bertismas and Ttsitskikilis, 1997). They have been used for approximating the MAP problem in a general graphical model by Santos (1991). More recently, LP relaxations have been used for error-correcting codes (Feldman et al., 2003), and for protein folding (Kingsford et al., 2005). LP relaxations have an advantage over other approximate inference schemes in that they come with an optimality guarantee – when the solution to the linear program is integer, then the LP solution is guaranteed to give the global optimum of the posterior probability.

The research described in this paper grew out of our experience in using LP relaxations for problems in computer vision, computational biology and statistical physics. In all these fields, the number of variables in a realistic problem may be on the order of 10^6 or more. We found that using powerful, off-the-shelf LP solvers, these problems cannot be solved using standard desktop hardware. However, linear programs that arise out of LP relaxations for graphical models have a common structure and are a small subset of all possible linear programs. The challenge is to find an LP solver that takes advantage of this special structure.

Tree-reweighted belief propagation (TRBP) is a variant of belief propagation (BP) suggested by Wainwright and colleagues (Wainwright et al., 2002), that has been shown to find the same solution as LP relaxations. Each iteration of TRBP is similar in time and space complexity to that of ordinary BP and hence it can be straightforwardly applied to very large graphical models. However, little is known regarding the convergence properties of TRBP nor about the actual number of iterations needed to solve large problems.

In this paper we compare tree-reweighted BP and powerful commercial LP solvers (CPLEX) on relaxations of real-world graphical models from the fields of computer vision and computational biology. We find that TRBP almost always finds the solution significantly faster than all the solvers in CPLEX and more importantly, TRBP can be applied to large scale problems for which the solvers in CPLEX cannot be applied. Using TRBP we can find the MAP configurations in a matter of minutes for a large range of real world problems.

2. MAP, Integer Programming and Linear Programming

We briefly review the formalism of graphical models. We use x to denote a vector of hidden variables and y to denote the observation vector. We assume the conditional distribution $\Pr(x|y)$ is Markovian with respect to a graph G – that is, it factorizes into a product of potential functions defined on the cliques of the graph G. In this paper, we focus on pairwise Markov Random Fields and assume that

$$\begin{aligned}\Pr(x|y) &= \frac{1}{Z}\prod_{<ij>}\Psi_{ij}(x_i,x_j)\prod_i\Psi_i(x_i)\\ &= \frac{1}{Z}e^{-\sum_{<ij>}E_{ij}(x_i,x_j)-\sum_i E_i(x_i)},\end{aligned}$$

where $<ij>$ refers to all pairs of nodes that are connected in the graph G and we define $E_{ij}(x_i,x_j)$, $E_i(x_i)$ as the negative logarithm of the potential $\Psi_{ij}(x_i,x_j),\Psi_i(x_i)$.

The MAP assignment is the vector x^* which maximizes the posterior probability:

$$\begin{aligned}x^* &= \arg\max_x \prod_{<ij>}\Psi_{ij}(x_i,x_j)\prod_i\Psi_i(x_i)\\ &= \arg\min_x \sum_{<ij>}E_{ij}(x_i,x_j)+\sum_i E_i(x_i).\end{aligned}$$

To define the LP relaxation, we first reformulate the MAP problem as one of integer programming. We introduce indicator variables $q_i(x_i)$ for each individual variable and additional indicator variables $q_{ij}(x_i,x_j)$ for all connected pairs of nodes in the graph. Using these indicator variables we define the integer program:

minimize

$$J(\{q\}) = \sum_{<ij>} \sum_{x_i,x_j} q_{ij}(x_i,x_j) E_{ij}(x_i,x_j) + \sum_i \sum_{x_i} q_i(x_i) E_i(x_i)$$

subject to

$$\begin{aligned} q_{ij}(x_i,x_j) &\in \{0,1\}, \\ \sum_{x_i,x_j} q_{ij}(x_i,x_j) &= 1, \\ \sum_{x_i} q_{ij}(x_i,x_j) &= q_j(x_j), \end{aligned}$$

where the last equation enforces the consistency of the pairwise indicator variables with the singleton indicator variable.

This integer program is completely equivalent to the original MAP problem, and is hence computationally intractable. We can obtain the linear programming relaxation by allowing the indicator variables to take on non-integer values. This leads to the following problem:

The LP relaxation of MAP:

minimize

$$J(\{q\}) = \sum_{<ij>} \sum_{x_i,x_j} q_{ij}(x_i,x_j) E_{ij}(x_i,x_j) + \sum_i \sum_{x_i} q_i(x_i) E_i(x_i)$$

subject to

$$\begin{aligned} q_{ij}(x_i,x_j) &\in [0,1], && (1) \\ \sum_{x_i,x_j} q_{ij}(x_i,x_j) &= 1, && (2) \\ \sum_{x_i} q_{ij}(x_i,x_j) &= q_j(x_j). && (3) \end{aligned}$$

This is now a linear program – the cost and the constraints are linear. It can therefore be solved in polynomial time and we have the following guarantee:

Lemma If the solutions $\{q_{ij}(x_i,x_j), q_i(x_i)\}$ to the MAP LP relaxation are all *integer*, that is $q_{ij}(x_i,x_j), q_i(x_i) \in \{0,1\}$, then $x_i^* = \arg\max_{x_i} q_i(x_i)$ is the MAP assignment.+

2.1 The Need for Special Purpose LP Solvers

Given the tremendous amount of research devoted to LP solvers, it may seem that the best way to solve LP relaxations for graphical models, would be to simply use an industrial-strength, general-purpose LP solver. However, by relaxing the MAP into a linear program we increase the size of the problem tremendously. Formally, denote by k_i the number of possible states of node i. The number of variables and constraints in the LP relaxation is given by

$$\begin{aligned} N_{variables} &= \sum_i k_i + \sum_{<i,j>} k_i k_j, \\ N_{constraints} &= \sum_{<i,j>} (k_i + k_j + 1). \end{aligned}$$

The additional $\sum_{<i,j>} 2k_i k_j$ bound constraints, derived from equation (1), are usually not considered part of the constraint matrix.

As an example, consider an image processing problem (an example of such a problem, the stereo problem, is discussed in Section 4.1). If the image is a modest 200×200 pixels and each pixel can take on 30 discrete values, then the LP relaxation will have over 72 million variables and four million constraints. Obviously, we need a solver that can somehow take advantage of the problem structure in order to deal with such a large-scale problem.

3. Solving Linear Programs Using Tree-Reweighted Belief Propagation

Tree-reweighted belief propagation (TRBP) is a variant of belief propagation introduced by Wainwright and colleagues (Wainwright et al., 2002). We start by briefly reviewing ordinary max-product belief propagation [see e.g. (Yedidia et al., 2001; Pearl, 1988)]. The algorithm receives as input a graph G and the potentials Ψ_{ij}, Ψ_i. At each iteration, a node i sends a message $m_{ij}(x_j)$ to its neighbor in the graph j. The messages are updated as follows:

$$m_{ij}(x_j) \leftarrow \alpha_{ij} \max_{x_i} \Psi_{ij}(x_i, x_j) \Psi_i(x_i) \prod_{k \in N_i \setminus j} m_{ki}(x_i) \tag{4}$$

where $N_i \setminus j$ refers to all neighbors of node i except j. The constant α_{ij} is a normalization constant typically chosen so that the messages sum to one (the normalization has no influence on the final beliefs). After the messages have converged, each node can form an estimate of its local "belief" defined as

$$b_i(x_i) \propto \Psi_i(x_i) \prod_{j \in N_i} m_{ji}(x_i).$$

It is easy to show that when the graph is singly-connected, choosing an assignment that maximizes the local belief will give the MAP estimate (Pearl, 1988). In fact, when the graph is a chain, equation 4 is simply a distributed computation of dynamic programming. When the graph has cycles, ordinary BP is no longer guaranteed to converge, nor is there a guarantee that it can be used to find the MAP.

In tree-reweighted BP (TRBP), the algorithm receives as input an additional set of *edge appearance probabilities*, ρ_{ij}. These edge appearance probabilities are essentially free parameters of the algorithm and are derived from a distribution over spanning trees of the graph G. They represent the probability of an edge (ij) appearing in a spanning tree under the chosen distribution. As in standard belief propagation, at each iteration a node i sends a message $m_{ij}(x_j)$ to its neighbor in the graph j. The messages are updated as follows:

$$m_{ij}(x_j) \leftarrow \alpha_{ij} \max_{x_i} \Psi_{ij}^{1/\rho_{ij}}(x_i, x_j) \Psi_i(x_i) \frac{\prod_{k \in N_i \setminus j} m_{ki}^{\rho_{ki}}(x_i)}{m_{ji}^{1-\rho_{ji}}(x_i)}. \tag{5}$$

Note that for $\rho_{ij} = 1$ the algorithm reduces to standard belief propagation.

After one has found a fixed-point of these message update equations, the singleton and pairwise beliefs are defined as

$$b_i(x_i) \quad \propto \quad \Psi_i(x_i) \prod_{j \in N_i} m_{ji}^{\rho_{ji}}(x_i),$$

$$b_{ij}(x_i,x_j) \propto \Psi_i(x_i)\Psi_j(x_j)\Psi_{ij}^{1/\rho_{ij}}(x_i,x_j)\cdot\frac{\prod_{k\in N_i\setminus j} m_{ki}^{\rho_{ki}}(x_i)}{m_{ji}^{1-\rho_{ji}}(x_i)}\frac{\prod_{k\in N_j\setminus i} m_{kj}^{\rho_{kj}}(x_j)}{m_{ij}^{1-\rho_{ij}}(x_j)}.$$

The relationship between TRBP and the solution to the LP relaxation has been studied by (Wainwright et al., 2002; Kolmogorov, 2005; Kolmogorov and Wainwright, 2005) and is a subject of ongoing research. We briefly summarize some of the relationships.

Given a set of TRBP beliefs, we define the *sharpened beliefs* as follows:

$$\begin{aligned} q_i(x_i) &\propto \delta(b_i(x_i) - \max_{x_i} b_i(x_i)), \\ q_{ij}(x_i,x_j) &\propto \delta(b_{ij}(x_i,x_j) - \max_{x_i,x_j} b_{ij}(x_i,x_j)), \end{aligned}$$

where $\delta(\cdot)$ is the Dirac delta function ($\delta(0)=1$ and $\delta(x)=0$ for all $x\neq 0$). That is, we get a uniform distribution over all the maximizing values and assign 0 probability to all non-maximizing values. To illustrate this definition, a belief vector $(0.6,0.4)$ would be sharpened to $(1,0)$ and a belief vector $(0.4,0.4,0.2)$ would be sharpened to $(0.5,0.5,0)$.

Using these sharpened beliefs, the following properties hold:

- At any iteration, and in particular in fixed-point, the TRBP beliefs provide a lower bound on the solution of the LP (see appendix A).
- If there exists a unique maximizing value for the pairwise beliefs $b_{ij}(x_i,x_j)$ then the sharpened beliefs solve the LP. In that case $x_i^* = \arg\max_{x_i} b_i(x_i)$ is the MAP.
- Suppose the TRBP beliefs have ties. If there exists x^* such that x_i^*, x_j^* maximize $b_{ij}(x_i,x_j)$ and x_i^* maximize $b_i(x_i)$, then x^* is the MAP. In that case, define q_{ij}^*, q_i^* as indicator variables for x^*, then q_{ij}^*, q_i^* are a solution for the LP.
- If the sharpened beliefs at a fixed-point of TRBP satisfy the LP constraints (equations 1-3), then the sharpened beliefs are a solution to the LP relaxation.
- Suppose the TRBP beliefs have ties. If there exists $\tilde{b}_i, \tilde{b}_{ij}$ that satisfy the LP constraints and for all x_i, x_j, $\tilde{b}_i(x_i)=0$ if $q_i(x_i)=0$ and $\tilde{b}_{ij}(x_i,x_j)=0$ if $q_{ij}(x_i,x_j)=0$, then $\tilde{b}_i, \tilde{b}_{ij}$ are a solution to the LP relaxation.

The lower-bound property is based on Lagrangian duality and is proven in (Wainwright et al., 2002; Kolmogorov, 2005). The subsequent properties follow from the lower-bound property.

Based on these properties, we use the following algorithm to extract the LP solution and the MAP from TRBP beliefs:

1. Run TRBP until convergence and identify the tied nodes x_T.
2. For all non-tied nodes, x_{NT}, set $x_i^* = \arg\max_{x_i} b_i(x_i)$.
3. Construct a new graphical model that includes only the tied nodes and the possible states are only those that maximize the beliefs. The pairwise potentials are 1 if the pair maximizes the pairwise belief and ε otherwise. Use the junction tree algorithm (Cowell, 1998) to find x_T^*, the MAP in this new graphical model. If x_T^* has energy equal to zero then $x^* = (x_{NT}^*, x_T^*)$ is the MAP and q_{ij}^*, q_i^*, defined as indicator variables for x^*, are a solution to the LP.

3.1 TRBP Complexity

Little is known about the number of iterations needed for TRBP to converge, but what is relatively straightforward to calculate is the cost per iteration. In every TRBP iteration, each node calculates and sends messages to all its neighbors. Thus the number of message updates is two times the number of edges in the graph. Each message update equation involves point-wise multiplication of vectors of length k_i followed by a matrix by vector max-multiplication,[1] where the size of the matrix is $k_j \times k_i$. By working in the log domain, the point-wise multiplications can be transformed into summations, and the raising of the messages to the powers $\rho_{ij}, 1-\rho_{ij}$ are transformed into multiplications. In summary, the dominant part of the computation is equivalent to $2|\mathcal{E}|$ multiplications of a $k_i \times k_j$ matrix times a $k_j \times 1$ vector (where $|\mathcal{E}|$ is the number of edges in the graph).

The amount of memory needed depends on the particular implementation. In the simplest implementation, we would need to store the potentials Ψ_{ij}, Ψ_i in memory. The size of the potentials is exactly the number of variables in the linear program: $\sum_i k_i + \sum_{<i,j>} k_i k_j$. Additionally, storing the messages in memory requires $\sum_{<i,j>} k_i + k_j$ (which is typically small relative to the memory required for the pairwise potentials). In many problems, however, the pairwise potentials can be stored more compactly. For example, in the Potts model, the pairwise potential $\Psi_{ij}(x_i, x_j)$ is a $k \times k$ table that has only two unique values: 1 on the diagonal and $e^{-\lambda_{ij}}$ for the off-diagonal terms. Additional implementation techniques for reducing the memory requirements of BP appear in (Felzenszwalb and Huttenlocher, 2004; Kolmogorov, 2005).

4. The Benchmark Problems

We constructed benchmark problems from three domains: stereo vision, side-chain prediction and protein design. We give here a short overview of how we constructed the graphical models in all three cases. The exact graphical models can be downloaded from the JMLR web site.

4.1 Stereo Vision

The stereo problem is illustrated in Figure 1. Given a stereo pair of images, $Left(u,v)$ and $Right(u,v)$, the problem is to find the disparity of each pixel in a reference image. This disparity can be straightforwardly translated into depth from the camera.

The main cue for choosing disparities are the similarities of local image information in the left and right image. That is, we search for a disparity so that

$$Left(u,v) \approx Right(u + disp(u,v), v),$$

or equivalently,

$$disp^*(u,v) = \arg\min_{disp(u,v)} \left[Left(u,v) - Right(u + disp(u,v), v)\right]^2.$$

This criterion by itself is typically not enough. Locally, there can be many disparities for a pixel that are almost equally good. The best algorithms currently known for the stereo problem are those that minimize a global energy function (Scharstein and Szeliski, 2002):

$$disp^* = \arg\min_{disp} \sum_{u,v} dissim\left[Left(u,v), Right(u + disp(u,v), v)\right] + \lambda \cdot smoothness(disp),$$

1. Max-multiplication of a matrix by a vector is equivalent to ordinary matrix multiplication but all summations are replaced by maximizations.

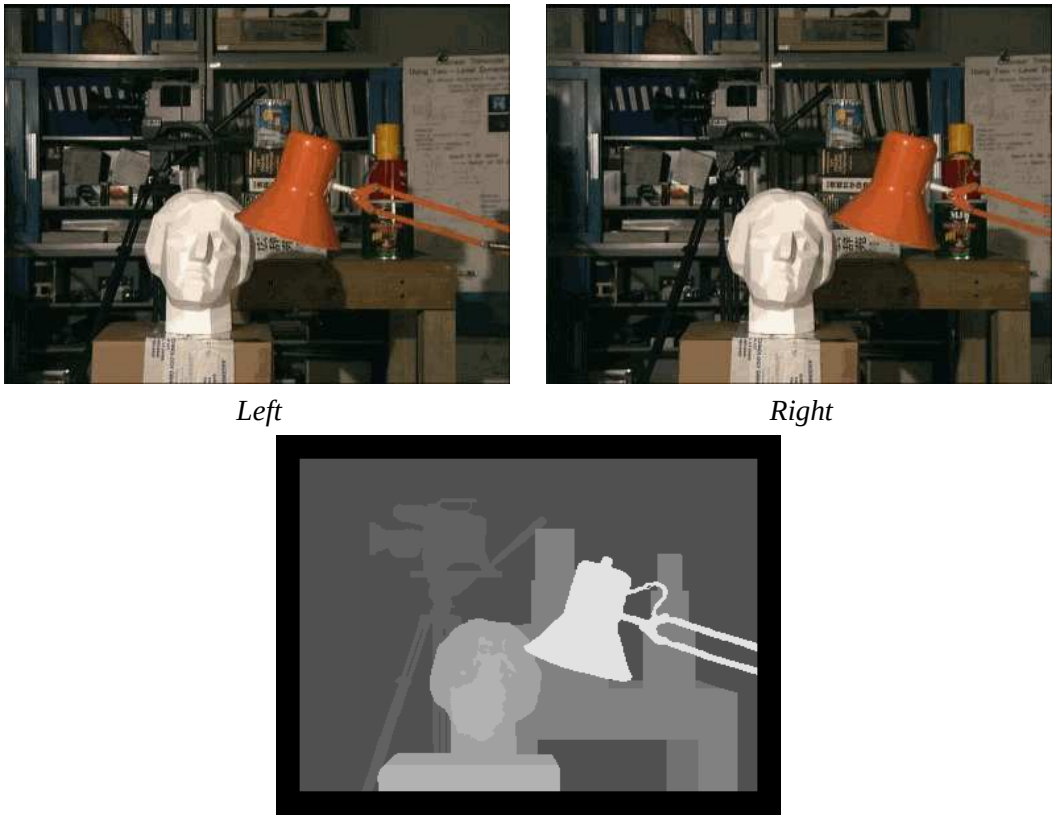


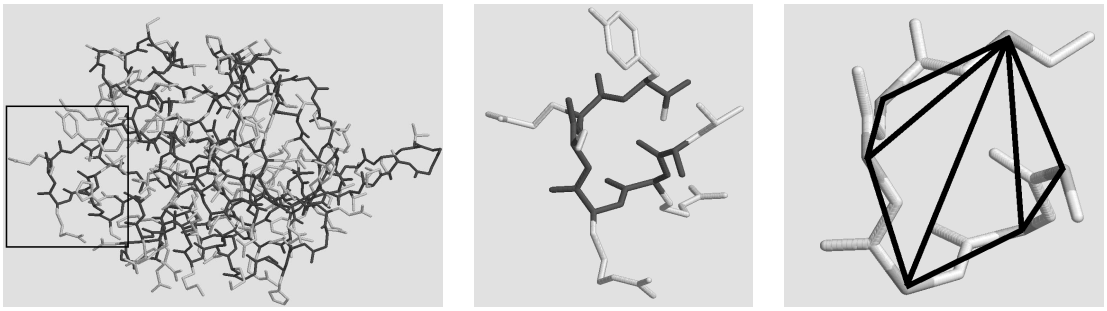
Figure 1: An illustration of the stereo problem. Given two images taken from slightly different viewpoints ($Left, Right$) we search for the disparity of each pixel. The best results for this problem use energy minimization formulations which are equivalent to solving the MAP for a grid graphical model.

where $smoothness(disp)$ is a cost that penalizes disparity fields where neighboring pixels have different disparities and $dissim[Left(u,v), Right(u',v')]$ measures the dissimilarity of the left and right image at corresponding locations.

We associate each disparity $disp(u,v)$ with an assignment of a node x_i in a two dimensional grid graph. If we define x to be the disparity field, and $P(x|y) \propto \exp(-E(x))$ where $E(x)$ is the energy function, minimizing the energy is equivalent to maximizing P(x). Furthermore, since $E(x)$ is a sum of singleton and pairwise terms, $P(x)$ will factorize with respect to the two-dimensional grid:

$$\begin{aligned} Pr(x|y) &\propto \prod_i \Psi_i(x_i) \prod_{<ij>} \Psi_{ij}(x_i, x_j) = \\ &= e^{-\sum_i E_i(x_i) - \sum_{<ij>} E_{ij}(x_i, x_j)}. \end{aligned}$$

The problem of finding the most probable set of disparities is NP hard. Good approximate solutions can be achieved using algorithms based on min-cut/max-flow formulations (Boykov et al.,



(a) **(b)** **(c)**

Figure 2: **(a)** Cow actin binding protein (PDB code 1pne). **(b)** A closer view of its 6 C-terminal residues. Given the protein backbone (black) and the amino acid sequence, side-chain prediction is the problem of predicting the native side-chain conformation (gray). **(c)** Problem representation as a graphical model for those C-terminal residues shown in (b) (nodes located at C^{α} atom positions, edges drawn in black).

1999; Kolmogorov and Zabih, 2004) and Belief Propagation (Felzenszwalb and Huttenlocher, 2004; Tappen and Freeman, 2003; Sun et al., 2002).

In this work we use the same energy function used by Tappen and Freeman (2003). The local cost is based on the Birchfield-Tomasi matching cost (Birchfield and Tomasi, 1998) and the pairwise energy penalizes for neighboring pixels having different disparities. The amount of penalty depends only on the intensity difference between the two pixels and therefore, for each pair of neighboring pixels, the penalty for violating the smoothness constraint is constant. Thus the MRF is equivalent to a Potts model. Specifically, the pairwise energy penalty is defined using 3 parameters – s, P and T – and set to $P \cdot s$ when the intensity difference between the two pixels is smaller than a threshold T, and s otherwise.

We used four images from the standard Middlebury stereo benchmark set (Scharstein and Szeliski, 2003). By varying the parameters of the energy function, as in (Tappen and Freeman, 2003), we obtained 22 different graphical models. The parameters s, P, T are constant over the whole image.

4.2 Side-Chain Prediction

Proteins are chains of simpler molecules called *amino acids*. All amino acids have a common structure – a central carbon atom (C^{α}) to which a hydrogen atom, an amino group (NH_2) and a carboxyl group ($COOH$) are bonded. In addition, each amino acid has a chemical group called the *side-chain*, bound to C^{α}. This group distinguishes one amino acid from another and gives its distinctive properties. Amino acids are joined end to end during protein synthesis by the formation of peptide bonds. An amino acid unit in a protein is called a *residue*. The formation of a succession of peptide bonds generates the *backbone* (consisting of C^{α} and its adjacent atoms, N and CO, of each reside), upon which the side-chains are hanged (Figure 2).

The *side-chain prediction* problem is defined as follows: given the 3 dimensional structure of the backbone we wish to predict the placements of the side-chains. This problem is considered of central importance in protein-folding and molecular design and has been tackled extensively using

a wide variety of methods. Typically, an energy function is defined over a discretization of the side-chain angles and search algorithms are used to find the global minimum. Even when the energy function contains only pairwise interactions, the configuration space grows exponentially and it can be shown that the prediction problem is NP-complete (Fraenkel, 1997; Pierce and Winfree, 2002).

Formally, our search space is a set of energetically preferred conformations (called *rotamers*) and we wish to minimize an energy function that is typically defined in terms of pairwise interactions among nearby residues and interactions between a residue and the backbone:

$$E(r) = \sum_{<ij>} E_{ij}(r_i, r_j) + \sum_i E_i(r_i, backbone),$$

where $r = (r_1, ..., r_N)$ denotes an assignment of rotamers for all residues.

Since we have a discrete optimization problem and the energy function is a sum of pairwise interactions, we can transform the problem into a graphical model with pairwise potentials. Each node corresponds to a residue, and the state of each node represents the configuration of the side-chain of that residue. Since $E(r)$ is a sum of singleton and pairwise terms, $P(r)$ will factorize:

$$\begin{aligned} P(r) = \frac{1}{Z} e^{-E(r)} &= \frac{1}{Z} e^{-\sum_i E_i(r_i) - \sum_{<ij>} E_{ij}(r_i, r_j)} \\ &= \frac{1}{Z} \prod_i \Psi_i(r_i) \prod_{<ij>} \Psi_{ij}(r_i, r_j) \end{aligned} \quad (6)$$

where Z is an explicit normalization factor. Equation (6) requires multiplying Ψ_{ij} for all pairs of residues i, j but in all reasonable energy functions the pairwise interactions go to zero for atoms that are sufficiently far away. Thus we only need to calculate the pairwise interactions for nearby residues. To define the topology of the undirected graph, we examine all pairs of residues i, j and check whether there exists an assignment r_i, r_j for which the energy is nonzero. If it exists, we connect nodes i and j in the graph and set the potential to be: $\Psi_{ij}(r_i, r_j) = e^{-E_{ij}(r_i, r_j)}$.

Figure 2(c) shows a subgraph of the undirected graph. The graph is relatively sparse (each node is connected to nodes that are close in 3D space) but contains many small loops. A typical protein in the data set gives rise to a model with hundreds of loops of size 3.

As a data set we used 370 X-ray crystal structures with resolution better than or equal to 2Å, R factor below 20% and mutual sequence identity less than 50%. Each protein consisted of a single chain and up to 1,000 residues. Protein structures were acquired from the Protein Data Bank site (`http://www.rcsb.org/pdb`). For each protein, we have built two representing graphical models:

1. Using the SCWRL energy function (Canutescu et al., 2003), which approximates the repulsive portion of Lennard-Jones 12-6 potential.

2. Using the more elaborate energy function used in the Rosetta program (Kuhlman and Baker, 2000) which is comprised of (Rohl et al., 2004): (1) the attractive portion of the 12-6 Lennard-Jones potential, (2) The repulsive portion of a 12-6 Lennard Jones potential. This term is dampened in order to compensate for the use of a fixed backbone and rotamer set, (3) solvation energies calculated using the model of Lazaridis and Karplus (1999), (4) an approximation to electrostatic interactions in proteins, based on PDB statistics, (5) hydrogen-bonding potential (Kortemme et al., 2003) and (6) backbone dependent internal free energies of the rotamers estimated from PDB statistics performed by Dunbrack and Kurplus (1993).

The Rosetta energy function accounts for distant interactions and therefore gives rise to denser graphical models, compared to SCWRL's. In both cases we used Dunbrack and Kurplus (1993) backbone dependent rotamer library to define up-to 81 configurations for each side-chain. The side-chain prediction data sets are publicly available and can be downloaded from `http://www.jmlr.org/papers/volume7/yanover06a/SCWRL_SCP_Dataset.tgz` and `http://www.jmlr.org/papers/volume7/yanover06a/Rosetta_SCP_Dataset.tgz`.[2]

4.3 Protein Design

The protein design problem is the inverse of the protein folding problem. Given a particular 3D shape, we wish to find a sequence of amino-acids that will be as stable as possible in that 3D shape. Typically this is done by finding a set of (1) amino-acids and (2) rotamer configurations that minimizes an approximate energy [see (Street and Mayo, 1999) for a review of computational protein design].

While the problem is quite different from side-chain prediction it can be solved using the same graph structure. The only difference is that now the nodes do not just denote rotamers but also the identity of the amino-acid at that location. Thus, the state-space here is significantly larger than in the side-chain prediction problem. We, again, used the Rosetta energy function to define the pairwise and local potentials (Kuhlman and Baker, 2000). As a data set we used 97 X-ray crystal structures, 40-180 amino acids long. For each of these proteins, we allowed all residues to assume any rotamer of any amino acid. There are, therefore, hundreds of possible states for each node. The protein design data set is available from `http://www.jmlr.org/papers/volume7/yanover06a/Rosetta_Design_Dataset.tgz`.

5. Experiments

We compared TRBP to the LP solvers available from CPLEX (CPLEX *v*9.0, `tomlab.biz`). CPLEX is widely considered to be one of the most powerful LP packages available commercially and (according to the company's website) is used in 95% of all academic papers that reference an LP solver. In addition to its widespread use, CPLEX is well suited for our empirical study because it is highly optimized for solving LP relaxations (as opposed to arbitrary LPs). Indeed as the original programmer of CPLEX notes "the solution of integer programs is the dominant application of linear programming in practice" (Bixby, 2001) and CPLEX contains a large number of optimizations that are based on exploiting the sparse structure of LPs that arise from LP relaxations (Bixby, 2001).

Specifically, we tried the following solvers from CPLEX.

1. *Primal* and *dual* simplex solvers.

2. CPLEX has a very efficient algorithm for *network* models. Network constraints have the following property: each non-zero coefficient is either a $+1$ or a -1 and column appearing in these constraints has exactly 2 nonzero entries, one with a $+1$ coefficient and one with a -1 coefficient. CPLEX can also automatically extract networks that do not adhere to the above conventions as long as they can be transformed to have those properties.

2. Note that loading a protein file in Matlab requires using the sparse_cell package available from `http://www.cs.huji.ac.il/~cheny`.

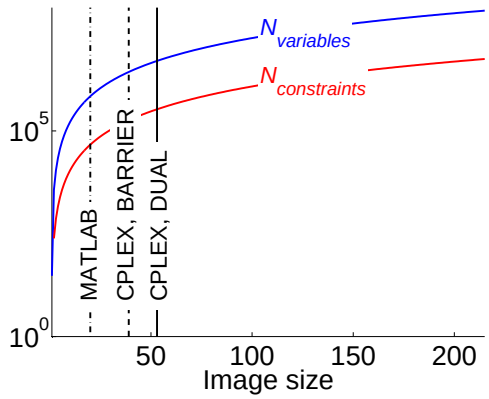


Figure 3: The number of variables and constraints in the LP relaxation of the stereo disparity problem as a function of the size of the image. The largest image that could be solved using the CPLEX solvers is approximately 50×50 while TRBP can be run on full size images.

3. The *barrier* algorithm is a primal-dual logarithmic barrier algorithm which generates a sequence of strictly positive primal and dual solutions. The barrier algorithm is highly optimized for large, sparse problems.

4. CPLEX provides a *sifting* algorithm which can be effective on problems with many more variables than equations. Sifting solves a sequence of LP subproblems where the results from one subproblem are used to select columns from the original model for inclusion in the next subproblem.

5. The *concurrent* optimizer can apply multiple algorithms to a single linear programming problem. Each algorithm operates on a different CPU.

We used the `tomlab` package that provides a Matlab interface to CPLEX 9.0. Our TRBP implementation was written in C++ and linked to Matlab as a cmex file. The TRBP implementation is completely general and receives as input a graph and the potential functions. It iterates the message updating equations (equation 5) until convergence. To improve the convergence properties "dampening" is used – we only move the new messages halfway towards the new value of the message. The messages are represented in the log domain so that multiplication is replaced with summation. Convergence is declared when the beliefs change by no more than 10^{-8} between successive iterations and the same threshold is used to determine ties. The edge appearance probabilities ρ_{ij} are automatically calculated for a given graph by greedily constructing a set of spanning trees until all edges in the graph appear in exactly one spanning tree. The junction tree algorithm needed for the post-processing of the TRBP beliefs is performed in Matlab using Kevin Murphy's BNT package (Murphy, 2001). Despite the Matlab implementation, the junction tree run-time is negligible compared to the TRBP run times (typically less than 30 seconds for the junction tree). We also compared the run-times of ordinary BP by running the same code but with $\rho_{ij} = 1$ for all edges. All algorithms were run on a dual processor Pentium 4 with $4G$ memory (but using a single processor only).

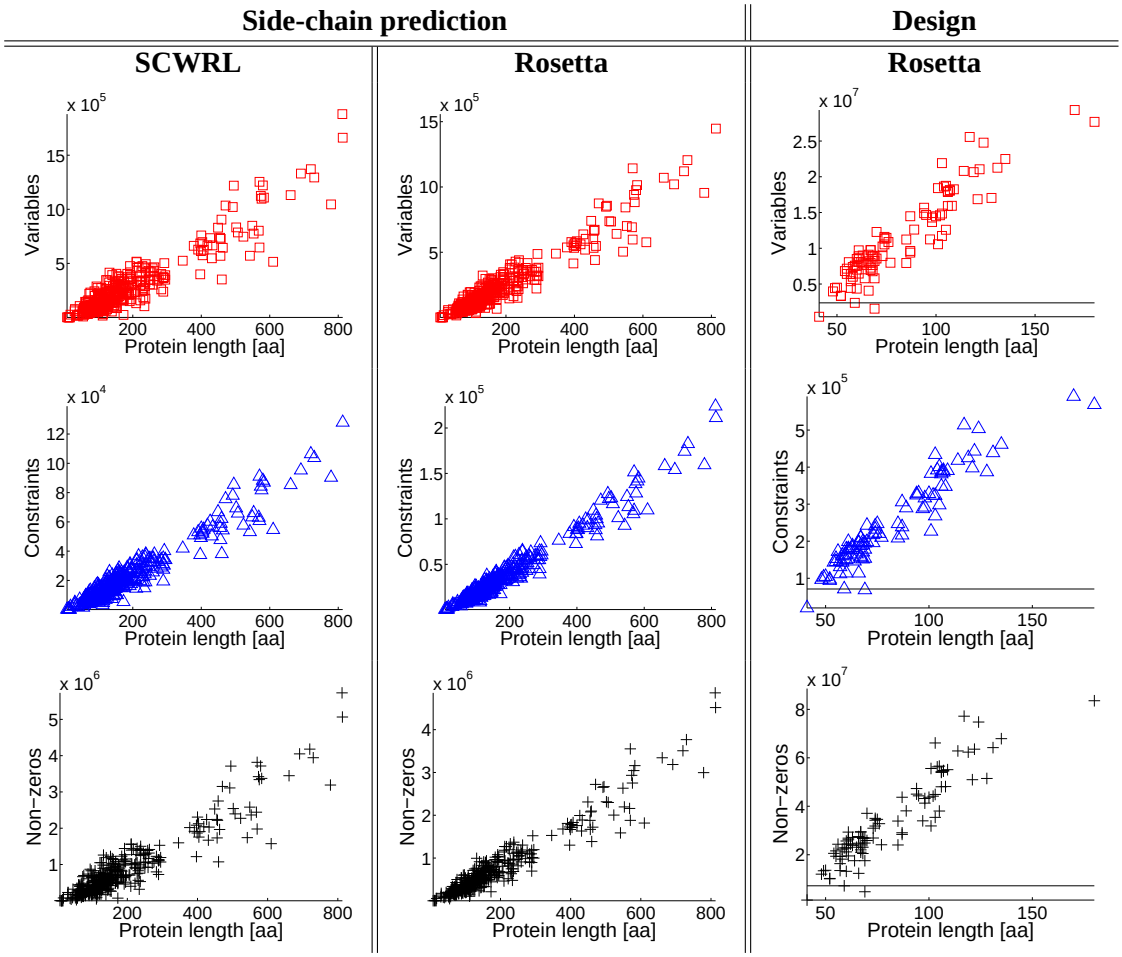


Figure 4: The number of variables, constraints and non-zero entries in the constraint matrix for our benchmark problems in side-chain prediction, using SCWRL (left) and Rosetta (middle) energy functions, and protein design (right). For the side-chain prediction problem both TRBP and the CPLEX solvers could solve the LP relaxation for all proteins in the database. For the protein design problem, on the other hand, the CPLEX solvers could only solve a small fraction of the database ($3/97$) while TRBP could solve the relaxations for all the proteins in the database (the horizontal line in the plots in the right column indicates the largest model that could be solved using CPLEX).

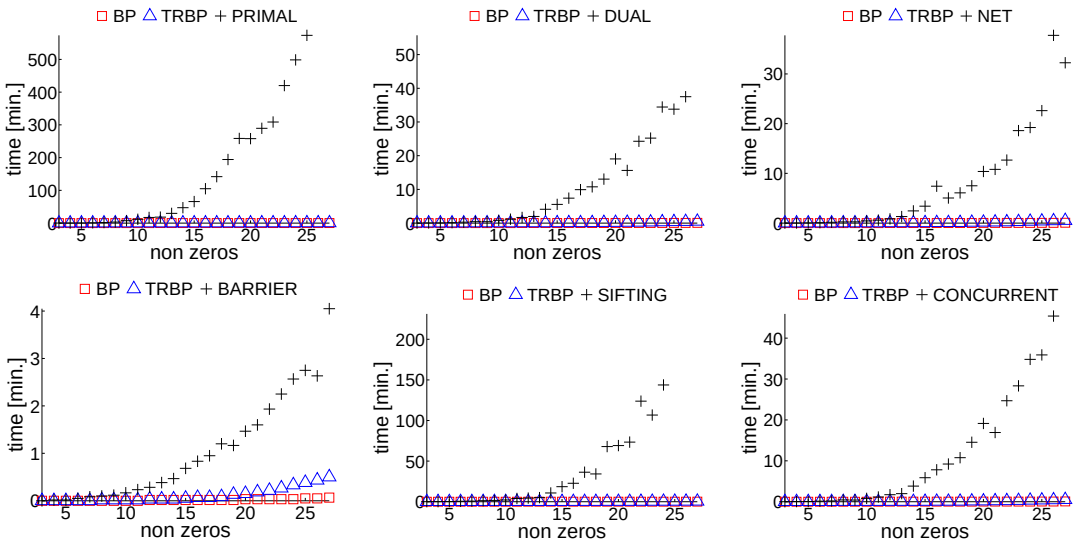


Figure 5: Comparison of run-times of the six solvers from CPLEX and TRBP on a set of subproblems constructed from the "map" image in the Middlebury stereo benchmark set. The barrier method is the fastest of the CPLEX solvers but it is still significantly slower than TRBP for relatively large problems.

The first question we asked was: what is the largest problem in each data set that can be solved within 12 hours by each of the solvers ? Figure 3 shows the results for a standard stereo benchmark image (the "map" image from the Middlebury stereo benchmark set (Scharstein and Szeliski, 2003)). We constructed smaller problems by taking subimages from the full image. Out of the CPLEX solvers, the dual simplex algorithm could solve the largest subproblem (the barrier algorithm requires more memory) but it could not solve an image larger than approximately 50×50 pixels. In contrast, TRBP can be run on the full benchmark images (approximately 250×250 pixels). Figure 4 shows the problem sizes for the side-chain prediction and the protein design problems. For the side-chain prediction problem all solvers could be applied to the full benchmark set. However for the protein design problem (in which the state space is much larger) the CPLEX solvers could solve only 2 out of the 96 problems in the database (this is indicated by the horizontal line in the plots in the right column) while TRBP could solve them all.

In the second experiment we asked: how do the run-times of the solvers compare in settings where all solvers can be applied. Figure 5 compares the run-times on the sequence of subproblems constructed from the Middlebury stereo benchmark set. As can be seen, the barrier method is the fastest of the CPLEX solvers but it is still significantly slower than TRBP on large problems.

Figure 6 compares the run times of the different solvers on the side-chain prediction graphical models. Again, the barrier method is the fastest of the CPLEX solvers (with dual simplex and network solvers providing similar performance with less memory requirements) but is significantly slower than TRBP for large problems. Figure 7 shows the run times of TRBP, BP, and the barrier CPLEX solver on the protein design problem. For the few cases for which the barrier method did not run out of memory, TRBP is significantly faster.

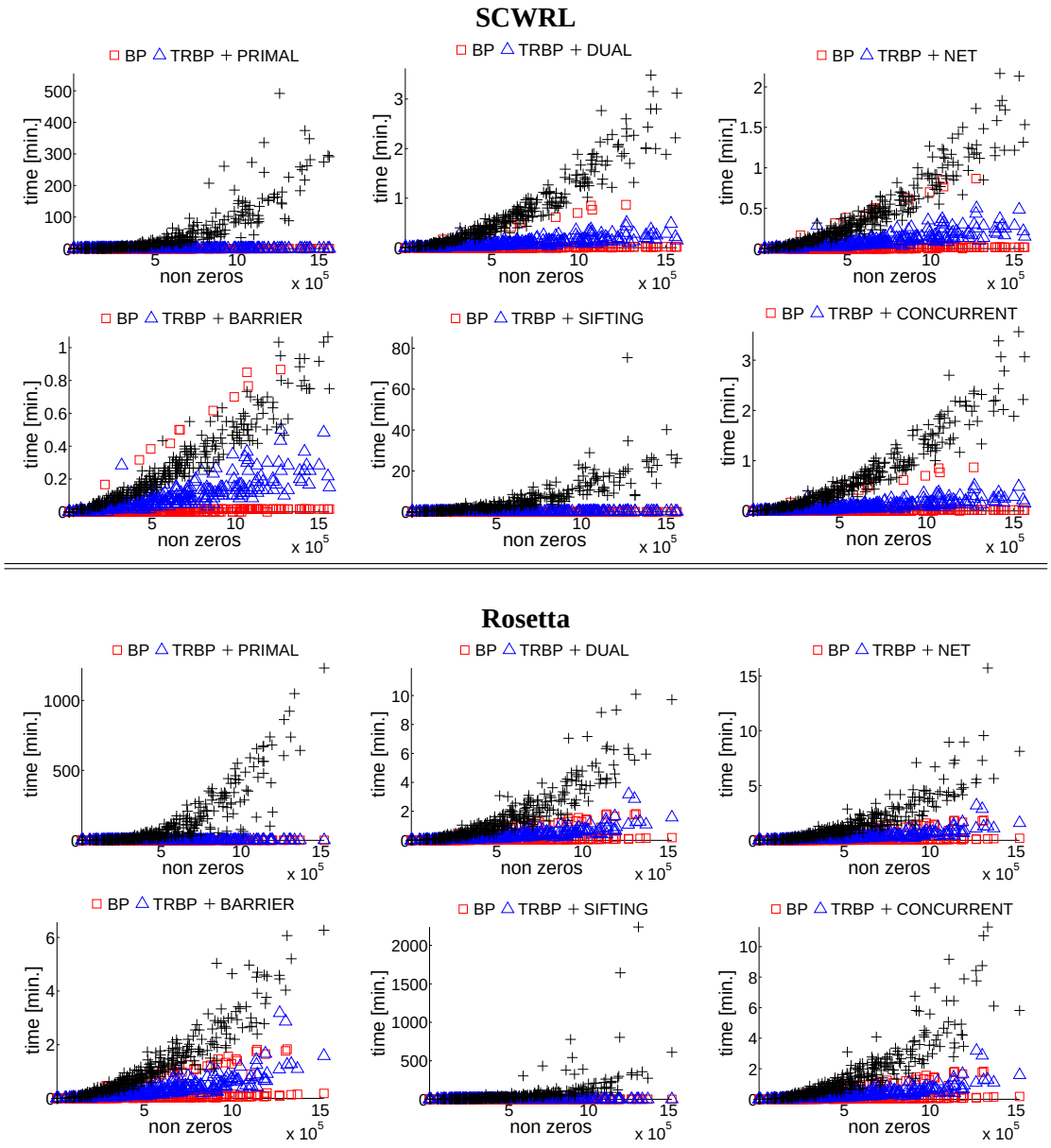


Figure 6: A comparison of the run-times of the different solvers in CPLEX and TRBP on the side-chain prediction benchmark. Again the barrier method is the fastest of the CPLEX solvers (with dual simplex and the network solver providing similar performance with less memory requirements). TRBP consistently converges faster than the barrier method and the difference becomes more significant as the problem size increases.

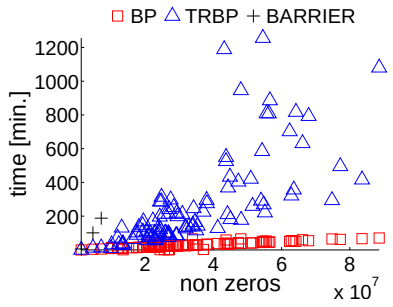


Figure 7: A comparison of the run-times of the barrier method and TRBP on the protein design problem. For the few cases in which the barrier method did not run out of memory, TRBP is significantly faster.

In the third set of experiments we asked: in what fraction of the runs can we use the results of the LP relaxation to find the MAP? We define a run of TRBP as "successful" if the TRBP beliefs allowed us to find the MAP of the graphical model (i.e. if we could find an assignment x^* that maximized the pairwise and singleton beliefs). In the stereo benchmark we could directly find the MAP in 12 out of 22 cases, but by using additional algorithms on the TRBP output we could find the MAP on 19 out of the 22 cases (Meltzer et al., 2005). Figure 8 shows the success rate for TRBP in the side-chain prediction problems. For these problems, TRBP's success rate was over 90% for proteins in our database with length less than 200 amino acids. As the proteins become larger, the problem becomes more complex and the success rate decreases. The figures also show the fraction of times in which the TRBP beliefs allowed us to solve the linear program. The protein design problem is, apparently, a more difficult problem and the success rate is therefore much lower – the MAP assignment could be found for 2 proteins only and TRBP beliefs allowed us to solve the LP relaxations for 6 proteins only. Note, however, that we could still use the TRBP beliefs to obtain a lower bound on the optimal solution.

We also assessed the success rate of the standard LP solvers, which we defined as a case when the LP solution was nonfractional. We found that *in all cases in which the LP solution was nonfractional the TRBP beliefs had a unique maximum*. Thus the success rate of the standard LP solvers was strictly less than that of TRBP (since TRBP also allows for obtaining a solution with partially tied beliefs).

6. What is TRBP's Secret?

Given the performance advantages of TRBP over the solvers in CPLEX, it is natural to ask "what is TRBP's secret?". The first thing to emphasize in this context is that *TRBP is not a general purpose LP solver*. It can only solve a tiny fraction of linear programs with a very special structure.

To see this structure, consider the general LP problem: minimize $c^T q$ subject to $Aq = b$ and $Cq < d$. If we translate LP relaxations of MAP into this form we find that the equality matrix A, the inequality matrix C, and the vectors b, d all contain only elements in $\{-1, 0, 1\}$. Tardos (1986) has shown that linear programs with integer constraint matrices can be solved with a strongly

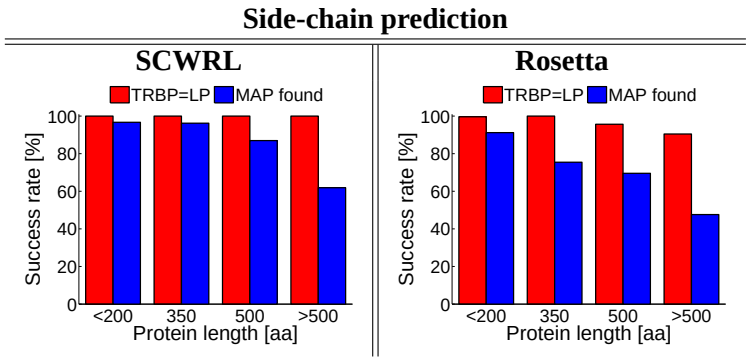


Figure 8: Success rate of TRBP on the side-chain prediction problems. A run of the algorithm was considered a success if we could use the TRBP beliefs to find the MAP of the graphical model. The figures also show the fraction of times in which the TRBP beliefs allowed us to solve the linear program.

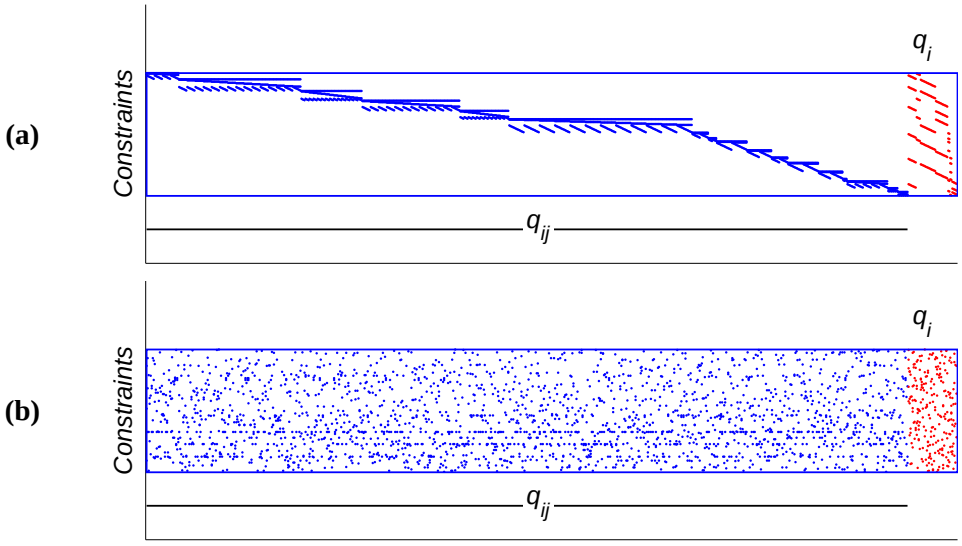


Figure 9: The sparsity pattern of a typical equality matrix A **(a)** and a random permutation of this matrix **(b)**. Blue and red dots indicate $+1$ and -1 entries respectively.

polynomial algorithm (suggesting that they are easier to solve than general purpose LPs for which no strongly polynomial algorithm is known).

The matrix A that arises in LP relaxations of MAP has additional structure, beyond the fact that its elements are in $\{-1,0,1\}$. Figure 9(a) shows the sparsity pattern of the matrix A for a small graphical model. The matrix is sparse and has a special block form. The special structure arises from the fact that we only have a consistency constraint for a pairwise indicator q_{ij} to the two singleton indicators, that involve nodes i and j. There is no interaction between the pairwise indicators q_{ij} and any other pairwise indicator q_{kl} nor is there an interaction with any other singleton

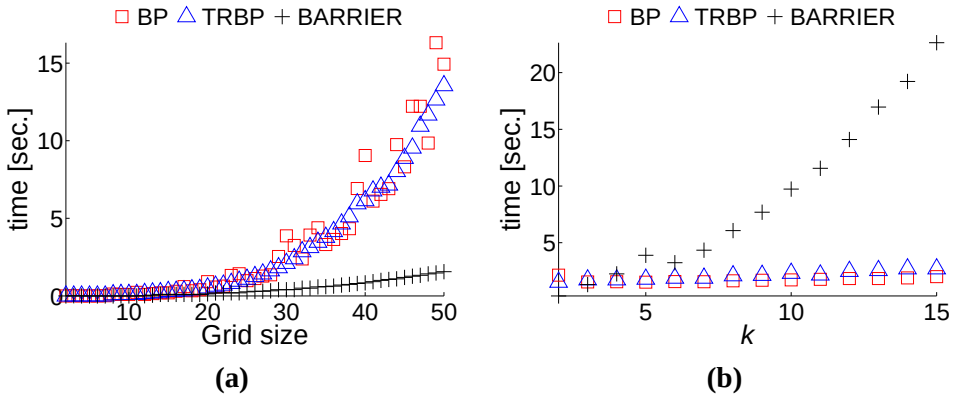


Figure 10: A comparison of the run-times of the barrier method and TRBP on **(a)** binary spin glass models (Potts models with positive and negative λ_{ij}), as a function of grid size and on **(b)** a 25×25 grid Potts model, as a function of the number of possible states, k. Each datapoint represents the average over 10 random samplings of λ_{ij}.

indicator q_k in the graph. For comparison, Figure 9(b) shows a random permutation of the matrix – this has the same sparsity pattern but without the block structure.

Note that TRBP does not even represent the matrix A explicitly. Instead, TRBP explicitly represents the graph G which implicitly defines the matrix A. In contrast, the CPLEX solvers explicitly represent A and this matrix implicitly represents the graph G (by finding the correct permutation of A that reveals the block structure, it is possible to reconstruct the graph G). We believe that this difference in representation may be responsible for TRBP's superior performance.

To investigate the conjecture that TRBP's advantage is related to an explicit representation of the graph structure, we compared the run-times of the barrier LP solver and TRBP on spin glass models (Potts models with positive and negative λ_{ij}) with different numbers of possible states per node, k. Note that the size of the block in the constraint matrix in Figure 9 is directly related to k – for binary nodes the blocks are of size 5×4 and we conjectured that when the blocks are small, TRBP's advantage will decrease.

As Figure 10(a) shows, for binary nodes the barrier solver was consistently faster than TRBP. However, as we increased k, and consequently – the size of the blocks in A, the barrier solver became much slower than TRBP (Figure 10(b)). This seems to support the assumption that the explicit representation of block structure was responsible for TRBP's superior performance in our benchmark set. In the benchmark set, k was (at least) in the order of dozens. We should also note, that even for binary problems, the barrier method will run out of memory much faster than TRBP.

7. Discussion

As pointed out in (Bixby, 2001), advances in hardware and in LP algorithms have greatly expanded the size of problems that can be solved using LP relaxations. Despite this progress, many real world problems are still too large to be handled using desktop hardware and standard LP solvers. In this paper we have experimented with the powerful solvers in CPLEX on LP relaxations of the MAP problem for graphical models from the fields of computer vision and computational biology. Despite

the many optimizations in CPLEX for exploiting sparsity, we found that many of the graphical models gave rise to linear programs that were beyond the capability of all the solvers in CPLEX. In contrast, tree-reweighted BP could be applied to all the linear programs in our database and almost always gave faster solutions. By running the junction tree algorithm on a reduced graphical model defined by the nodes for which the TRBP beliefs had ties, we could find the MAP solution for a large range of real-world problems.

The LP solvers available in CPLEX are of course only a subset of the large number of LP algorithms suggested in the literature and it may very well be possible to design LP solvers that outperform TRBP on our benchmark set. To stimulate research in this direction, both the linear programs used in this paper and our implementation of TRBP are available on the internet. One direction of research that we are currently working on, involves tighter LP relaxations (Meltzer et al., 2005). As the problems become more complex, the standard LP relaxation of MAP is apparently not tight enough and solving the LP often does not enable solving for the MAP. We are exploring methods for solving a sequence of tighter and tighter relaxations and are interested in a method that will allow us to use some of the computations used in one relaxation in solving a tighter relaxation. We believe this research direction offers great potential benefit for interaction between researchers in the field of graphical models and convex optimization.

Acknowledgments

C. Y. is supported by Yeshaya Horowitz Association through the Center for Complexity Science.

Appendix A. Deriving Bounds for the LP Solution Using TRBP

In this section, we give the formula for calculating a bound on the LP solution from TRBP fixed-point beliefs b_{ij}, b_i. We assume that the beliefs have been normalized so that $\max_{x_i,x_j} b_{ij}(x_i,x_j) = 1$ and $\max_{x_i} b_i(x_i) = 1$. Note that this normalization does not change the nature of fixed-points so in case we have any set of fixed-point beliefs, we can just divide every pairwise belief by the maximal value in that belief and similarly divide every singleton belief by its maximal value. The normalized beliefs will still be fixed-points.

It can be shown (Wainwright et al., 2002) that any fixed-point of TRBP satisfies the "admissibility" equation. For any assignment x, the probability (or equivalently the energy) can be calculated from the original potentials or from the beliefs:

$$\begin{aligned} Z\Pr(x) &= \prod_{ij} \Psi_{ij}(x_i,x_j)\Psi_i(x_i) \\ &= K(b)\prod_{ij} b_{ij}^{\rho_{ij}}(x_i,x_j)\prod_i b_i^{c_i}(x_i) \end{aligned}$$

with $c_i = 1 - \sum_j \rho_{ij}$ and $K(b)$ is a constant *independent of* x. $K(b)$ can be calculated from any assignment x, e.g. $x^0 = 0$ where all nodes are in their first state, by

$$K(b) = \frac{\prod_{ij}\Psi_{ij}(x_i^0,x_j^0)\Psi_i(x_i^0)}{\prod_{ij} b_{ij}^{\rho_{ij}}(x_i^0,x_j^0)\prod_i b_i^{c_i}(x_i^0)} \tag{7}$$

Similarly, it can be shown that for any q_{ij}, q_i that satisfy the LP constraints, one can calculate the energy from the beliefs:

$$\begin{aligned} J(q) &= \sum_{<ij>} \sum_{x_i,x_j} q_{ij}(x_i,x_j) E_{ij}(x_i,x_j) + \sum_i \sum_{x_i} q_i(x_i) E_i(x_i) \\ &= -\ln K(b) - \sum_{<ij>} \rho_{ij} \sum_{x_i,x_j} q_{ij}(x_i,x_j) \ln b_{ij}(x_i,x_j) - \sum_i c_i \sum_{x_i} q_i(x_i) \ln b_i(x_i). \end{aligned}$$

By using the admissibility constraint and the properties of the numbers ρ_{ij}, c_i it can be shown that $J(q) \geq -\ln K(b)$. Direct inspection shows that if $q_i j, q_i$ are the sharpened beliefs then they achieve the bound (since they are nonzero only when $b_{ij}(x_i,x_j) = 1$ or $b_i(x_i) = 1$).

References

D. Bertismas and J. Ttsitskikilis. *Introduction to linear optimization*. Athena Scientific, 1997.

S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, 1998.

R. E. Bixby. Solving real-world linear programs: a decade and more of progress. *Operations Research*, 2001.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 1999.

A. Canutescu, A. Shelenkov, and R. L. Dunbrack. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Sci*, 12(9):2001–2014, 2003.

R. Cowell. Advanced inference in Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1998.

R. L. Dunbrack and M. Kurplus. Back-bone dependent rotamer library for proteins: Application to side-chain predicrtion. *J. Mol. Biol*, 230:543–574, 1993.

J. Feldman, D. Karger, and M. J. Wainwright. LP decoding. In *Allerton Conference on Communication, Control, and Computing*, 2003.

P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *Proceedings of IEEE CVPR*, pages 261–268, 2004.

A. S. Fraenkel. Protein folding, spin glass and computational complexity. In *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, held at the University of Pennsylvania, June 23 – 25, 1997*, pages 175–191, 1997.

C. L. Kingsford, B. Chazelle, and M. Singh. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, 21(7):1028–1039, 2005. doi: 10.1093/bioinformatics/bti144.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. In *Proceedings AI Stats*, 2005.

V. Kolmogorov and M. J. Wainwright. On the optimality of tree-reweighted max-product message passing. In *Uncertainty in Artificial Intelligence (UAI)*, 2005.

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

T. Kortemme, A. V. Morozov, and D. Baker. An orientation-dependent hydrogen bonding potential improves prediction of specificity and structure for proteins and protein-protein complexes. *Journal of Molecular Biology*, 326(4):1239–1259, 2003.

B. Kuhlman and D. Baker. Native protein sequences are close to optimal for their structures. *PNAS*, 97(19):10383–10388, 2000.

T. Lazaridis and M. Karplus. Effective energy function for proteins in solution. *Proteins: Structure, Function, and Genetics*, 35(2):133–152, 1999.

R. Marinescu, K. Kask, and R. Dechter. Systematic vs. non-systematic algorithms for solving the MPE task. In *Proceedings of Uncertainty in Artificial Intelligence (UAI 2003)*, 2003.

T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *Proceedings International Conference on Computer Vision (ICCV)*, 2005.

K. Murphy. The bayes net toolbox for matlab. *Computing Science and Statistics*, 33, 2001.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

N. A. Pierce and E. Winfree. Protein Design is NP-hard. *Protein Eng.*, 15(10):779–782, 2002.

C. A. Rohl, C. E. M. Strauss, D. Chivian, and D. Baker. Modeling structurally variable regions in homologous proteins with rosetta. *Proteins: Structure, Function, and Bioinformatics*, 55(3): 656–677, 2004.

E. G. Santos. On the generation of alternative explanations with implications for belief revision. In *UAI*, 1991.

D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Compter Vision*, 2002.

D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume 1, pages 195–202, 2003.

Y. Shimony. Finding the MAPs for belief networks is NP-hard. *Aritifical Intelligence*, 68(2):399–410, 1994.

A. G. Street and S. L. Mayo. Computational protein design. *Structure with folding and design*, 7: r105–r109, 1999.

J. Sun, H.-Y. Shum, and N.-N. Zheng. Stereo matching using belief propagation. In *ECCV*, volume 2, pages 510–524, 2002.

M. Tappen and W. T. Freeman. Graph cuts and belief propagation for stereo, using identical MRF parameters. In *ICCV*, 2003.

E. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

M. J. Wainwright, T. Jaakkola, and A. S. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. In *Allerton Conference on Communication, Control, and Computing*, 2002.

C. Yanover and Y. Weiss. Approximate inference and protein folding. *Advances in Neural Information Processing Systems*, 2002.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *IJCAI (distinguished lecture track)*, 2001.

Journal of Machine Learning Research 7 (2006) 1909–1936 Submitted 10/05; Revised 3/06; Published 9/06

Incremental Support Vector Learning: Analysis, Implementation and Applications

Pavel Laskov LASKOV@FIRST.FHG.DE
Christian Gehl CGEHL@FIRST.FHG.DE
Stefan Krüger KRUEGERS@FIRST.FHG.DE
Fraunhofer-FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany

Klaus-Robert Müller KLAUS@FIRST.FHG.DE
Fraunhofer-FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany
and
University of Potsdam
August-Bebelstrasse 89
14482 Potsdam, Germany

Editors: Kristin P. Bennett, Emilio Parrado-Hernández

Abstract

Incremental Support Vector Machines (SVM) are instrumental in practical applications of online learning. This work focuses on the design and analysis of efficient incremental SVM learning, with the aim of providing a fast, numerically stable and robust implementation. A detailed analysis of convergence and of algorithmic complexity of incremental SVM learning is carried out. Based on this analysis, a new design of storage and numerical operations is proposed, which speeds up the training of an incremental SVM by a factor of 5 to 20. The performance of the new algorithm is demonstrated in two scenarios: learning with limited resources and active learning. Various applications of the algorithm, such as in drug discovery, online monitoring of industrial devices and and surveillance of network traffic, can be foreseen.

Keywords: incremental SVM, online learning, drug discovery, intrusion detection

1. Introduction

Online learning is a classical learning scenario in which training data is provided one example at a time, as opposed to the batch mode in which all examples are available at once (e.g. Robbins and Munro (1951); Murata (1992); Saad (1998); Bishop (1995); Orr and Müller (1998); LeCun et al. (1998); Murata et al. (2002)).

Online learning is advantageous when dealing with (a) very large or (b) non-stationary data. In the case of non-stationary data, batch algorithms will generally fail if ambiguous information, e.g. different distributions varying over time, is present and is erroneously integrated by the batch algorithm (cf. Murata (1992); Murata et al. (2002)). Many problems of high interest in machine learning can be naturally viewed as online ones. An important practical advantage of online algorithms is

that they allow to incorporate additional training data, when it is available, without re-training from scratch. Given that training is usually the most computationally intensive task, it is not surprising that availability of online algorithms is a major pre-requisite imposed by practitioners that work on large data sets (cf. LeCun et al. (1998)) or even have to perform real-time estimation tasks for continuous data streams, such as in intrusion detection (e.g. Laskov et al. (2004); Eskin et al. (2002)), web-mining (e.g. Chakrabarti (2002)) or brain computer interfacing (e.g. Blankertz et al. (2003)).

In the 1980's online algorithms were investigated in the context of PAC learning (e.g. Angluin (1988); Littlestone et al. (1991)). With the emergence of Support Vector Machines (SVM) in the mid-1990's, interest to online algorithms for this learning method arose as well. However, early work on this subject (e.g. Syed et al., 1999; Rüping, 2002; Kivinen et al., 2001; Ralaivola and d'Alché Buc, 2001) provided only approximate solutions.

An exact solution to the problem of online SVM learning has been found by Cauwenberghs and Poggio (2001). Their incremental algorithm (hereinafter referred to as a C&P algorithm) updates an optimal solution of an SVM training problem after one training example is added (or removed).

Unfortunately acceptance of the C&P algorithm in the machine learning community has been somewhat marginal, not to mention that it remains widely unknown to potential practitioners. Only a handful of follow-up publications is known that extend this algorithm to other related learning problems (e.g. Martin, 2002; Ma et al., 2003; Tax and Laskov, 2003); to our knowledge, no successful practical applications of this algorithm has been reported.

At a first glance, a limited interest to incremental SVM learning may seem to result the absence of well-accepted implementations, such as its counterparts SVM^{light} (Joachims, 1999), SMO (Platt, 1999) and LIBSVM (Chang and Lin, 2000) for batch SVM learning. The original Matlab implementation by the authors[1] has essentially the semantics of batch learning: training examples are loaded all at once (although learned one at a time) and unlearning is only used for computation of the – ingenious – leave-one-out bound.

There are, however, deeper reasons why incremental SVM may not be so easy to implement. To understand them – and to build a foundation for an efficient design and implementation of the algorithm, a detailed analysis of the incremental SVM technique is carried out in this paper. In particular we address the "accounting" details, which contain pitfalls of performance bottlenecks unless underlying data structures are carefully designed, and analyze convergence of the algorithm.

The following are the main results of our analysis:

1. Computational complexity of a minor iteration of the algorithm is quadratic in the number of training examples learned so far. The actual runtime depends on the balance of memory access and arithmetic operations in a minor iteration.

2. The main incremental step of the algorithm is guaranteed to bring progress in the objective function if a kernel matrix is positive semi-definite.

Based on the results of our analysis, we propose a new storage design and organization of computation for a minor iteration of the algorithm. The idea is to judiciously use row-major and column-major storage of matrices, instead of one-dimensional arrays, in order to possibly eliminate selection operations. The second building block of our design is gaxpy-type matrix-vector multiplication, which allows to further minimize selection operations which cannot be eliminated by storage

1. http://bach.ece.jhu.edu/pub/gert/svm/incremental/

design alone. Our experiments show that the new design improves computational efficiency by the factor of 5 to 20.

To demonstrate applicability of incremental SVM to practical applications, two learning scenarios are presented. Learning with limited resources allows to learn from large data sets with as little as 2% of the data needed to be stored in memory. Active learning is another powerful technique by means of which learning can be efficiently carried out in large data sets with limited availability of labels. Various applications of the algorithm, such as in drug discovery, online monitoring of industrial devices and and surveillance of network traffic, can be foreseen.

In order to make this contribution self-contained, we begin with the presentation of the C&P algorithm, highlighting the details that are necessary for a subsequent analysis. An extension of the basic incremental SVM to one-class classification is presented in Section 3. Convergence analysis of the algorithm is carried out in Section 4. Analysis of computational complexity, design of efficient storage and organization of operations are presented and evaluated in Section 5. Finally, potential applications of incremental SVM for learning with limited resources and active learning are illustrated in Section 6.

2. Incremental SVM Algorithm

In this section we present the basic incremental SVM algorithm. Before proceeding with our presentation we need to establish some notation.

2.1 Preliminaries

We assume the training data and their labels are given by a set

$$\{(x_1,y_1),\ldots,(x_n,y_n)\}.$$

The inner product between data points in a feature space is defined by a kernel function $k(x_i,x_j)$. The $n\times n$ kernel matrix K^0 contains the inner product values for all $1\le i,j\le n$. The matrix K is obtained from the kernel matrix by incorporating the labels:

$$K = K^0 \odot (yy^T).$$

The operator $\odot$ denotes the element-wise matrix product, and a vector y denotes labels as an $n\times 1$ vector. Using this notation, the SVM training problem can be formulated as

$$\max_{\mu} \min_{\substack{0\le\alpha\le C\\ y^T\alpha=0}} W := -1^T\alpha + \tfrac{C}{2}\alpha^T K\alpha + \mu y^T\alpha. \tag{1}$$

Unlike the classical setting of the SVM training problem, which is usually formulated as maximization or minimization, the problem (1) is a saddle-point formulation obtained by incorporating the equality constraint directly into the cost function. The reason for such construction will become clear shortly.

2.2 Derivation of the Basic Incremental SVM Algorithm

The main building block of the incremental SVM is a procedure for adding one example to an existing optimal solution. When a new point x_c is added, its weight α_c is initially set to 0. If this

assignment is not an optimal solution, i.e. when x_c should become a support vector, the weights of other points and the threshold μ must be updated in order to obtain an optimal solution for the enlarged data set. The procedure can be reversed for a removal of an example: its weight is forced to zero while updating weights of the remaining examples and the threshold μ so that the solution obtained with $\alpha_c = 0$ is optimal for the reduced data set. For the remaining part of this paper we only consider addition of examples.

The saddle point of the problem (1) is given by the Kuhn-Tucker conditions:

$$g_i := -1 + K_{i,:}\alpha + \mu y_i \begin{cases} \geq 0, & \text{if } \alpha_i = 0 \\ = 0, & \text{if } 0 < \alpha_i < C \\ \leq 0, & \text{if } \alpha_i = C \end{cases} \tag{2}$$

$$\frac{\partial W}{\partial \mu} := y^T \alpha = 0. \tag{3}$$

Before an addition of a new example x_c, the Kuhn-Tucker conditions are satisfied for all previous examples. The goal of the weight update in the incremental SVM algorithm is to find a weight assignment such that the Kuhn-Tucker conditions are satisfied for the enlarged data set.

Let us introduce some further notation. Let the set S denote unbounded support vectors ($0 < \alpha_i < C$), the set E denote bounded support vectors ($\alpha_i = C$), and the set O denote non-support vectors ($\alpha_i = 0$); let $R = E \cup O$. These index sets induce respective partitions on the kernel matrix K and the label vector y (we shall use the lower-case letters s, e, o and r for such partitions).

By writing out the Kuhn-Tucker conditions (2)–(3) before and after an update $\Delta\alpha$ we obtain the following condition that must be satisfied after an update:

$$\begin{bmatrix} \Delta g_c \\ \Delta g_s \\ \Delta g_r \\ 0 \end{bmatrix} = \begin{bmatrix} y_c & K_{cs} \\ y_s & K_{ss} \\ y_r & K_{rs} \\ 0 & y_s^T \end{bmatrix} \underbrace{\begin{bmatrix} \Delta\mu \\ \Delta\alpha_s \end{bmatrix}}_{\Delta s} + \Delta\alpha_c \begin{bmatrix} K_{cc}^T \\ K_{cs}^T \\ K_{cr}^T \\ y_c \end{bmatrix}. \tag{4}$$

One can see that $\Delta\alpha_c$ is in equilibrium with $\Delta\alpha_s$ and μ: any change to $\Delta\alpha_c$ must be absorbed by the appropriate changes in $\Delta\alpha_s$ and μ in order for the condition (4) to hold.

The main equilibrium condition (4) can be further refined as follows. It follows from (2) that $\Delta g_s = 0$. Then lines 2 and 4 of the system (4) can be re-written as

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & \alpha_s^T \\ \alpha_s & K_{ss} \end{bmatrix} \Delta s + \begin{bmatrix} \alpha_c \\ K_{cs}^T \end{bmatrix} \Delta\alpha_c. \tag{5}$$

This linear system is easily solved for Δs as follows:

$$\Delta s = \beta \Delta\alpha_c, \tag{6}$$

where

$$\beta = -\underbrace{\begin{bmatrix} 0 & \alpha_s^T \\ \alpha_s & K_{ss} \end{bmatrix}^{-1}}_{Q} \underbrace{\begin{bmatrix} \alpha_c \\ K_{cs}^T \end{bmatrix}}_{\vec{\eta}} \tag{7}$$

is the gradient of the manifold of optimal solutions parameterized by α_c.

One can further substitute (6) into lines 1 and 3 of the system (4):

$$\begin{bmatrix}\Delta g_c \\ \Delta g_r\end{bmatrix} = \gamma \Delta\alpha_c, \tag{8}$$

where

$$\gamma = \begin{bmatrix} y_c & K_{cs} \\ y_r & K_{rs}\end{bmatrix}\beta + \begin{bmatrix}K_{cc} \\ K_{cr}^T\end{bmatrix} \tag{9}$$

is the gradient of the manifold of gradients g_r at an optimal solution parameterized by α_c.

The upshot of these derivations is that the update is controlled by very simple sensitivity relations (6) and (8), where β is sensitivity of Δs with respect to $\Delta\alpha_c$ and γ is sensitivity of $\Delta g_{c,r}$ with respect to $\Delta\alpha_c$.

2.3 Accounting

Unfortunately the system (4) cannot be used directly to obtain the new SVM state. The problem lies in the changing composition of the sets S and R with the change of Δs and $\Delta\alpha_c$ in Eq. (4). To handle this problem, the main strategy of the algorithm is to identify the largest increase $\Delta\alpha_c$ such that some point migrates between the sets S and R. Four cases must be considered to account for such structural changes:

1. Some α_i in S reaches a bound (an upper or a lower one). Let ε be a small number. Compute the sets

   $$I_+^S = \{i \in S : \beta_i > \varepsilon\}$$
   $$I_-^S = \{i \in S : \beta_i < -\varepsilon\}.$$

   The examples in set I_+^S have positive sensitivity with respect to the weight of the current example; that is, their weight would increase by taking the step $\Delta\alpha_c$.[2] These examples should be tested for reaching the upper bound C. Likewise, the examples in set I_-^S should be tested for reaching zero. The examples with $-\varepsilon < \beta_i < \varepsilon$ should be ignored, as they are insensitive to $\Delta\alpha_c$. Thus the possible weight updates are

   $$\Delta\alpha_i^{\max} = \begin{cases} C - \alpha_i, & \text{if } i \in I_+^S \\ -\alpha_i, & \text{if } i \in I_-^S, \end{cases}$$

   and the largest possible $\Delta\alpha_c^S$ before some example in S moves to R is

   $$\Delta\alpha_c^S = \operatorname*{absmin}_{i \in I_+^S \cup I_-^S} \frac{\Delta\alpha_i^{\max}}{\beta_i}, \tag{10}$$

   where

   $$\operatorname*{absmin}_i(x) := \min_i |x_i| \cdot \operatorname{sign}(x_{(\operatorname*{argmin}_i |x_i|)}).$$

2. Some g_i in R reaches zero. Compute the sets

   $$I_+^R = \{i \in E : \gamma_i > \varepsilon\}$$
   $$I_-^R = \{i \in O : \gamma_i < -\varepsilon\}.$$

2. It can be shown that the step $\Delta\alpha_c$ is always positive in the incremental case.

The examples in set I_+^R have positive sensitivity of the gradient with respect to the weight of the current example. Therefore their (negative) gradients can potentially reach zero. Likewise, gradients of the examples in set I_-^R are positive but are pushed towards zero with the increasing weight of the current example. Thus the largest increase $\Delta\alpha_c^g$ before some point in R moves to S can be computed as

$$\Delta\alpha_c^R = \min_{i \in I_+^R \cup I_-^R} \frac{-g_i}{\gamma_i}. \tag{11}$$

3. g_c becomes zero. This case is similar to case 2, with the feasibility test in the form

$$\gamma_c > \varepsilon.$$

If the update is feasible the largest step $\Delta\alpha_c^g$ is computed as

$$\Delta\alpha_c^g = \frac{-g_c}{\gamma_c}. \tag{12}$$

4. α_c reaches C. The largest possible increment $\Delta\alpha_c^\alpha$ is clearly

$$\Delta\alpha_c^\alpha = C - \alpha_c. \tag{13}$$

Finally, the smallest of the four values

$$\Delta\alpha_c^{\max} = \min\left(\Delta\alpha_c^S, \Delta\alpha_c^R, \Delta\alpha_c^g, \Delta\alpha_c^\alpha\right) \tag{14}$$

constitutes the largest possible increment of α_c.

After the largest possible increment of α_c is determined, the updates Δs and Δg are carried out. The inverse matrix Q must be also re-computed in order to account for the new composition of the set S. Efficient update of this matrix is presented in section 2.4. The sensitivity vectors β and γ must be also re-computed. The process repeats until the gradient of the current example becomes zero or its weight reaches C. The high-level summary of incremental SVM algorithm is given in Algorithm 1.

2.4 Recursive Update of the Inverse Matrix

It is clearly infeasible to explicitly invert the matrix Q in Eq. (7) every time the set S is changed. Luckily, it is possible to use the fact that this set is always updated one element at a time – an example is either added to, or removed from the set S.

Consider addition first. When the example x_k[3] is added to the set S, the matrix to be inverted is partitioned as follows:

$$R := \begin{bmatrix} 0 & y_s^T & y_k \\ y_s & K_{ss} & K_{ks}^T \\ y_k & K_{ks} & K_{kk} \end{bmatrix} = \begin{bmatrix} Q^{-1} & \eta_k \\ \eta_k^T & K_{kk} \end{bmatrix}, \tag{15}$$

where

$$\eta_k = \begin{bmatrix} y_k \\ K_{ks}^T \end{bmatrix}.$$

3. A different index k is used to emphasize that this example is not the same as the current example x_c.

Algorithm 1 Incremental SVM algorithm: high-level summary.

1: Read example x_c, compute g_c.
2: **while** $g_c < 0$ & $\alpha_c < C$ **do**
3: Compute β and γ according to (7), (9).
4: Compute $\Delta\alpha_c^S$,$\Delta\alpha_c^R$,$\Delta\alpha_c^g$ and $\Delta\alpha_c^\alpha$ according to (10)–(13).
5: Compute $\Delta\alpha_c^{\max}$ according to (14).
6: $\alpha_c \leftarrow \alpha_c + \Delta\alpha_c^{\max}$.
7: $\alpha_s \leftarrow \beta\Delta\alpha_c^{\max}$
8: $g_{c,r} \leftarrow \gamma\Delta\alpha_c^{\max}$
9: Let k be the index of the example yielding the minimum in (14).
10: **if** $k \in S$ **then**
11: Move k from S to either E or O.
12: **else if** $k \in E \cup O$ **then**
13: Move k from either E or O to S.
14: **else**
15: $\{k = c$: do nothing, the algorithm terminates.$\}$
16: **end if**
17: Update Q recursively. $\{$See section 2.4.$\}$
18: **end while**

Define $\beta_k = -Q\eta_k$, and denote the enlarged inverse matrix by $\tilde{Q}$. Applying the Sherman-Morrison-Woodbury formula for block matrix inversion (see e.g. Golub and van Loan (1996)) to $\tilde{Q}$, we obtain:

$$\underbrace{\begin{bmatrix} Q^{-1} & \eta_k \\ \eta_k^T & K_{kk} \end{bmatrix}^{-1}}_{\tilde{Q}} = \begin{bmatrix} Q + \kappa^{-1}(Q\eta_k)(Q\eta_k)^T & -\kappa^{-1}Q\eta_k \\ -\kappa^{-1}(Q\eta_k)^T & \kappa^{-1} \end{bmatrix}$$

$$= \begin{bmatrix} Q + \kappa^{-1}\beta_k\beta_k^T & \kappa^{-1}\beta_k \\ \kappa^{-1}\beta_k^T & \kappa^{-1} \end{bmatrix}$$

$$= \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\kappa}\begin{bmatrix} \beta_k \\ 1 \end{bmatrix}\begin{bmatrix} \beta_k^T & 1 \end{bmatrix}, \tag{16}$$

where

$$\kappa = K_{kk} - \eta_k^T Q\eta_k. \tag{17}$$

Thus the update of the inverse matrix involves expansion with a zero row and column and addition of a rank-one matrix obtained via a matrix-vector multiplication. The running time needed for an update of the inverse matrix is quadratic in the size of Q, which is much better than explicit inversion.

The removal case is straightforward. Knowing the inverse matrix $\tilde{Q}$ and using (16), we can write

$$\tilde{Q} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} = \begin{bmatrix} Q + \kappa^{-1}\beta_k\beta_k^T & \kappa^{-1}\beta_k \\ \kappa^{-1}\beta_k^T & \kappa^{-1} \end{bmatrix}.$$

Re-writing this block matrix expression component-wise, we have:

$$Q = q_{11} - \kappa^{-1}\beta_k\beta_k^T \tag{18}$$
$$\beta_k = \kappa q_{12} = q_{12}/q_{22} \tag{19}$$
$$\beta_k^T = \kappa q_{21} = q_{21}/q_{22} \tag{20}$$
$$\kappa^{-1} = q_{22}. \tag{21}$$

Substituting the last three relations into the first we obtain:

$$Q = q_{11} - \frac{q_{12}q_{21}}{q_{22}}. \tag{22}$$

The running time of the removal operations is also quadratic in the size of Q.

3. Extension to One-Class Classification

Availability of labels, especially online, may not be possible in certain applications. For example, analysis of security logs is extremely time-consuming, and labels may be available only in limited quantities after forensic investigation. As another example, monitoring of critical infrastructures, such as power lines or nuclear reactors, must prevent a system from reaching a failure state which may lead to gravest consequences. Nevertheless, algorithms similar to SVM classification can be applied for data-description, i.e. for automatic inference of a concept descriptions deviations from which are to be considered abnormal. Since examples of the class to be learned are not (or rarely) available, the problem is known as "one-class classification".

The two well-known approaches to one-class classification are separation of data points from the origin (Schölkopf et al., 2001) and spanning of data points with a sphere (Tax and Duin, 1999). Although they use different geometric constructions, these approaches lead to similar, and in certain cases even identical, formulations of dual optimization problems. As we shall see, incremental learning can be naturally extended to both of these approaches.

The dual formulation of the "sphere" one-class classification is given by the following quadratic program:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i k(x_i,x_i) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j k(x_i,x_j) \\ \text{subject to:} \quad & 0 \le \alpha_i \le C,\ i = 1,\ldots,n \\ & \sum_{i=1}^{n} \alpha_i = 1. \end{aligned} \tag{23}$$

In order to extend the C&P algorithm for this problem consider the following abstract saddle-point quadratic problem:

$$\max_{\mu} \min_{\substack{0\le\alpha\le C \\ a^T\alpha+b=0}} W := -c^T\alpha + \tfrac{C}{2}\alpha^T H\alpha + \mu(a^T\alpha + b). \tag{24}$$

It can be easily seen that formulation (24) generalizes both problems (23) and (1), subject to following definition of the abstract parameters:

$$c = \mathrm{diag}(K), \quad H = K, \quad a = 1, \quad b = 1.$$

Algorithm 2 Initialization of incremental one-class SVM.

1: Take the first $\lfloor \frac{1}{C} \rfloor$ objects, assign them weight C and put them in E.
2: Take the next object c, assign $\alpha_c = 1 - \lfloor \frac{1}{C} \rfloor C$ and put it in S.
3: Compute the gradients g_i of all objects, using Eq. (2).
4: Compute μ so as to ensure non-positive gradients in E: $\mu = -\max_{i \in E} g_i$
5: Enter the main loop of the incremental algorithm.

The C&P algorithm presented in sections 2.2–2.4 can be applied to formulation (24) almost without modification. The only necessary adjustment is a special initialization procedure for identification of an initial feasible solution presented in Algorithm 2.[4]

4. Convergence Analysis

A very attractive feature of Algorithm 1 is that it obviously makes progress to an optimal solution if a non-zero update $\Delta\alpha_c$ is found. Thus potential convergence problems arise only when a zero update step is encountered. This can happen in several situations. First, if the set S is empty, no non-zero update of $\Delta\alpha_c$ is possible since otherwise the equality constraint of the SVM training problem is violated. Second, a zero update can occur when two or more points simultaneously migrate between the index sets. In this case, each subsequent point requires a structural update without improvement of $\Delta\alpha_c$. Furthermore, it must be guaranteed, that after a point migrates from one set to another, say from E to S, it is not immediately thrown out after the structure and the sensitivity parameters are re-computed. These issues constitute the scope of convergence analysis to be covered in this section. In particular we present a technique for handling the special case of the empty set S and show that immediate cycling is impossible if a kernel matrix is positive semi-definite.

4.1 Empty Set S

The procedure presented in the previous two sections requires a non-empty set of unbounded support vectors, otherwise a zero matrix must be inverted in Eq. (7). To handle this situation, observe that the main instrument needed to derive the sensitivity relations is pegging of the gradient to zero for unbounded support vectors, which follows from the Kuhn-Ticker conditions (2). Notice that the gradient can also be zero for some points in sets E and O. Therefore, if the set S is empty we can freely move some examples with zero gradients from the sets E and O to it and continue from there. The question arises: what if no points with zero gradients can be found in E or O?

With $\Delta\alpha = 0$, $\Delta\alpha_c = 0$ and no examples in the set S, the equilibrium condition (4) reduces to

$$\begin{aligned} \Delta g_c &= y_c \Delta\mu \\ \Delta g_r &= y_r \Delta\mu. \end{aligned} \tag{25}$$

This is an equilibrium relation between $\Delta g_{c,r}$ and the scalar $\Delta\mu$, in which sensitivity is given by the vector $[y_c; y_r]$. In other words, one can change μ freely until one of components in g_r or g_c hits zero, which would allow an example to be brought into S.

4. Through the presentation of the C&P algorithm it was assumed that a feasible solution is always available. This is indeed no problem for the classification SVM since the zero solution is always feasible. For the sphere formulation of one-class classification this is not the case.

The problem remains – since $\Delta\mu$ is free as opposed to non-negative $\Delta\alpha_c$ – to determine the direction in which the components of g_r are pushed by changes in μ. This can be done by first solving (25) for $\Delta\mu$, which yields the dependence of Δg_r on Δg_c:

$$\Delta g_r = -\frac{y_r}{y_c}\Delta g_c.$$

Since Δg_c must be non-negative (gradient of the current example is negative and should be brought to zero if possible), the direction of Δg_r is given by $-\frac{y_r}{y_c}$. Hence the feasibility conditions can be formulated as

$$I_+^R = \{i \in E : -\frac{y_i}{y_c} > \varepsilon\}$$
$$I_-^R = \{i \in O : -\frac{y_i}{y_c} < -\varepsilon\}.$$

The rest of the argument essentially follows the main case. The largest possible step $\Delta\mu^R$ is computed as

$$\Delta\mu^R = \min_{i \in I_+^R \cup I_-^R} \frac{-g_i}{y_i}. \tag{26}$$

and the largest possible step $\Delta\mu^c$ is computed as

$$\Delta\mu^c = -\frac{g_c}{y_c}. \tag{27}$$

Finally, the update $\Delta\mu^{\max}$ is chosen as

$$\Delta\mu^{\max} = \min(\Delta\mu^R, \Delta\mu^c). \tag{28}$$

4.2 Immediate Cycling

A more dangerous situation can occur if an example entering the set S is immediately thrown out at a next iteration without any progress. It is not obvious why this kind of immediate cycling cannot take place, since the sensitivity information contained in vectors β and γ does not seem to provide a clue what would happen to an example after the structural change. The analysis provided in this section shows that, in fact, such look-ahead information is present in the algorithm and that this property is related to positive semi-definiteness of a kernel matrix.

When an example is added to the set S, the corresponding entry in the sensitivity vector β is added at the end of this vector. Let us compute $\tilde{\beta}$ *after an addition of example k to set S*:

$$\tilde{\beta} = -\tilde{Q}\eta = -\begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} y_c \\ K_{cs} \end{bmatrix} - \frac{1}{\kappa}\begin{bmatrix} \beta_k\beta_k^T & \beta_k \\ \beta_k^T & 1 \end{bmatrix}\begin{bmatrix} y_c \\ K_{cs} \end{bmatrix}.$$

Computing the last line in the above matrix products we obtain:

$$\tilde{\beta}_{\text{end}} = -\frac{1}{\kappa}\underbrace{\left(\beta_k^T\begin{bmatrix} y_c \\ K_{cs\setminus\text{end}} \end{bmatrix} + K_{ck}\right)}_{\gamma_k}. \tag{29}$$

Let us assume that κ (defined in Eq. (17)) is non-negative. Examples with $\kappa = 0$ must be prevented from entering the set S externally, otherwise invertibility of Q is ruined; therefore $\kappa > 0$

for the examples entering the set S. Thus we can see that the sign of $\tilde{\beta}_{\text{end}}$ after the addition of the element k to set S is *the opposite* to the sign of γ_k before the addition. Recalling the feasibility conditions for inclusion of examples in set S, one can see that if an example is joining from the set O its $\tilde{\beta}$ after inclusion will be positive, and if an example is joining from the set E its $\tilde{\beta}$ after inclusion will be negative. Therefore, in no case will an example be immediately thrown out of the set S.

Thus to prove that immediate cycling is impossible it has to be shown that $\kappa \geq 0$. Two technical lemmas are useful before we proceed with the proof of this fact.

Lemma 1 *Let $z = -Q\eta_k$, $\tilde{z} = [z,1]^T$. Then $\kappa = \tilde{z}^T R \tilde{z}$.*

Proof By writing out the quadratic form we obtain:

$$\begin{aligned}\tilde{z}^T R \tilde{z} &= z^T Q^{-1} z + \eta_k^T z + z^T \eta_k + K_{kk} \\ &= \eta_k^T Q Q^{-1} Q \eta_k - 2\eta_k Q \eta_k + K_{kk} \\ &= K_{kk} - \eta_k Q \eta_k.\end{aligned}$$

The result follows by the definition of κ. ■

The next lemma establishes a sufficient condition for non-negativity of a quadratic form with a matrix of the special structure possessed by matrix R.

Lemma 2 *Let $\tilde{x} = [x_0, x]^T$, $\tilde{K} = \begin{bmatrix} 0 & y^T \\ y & K \end{bmatrix}$, where x_0 is a scalar, x,y are vectors of length n, and K is a positive semi-definite $n \times n$ matrix. If $x^T y = 0$ then $\tilde{x}^T \tilde{K} \tilde{x} \geq 0$.*

Proof By writing out the quadratic form we obtain

$$\tilde{x}^T \tilde{K} \tilde{x} = 2x_0 x^T y + x^T K x.$$

Since K is positive semi-definite the second term is greater than or equal to 0, whereas the first term vanished by the assumption of the lemma. ■

Finally, the intermediate results of the lemmas are used in the main theorem of this section.

Theorem 3 *If the kernel matrix K is positive semi-definite then $\kappa \geq 0$.*

Proof Lemma 1 provides a quadratic form representation of κ. Our goal is thus to establish its non-negativity using the result of Lemma 2.

Using the partition matrix inversion formula we can write Q (cf. Eq. 7) as

$$Q = \begin{bmatrix} -\frac{1}{\delta} & \frac{1}{\delta} y_s^T K_{ss}^{-1} \\ \frac{1}{\delta} K_{ss}^{-1} y_s & K_{ss}^{-1} - \frac{1}{\delta} K_{ss}^{-1} y_s y_s^T K_{ss}^{-1} \end{bmatrix},$$

where $\delta = y_s^T K_{ss}^{-1} y_s$. Substituting Q into the definition of z in Lemma 1 and explicitly writing out the first term, we obtain:

$$z := \begin{bmatrix} z_0 \\ z_{\setminus 0} \end{bmatrix} = -\begin{bmatrix} -\frac{1}{\delta} & \frac{1}{\delta} y_s^T K_{ss}^{-1} \\ \frac{1}{\delta} K_{ss}^{-1} y_s & K_{ss}^{-1} - \frac{1}{\delta} K_{ss}^{-1} y_s y_s^T K_{ss}^{-1} \end{bmatrix} \begin{bmatrix} y_k \\ K_{sk} \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{\delta} y_k - \frac{1}{\delta} y_s^T K_{ss}^{-1} K_{sk} \\ -\frac{1}{\delta} K_{ss}^{-1} y_s y_k - K_{ss}^{-1} K_{sk} + \frac{1}{\delta} K_{ss}^{-1} y_s y_s^T K_{ss}^{-1} K_{sk} \end{bmatrix}.$$

Then

$$z_{\setminus 0}^T y_s + y_k = -\frac{1}{\delta} y_k \underbrace{y_s^T K_{ss}^{-1} y_s}_{\delta} - K_{sk}^T K_{ss}^{-1} y_s + \frac{1}{\delta} K_{sk}^T K_{ss}^{-1} y_s \underbrace{y_s^T K_{ss}^{-1} y_s}_{\delta} + y_k = 0$$

and the result follows by Lemma 2. ■

5. Runtime Analysis and Efficient Design

Let us now zoom in on computational complexity of Algorithm 1 which constitutes a minor iteration of the overall training algorithm.[5] Asymptotically, the complexity of a minor iteration is quadratic in a number of examples learned so far: re-computation of the gradient, β and γ involve matrix-vector multiplications, which have quadratic complexity, and the recursive update of an inverse matrix has also been shown (cf. Section 2.4) to be quadratic in the number of examples. These estimates have to be multiplied by a number of minor iterations needed to learn an example. The number of minor iterations depends on the structure of a problem, namely on how often examples migrate between the index sets until an optimal solution is found. This number cannot be control in the algorithm, and can be potentially exponentially large.[6]

For the practical purposes it is important to understand the constants hidden in asymptotic estimates. To this end, the analysis of a direct implementation of Algorithm 1 in Matlab is presented in Section 5.1. The focus of our analysis lies on the complexity of the main steps of a minor iteration in terms of arithmetic and memory access operations. This kind of analysis is important because arithmetics can be implemented much more efficiently than memory access. Performance-tuned numeric libraries, such as BLAS[7] or ATLAS,[8] make extensive use of the cache memory which is an order of magnitude faster than the main memory. Therefore, a key to efficiency of the incremental SVM algorithm lies in identifying performance bottlenecks associated with memory access operations and trying to eliminate them in a clever design. The results of our analysis are illustrated by profiling experiments in Section 5.2, in which relative complexity of the main computational operations, as a percentage of total running time, is measured for different kernels and sample sizes.

5. A major iteration corresponds to inclusion of a new example; therefore, all complexity estimates must be multiplied by a number of examples to be learned. This is, however, a (very) worst case scenario, since no minor iteration is needed for many points that do not become support vectors at the time of their inclusion.
6. The structure of the problem is determined by the geometry of a set of feasible solutions in a feature space. Since we essentially follow the outer boundary of the set of feasible solutions, we are bound by the same limitation as linear and non-linear programming in general, for which it is known that problems exist with exponentially many vertices in a set of feasible solutions (Klee and Minty, 1972).
7. http://www.netlib.org/blas/
8. http://math-atlas.sourceforge.net/

It follows from our analysis and experiments that the main difficulty in efficient implementation of incremental SVM indeed lies in selection of non-contiguous elements of matrices, e.g. in Eq. (9). The problem cannot be addressed within Matlab in which storage organization is one-dimensional. Furthermore, it must be realized, as our experience showed us, that merely re-implementing incremental SVM without addressing the tradeoff between selection and arithmetic operations, for example using C++ with one-dimensional storage, *does not* solve the problem. The solution proposed in Section 5.3, which allows to completely eliminate expensive selection operations at a cost of minor increase of arithmetic operations, is based on a mixture of row- and column-major storage and on the gaxpy-type (e.g. Golub and van Loan (1996)) matrix-vector products. The evaluation of the new design presented in Section 5.4 shows performance improvement of 5 to 20 times.

5.1 Computational Complexity of Incremental SVM

On the basis of the pseudo-code of Algorithm 1 we will now discuss the key issues that will later be used for a more efficient implementation of the incremental SVM.

Line 1: Computation of g_c is done according to Eq. (2). This calculation requires partial computation of the kernel row K_{cs} for the current example and examples in the set S. If the condition of the while loop in line 2 does not hold then the rest of the kernel row K_{cr} has to be computed for Eq. (9) in line 3. Computation of a kernel row is expensive since a subset of input points, usually stored as a matrix, has to be selected using the index sets S and R.

Line 3: The computation of γ via Eq. (9) is especially costly since a two-dimensional selection has to be performed to obtain the matrix K_{rs} ($O(sr)$ memory access operations), followed by a matrix-vector multiplication ($O(sr)$ arithmetic operations). The computation of β in line 3 is relatively easy because the inverse matrix Q is present and only the matrix-vector multiplication for Eq. (7) ($O(ss)$ arithmetic operations) has to be performed. The influence of γ and β for the algorithm scales with the size of set S and the number of data points.

Lines 4-8: These lines have minor runtime relevance because only vector-scalar calculations and selections are to be performed.

Lines 9-16: Administration operations for sets S and R have inferior complexity. If a kernel row of the example k is not present (in case of x_k entering the S from O) then it has to be re-computed for the update of the inverse matrix in line 17.

Line 17: The update of the inverse matrix requires the calculation of κ according to Eq. (17) ($O(ss)$ arithmetic operations), which is of a similar order as the computation of β. The expansion and rank-one matrix computation have also effect on the algorithm runtime. The expansion requires memory operation (in the naive implementation, a reallocation of the entire matrix) and is thus expensive for a large inverse matrix.

To summarize, the main performance bottlenecks of the algorithm are lines 1, 3, and 17, in which memory access operations take place.

5.2 Performance Evaluation

We now proceed with experimental evaluation of the findings of Section 5.1. As a test-bed the MNIST handwritten digits data set[9] is used to profile the training of an incremental SVM. For every digit, a test run is made on the data sets of size 1000, 2000, 3000, 5000 and 10,000 randomly drawn from the training data set. Every test run was performed for a linear kernel, a polynomial kernel of degree 2 and an RBF kernel with $\sigma = 30$. Profiles were created by the Matlab profiler.

Eight operations were identified where a Matlab implementation of Algorithm 1 spends the bulk of its runtime (varying from 75% to 95% depending on a digit). Seven of these operations pertain to the bottlenecks identified in Section 5.1. Another relatively expensive operation is augmentation of a kernel matrix with a kernel row. Figure 1 shows proportions of runtime spent in the most expensive operations for the digit 8.

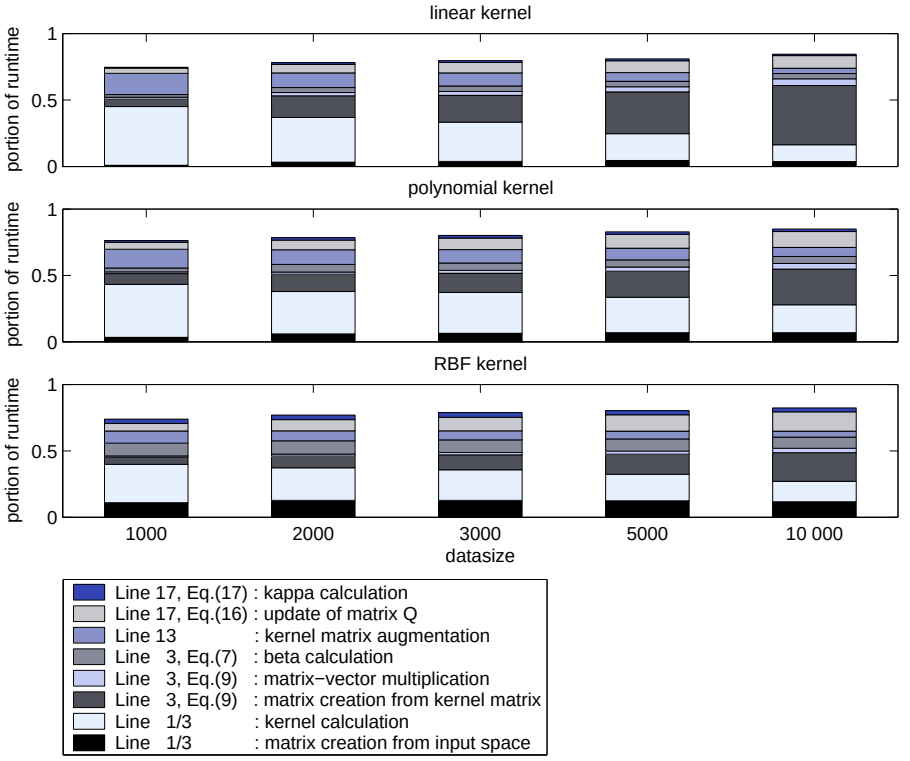


Figure 1: Profiling results for digit 8 (MNIST).

The analysis clearly shows that memory access operations dominate the runtime shown in Figure 1. It also reveals that the portion of kernel computation scales down with increasing data size for all three kernels.

9. This data set can be found at http://yann.lecun.com/exdb/mnist/, and contains 60,000 training and 10,000 test images of size 28×28.

5.3 Organization of Matrix Storage and Arithmetic Computations

Having found the weak spots in the Matlab implementation of the incremental SVM, we will now consider the possibilities for the efficiency improvement. In to order gain control over the storage design we choose C++ as an implementation platform. Numerical operations can be efficiently implemented by using the ATLAS library.

5.3.1 STORAGE DESIGN

As it was mentioned before, the difficulty of selection operations in Matlab result from storing matrices as one-dimensional arrays. For this type of storage, selection of rows and columns necessarily required a large amount of copying.

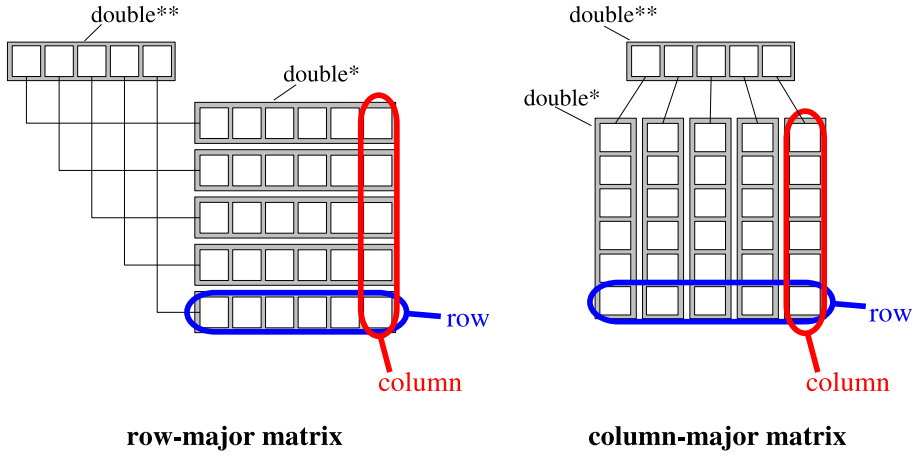


Figure 2: C++ matrix design

An alternative representation of a matrix in C++ is a pointer-to-pointer scheme shown in Figure 2. Depending on whether rows or columns of a matrix are stored in one-dimensional pointed to `double**` pointers, either a row-major or a column-major storage is realized.

What are the benefits of a pointer-to-pointer storage for our purposes? Such matrix representation has the advantage that selection along the pointer-of-pointer array requires hardly any time since only addresses of rows or columns need to be fetched. As a result one can completely eliminate selection operations in line 1 by storing the input data in a column-major matrix. Furthermore, by storing a kernel matrix in a row-major format (a) additions of kernel rows can be carried out without memory relocation, and (b) selections in the matrix K_{rs} are somewhat optimized since $r \gg s$. Another possible problem arises during addition of a new example when columns have to added to a kernel matrix. This problem can be solved by pre-allocation of columns which can be done in constant-time (amortized).

5.3.2 MATRIX OPERATIONS

The proposed memory design does not completely solve the problem: we still have to perform column selection in s when computing K_{rs}. Although tolerable for smaller number of support vectors,

Algorithm 3 C++ calculation of γ for (9) using gaxpy-type matrix-vector multiplication

Create an empty vector z
for $i = 1 : |s|$ **do**
  $z = \beta_{i+1} K_{s_i,:} + z$ {gaxpy call}
end for
Compute $\gamma = \beta_1 y_r + z_r + K_{cr}$

Algorithm 4 C++ calculation of γ for (9) using a naive matrix-vector multiplication

Create a matrix Z
$Z = \left[y_r^T ; K_{sr} \right]$
Compute $\gamma = \beta^T Z + K_{cr}$ {dgemm call}

the problem becomes acute when s grows with the arrival of more and more examples. Yet it turns out to be possible to eliminate even this selection by re-organizing the computation of γ.

Consider the following form of computing Eq. (9):[10]

$$\gamma^T = \beta_1 y_r^T + \beta_{2:\text{end}}^T K_{sr} + K_{cr}. \tag{30}$$

The first and the last terms are merely vectors, while the middle term is computed using a matrix-vector multiplication and selection over a matrix. Using the transposed matrix K_{sr} (still stored in a row-major form) at first seems counter-intuitive, as we argued in the previous section that expensive selection should be carried out over short indices in S and not the long indices in R. However, consider the following observation:

> Since $s \ll r$ we can just as well run the product $\beta_{2:\text{end}}^T K_{s,1:\text{end}}$ at a tolerable extra cost of $O(ss)$ arithmetic operations.

By doing so we do not need to worry about selection! The extra s elements in a product vector can be discarded (of course by a selection, however selection over a vector is cheap).

Still another problem with the form (30) of the γ-update remains. If we run it as an inner-product update, i.e. multiplying a row vector with columns of a matrix stored in a row-major format, this loop must be run over the elements in non-contiguous memory. This is as slow as using selection. However, by running it as a gaxpy-type update (e.g. Golub and van Loan, 1996) we end up with loops running over the *rows* of a kernel matrix, which brings a full benefit of performance-tuned numerics. The summary of the gaxpy-type computation of γ is given in Algorithm 3. For comparison, Algorithm 4 shows a naive implementation of the γ-update using inner-product operations (dgemm-type update).

This same construction can be also applied to the inverse matrix update in Eq. (16) by using the following format:

$$Q_i = \frac{\beta_i}{\kappa} \beta + Q_{i,:}. \tag{31}$$

By doing so explicit creation of a rank-one matrix can be avoided.

To summarize our design, by using the row-major storage of (transposed) matrix K and the gaxpy-type matrix-vector multiplication selection can be avoided by at a cost of extra $O(ss)$ arithmetic operations and performing a selection on a resulting vector.

10. For cleaner notation we ignore the first line in Eq. (9) here.

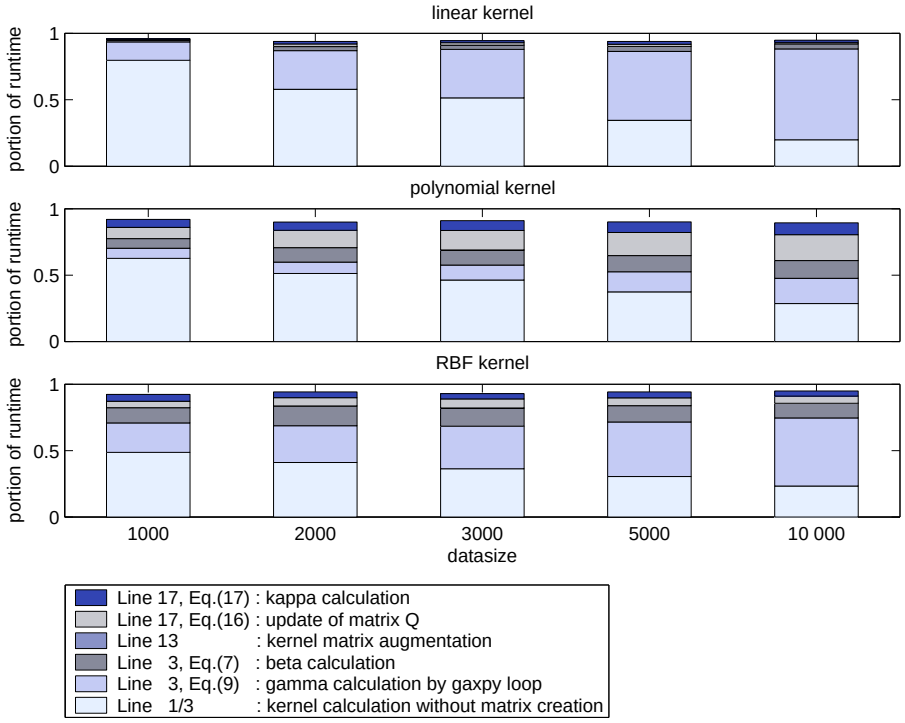


Figure 3: C++ runtime proportion digit 8 (MNIST)

5.4 Experimental Evaluation of the New Design

The main goal of the experiments to be presented in this section is to evaluate the impact of the new design of storage and arithmetic operations on the overall performance of incremental SVM learning. In particular, the following issues are to be investigated:

- How is the runtime profile of the main operations affected by the new design?
- How does the overall runtime of the new design scale with an increasing size of a training set?
- What is the overall runtime improvement over the previous implementations and how does it depend on the size of a training set?

To investigate the runtime profiles of the new design, check-pointing has been realized in our C++ implementation. The profiling experiment presented in Section 5.2 (cf. Figure 1) has been repeated for the new design, and the results are shown in Figure 3. The following effects of the new design can be observed:

1. The selection operation in Lines 1/3 is eliminated. The selection was necessary in a Matlab implementation due to one-dimensional storage – a temporary matrix had to be created in order to represent a sub-matrix of the data. In the new design using the column-major storage for the data matrix a column sub-matrix can be directly passed (as pointers to columns) without a need for selection.

2. Matrix creation has been likewise eliminated in the computation of γ in Line 3. However, the relative cost of the gaxpy computation in the new design remains as high as the relative cost of the combined matrix creation / matrix-vector multiplication operations in the Matlab implementation. The relative cost of the β computation is not affected by the new design.

3. Matrix augmentation in Line 13 takes place at virtually no computational cost – due to row-major storage of the kernel matrix.

4. The relative cost of operations in Line 17 is not affected by the new design.

The overall cost distribution of main operations remains largely the same in the new design, kernel computation having the largest weight for small training sets and gamma computation – for the large training sets. However, as we will see from the following experiments, the new design results in major improvement of the absolute running time.

Evaluation of the absolute running time is carried out by means of the scaling factor experiments. The same data set and the same SVM parameters are used as in the profiling experiments. Four implementations of incremental SVM are compared: the original Matlab implementation of C&P (with leave-one-out error estimation turned off), the Matlab implementation of Algorithm 1, the C++ implementation of Algorithms 1 & 3 and the C++ implementation of Algorithms 1 & 4. The latter configuration is used in order to verify that the performance gains indeed stem from the gaxpy-type updates rather than from switching from Matlab to C++.

The algorithms are run on the data sets ranging from 1000 to 10000 examples in size, and the training times are plotted against the training set size at a log-log scale. These plots are shown in Figure 4 (for the linear kernel) and 5 (for the RBF kernel). The results for the polynomial kernel are similar to the linear kernel and are not shown. Ten plots are shown separately for each of the digits.

One can see that the C++ implementation significantly outperforms both Matlab implementations. The RBF kernel is more difficult for training than the linear kernel for the MNIST data set, which is reflected by a larger proportion of support vectors (on average 15% for the RBF kernel compared to 5% with the linear kernel, at 10000 training examples). Because of this the experiments with the C&P algorithm at 10000 training points were aborted. The Matlab implementation of Algorithm 1 was able to crank about 15000 examples with the RBF kernel, whereas the C++ implementation succeeded to learn 28000 examples, before running out of memory for storing the kernel matrix and the auxiliary data structures (at about 3GB). The relative performance gain of the C++ implementation using gaxpy-updates against the "best Matlab competitor" and against the dgemm-updates is shown in Figure 6 (linear kernel, C&P algorithm) and Figure 7 (RBF kernel, Algorithm 1). Major performance improvement in comparison to Matlab and the naive C++ implementations can be observed, especially visible on larger training set sizes.

6. Applications

As it was mentioned in the introduction, various applications of incremental SVM learning can be foreseen. Two exemplare applications are presented in this session in order to illustrate some potential application domains.

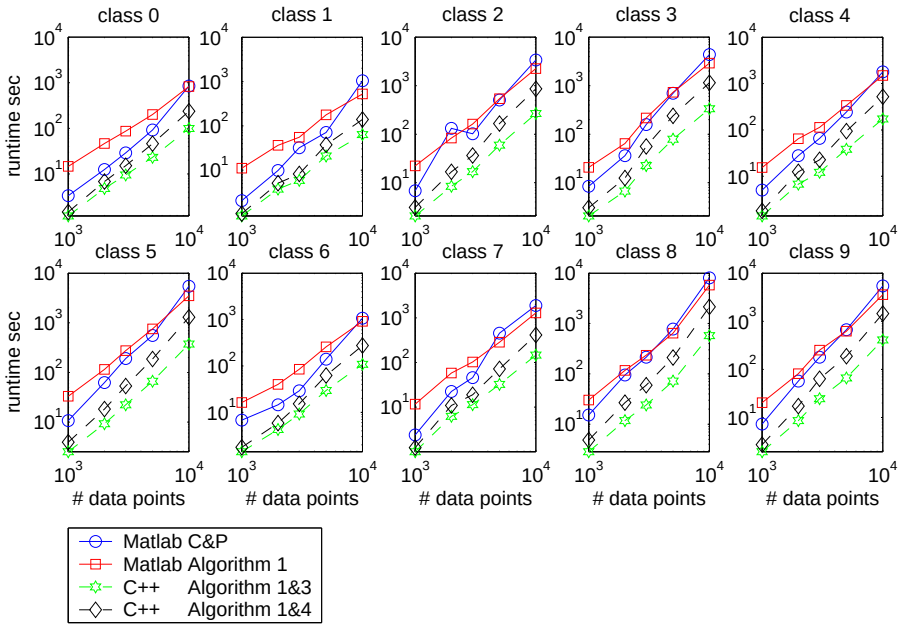


Figure 4: Scaling factor plots of incremental SVM algorithms with the linear kernel.

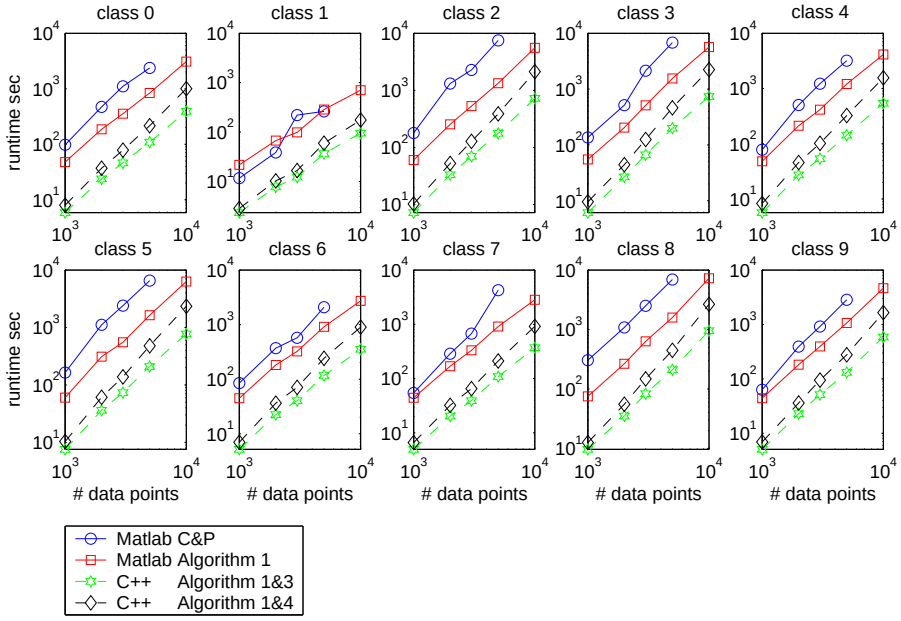


Figure 5: Scaling factor plots of incremental SVM algorithms with the RBF kernel ($\sigma = 30$).

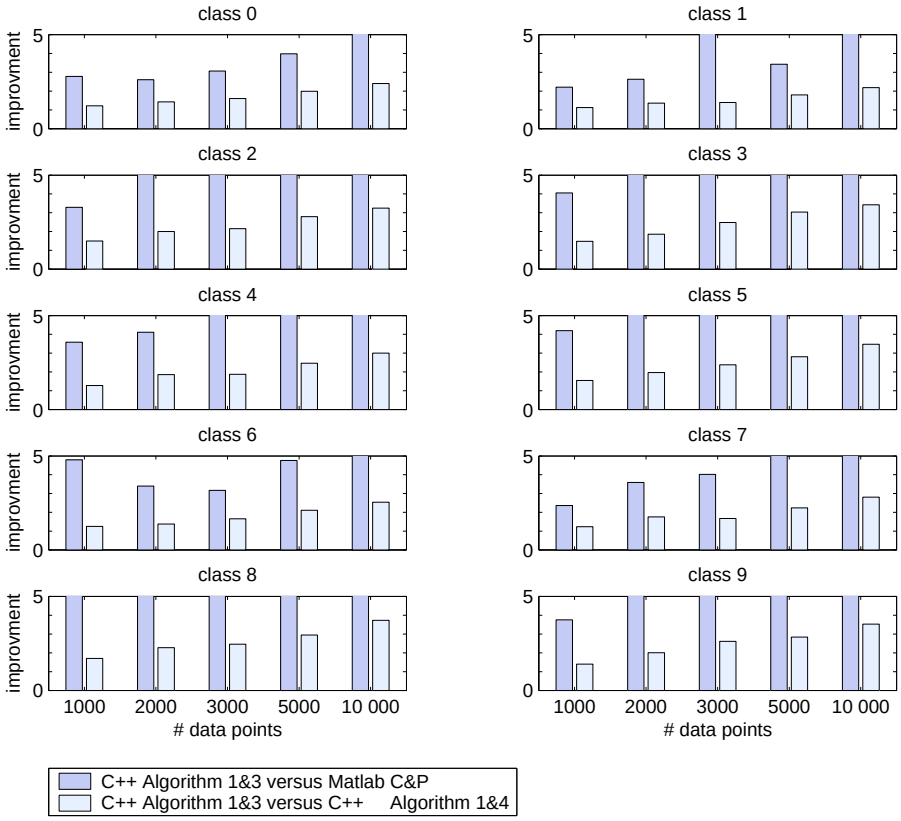


Figure 6: Runtime improvement, linear kernel.

6.1 Learning with Limited Resources

To make SVM learning applicable to very large data sets, a classifier has to be constrained to have a limited number of objects in memory. This is, in principle, exactly what an online classifier with fixed window size M does. Upon arrival of a new example, a least relevant example needs to be removed before a new example can be incorporated. A reasonable criterion for relevance is the value of the weight.

USPS experiment: learning with limited resources. As a proof of concept for learning with limited resources we train an SVM on the USPS data set under the limitations on the number of points that can be seen at a time. The USPS data set contains 7291 training and 2007 images of handwritten digits, size 16×16 (Vapnik, 1998). On this 10-class data set 10 support vector classifiers with a RBF kernel, $\sigma^2 = 0.3 \cdot 256$ and $C = 100$, were trained.[11] During the evaluation of a new object, it is assigned to the class corresponding to the classifier with the largest output. The total classification error on the test set for different window sizes M is shown in Figure 8.

One can see that the classification accuracy deteriorates marginally (by about 10%) until the working size of 150, which is about 2% of the data. True, by discarding "irrelevant" examples, one removes potential support vectors that cannot be recovered at a later stage. Therefore one can

11. The best model parameters as reported in (Vapnik, 1998) were used.

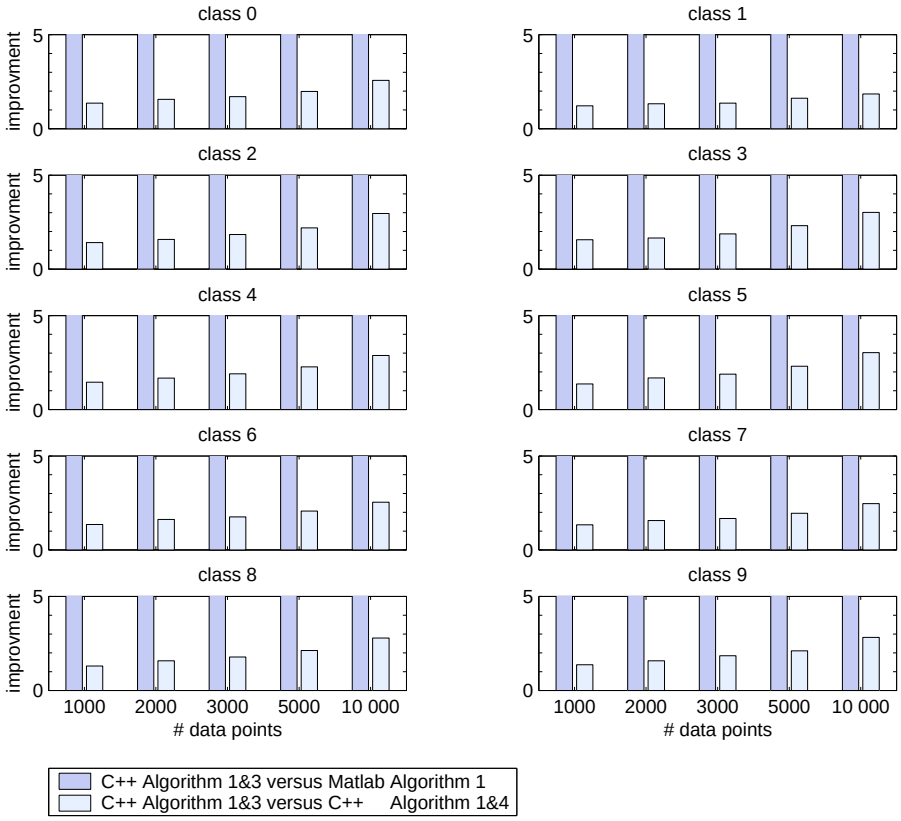


Figure 7: Runtime improvement, RBF kernel.

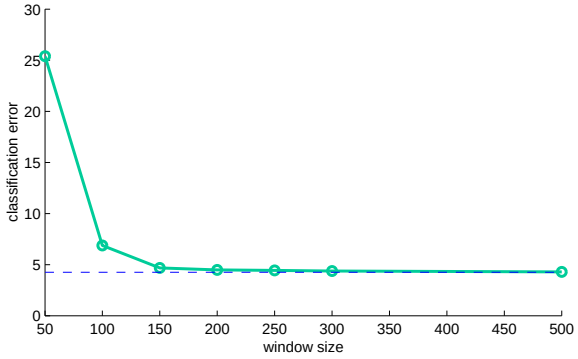


Figure 8: Test classification errors on the USPS data set, using a support vector classifier (RBF kernel, $\sigma^2 = 0.3 \cdot 256$) with a limited "window" of training examples.

expect that performance of a limited memory classifier would be worse than that of an unrestricted classifier. It is also obvious that no more points than the number of support vectors are eventually needed, although the latter number is not known in advance. The average number of support vectors

per each unrestricted 2-class classifier in this experiment is 274. Therefore the results above can be interpreted as reducing the storage requirement by 46% from the minimal at the cost of 10% increase of classification problem.

Notice that the proposed strategy differs from the caching strategy, typical for many SVM^{light}-like algorithms (Joachims, 1999; Laskov, 2002; Collobert and Bengio, 2001), in which kernel products are re-computed if the examples are found missing in the fixed-size cache and the accuracy of the classifier is not sacrificed. Our approach constitutes a trade-off between accuracy and computational load because kernel products never need to be re-computed. It should be noted, however, that computational cost of re-computing the kernels can be very significant, especially for the problems with complicated kernels such as string matching or convolution kernels.

6.2 Active Learning

Another promising application of incremental SVM is active learning. In this scenario, instead of having all data labelled beforehand, an algorithm "actively" chooses examples for which labels must be assigned by a user. Active learning can be extremely successful, if not indispensable, when labelling is expensive, e.g. in computer security or in drug discovery applications.

A very powerful active learning algorithm using SVM was proposed by Warmuth et al. (2003). Assume that the goal of learning is to identify "positive" examples in a data set. The meaning of positivity can vary across applications; for example, it can be binding properties of molecules in drug discovery applications, or hacker attacks in security applications. Selection of a next point to be labelled is carried out in the algorithm of Warmuth et al. (2003) using two heuristics that can be derived from an SVM classifier trained on points with known labels. The "largest positive" heuristic selects the point that has the largest classification score among all examples still unlabeled. The "near boundary" heuristic selects the point whose classification score has the smallest absolute value. Although the semantics of these two heuristics differ – in one case we trying to explore the space of positive examples as fast as possible, whereas in the other case the effort is focused on learning the boundary – in both cases the SVM has to be re-trained after each selection. In the original application of Warmuth et al. (2003) the data samples were relatively small, therefore one could afford re-training SVM from scratch after addition of new points. Obviously, a better way to proceed is by applying incremental learning as presented in this paper.

In the remaining part of this section experiments will be presented that prove the usefulness of active learning in the intrusion detection context. Since the observed data can contain thousands and even millions of examples it is clear that the problem can be addressed only using incremental learning. As a by-product of our experiments, it will be seen that active learning helps to uncover the structure of a learning problem revealed by the number of support vectors.

The underlying data for the experiments is taken from the KDD Cup 1999 data set.[12] As a training set, 1000 examples are randomly drawn with an attack rate of 10 percent. The incremental SVM was run with the linear kernel and the RBF kernel with $\sigma = 30$. An independent set of the same length with the same attack distribution is used for testing. The results are not averaged over multiple repetitions in order not to "disturb" the semantics of different phases of active learning as can be seen from the ROC curves. However, similar behavior was observed over multiple experiments.

KDD Cup experiment: active learning. Consider the following learning scenario. Assume that we can run an anomaly detection tool over our data set which ranks all the points according to

12. http://www-cse.ucsd.edu/users/elkan/clresults.html

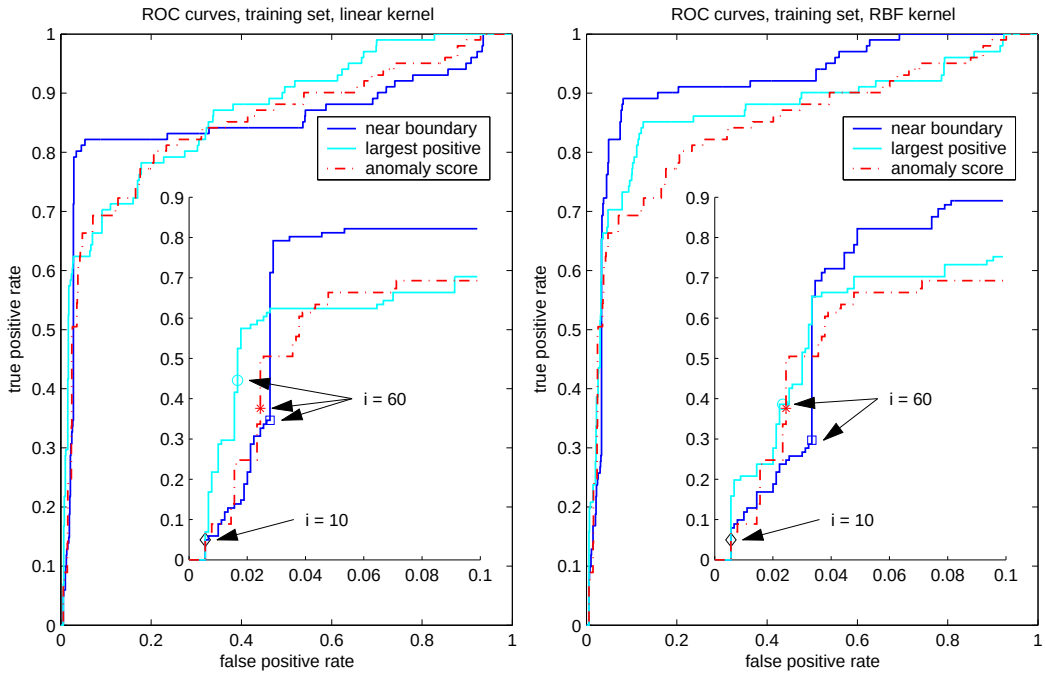


Figure 9: linear kernel, n=10 and m=50

their degree of anomaly. No labels are needed for this; however, if we know them, we can evaluate anomaly detection by a ROC curve. It is now the goal to use active learning to see if a ROC curve of anomaly detection can be improved.

We take the first n examples with the highest anomaly scores and train a (batch) SVM to obtain an initial model. After that we turn to active learning and learn the next m examples. We are now ready to classify the remaining examples by the trained SVM. The question arises: after spending manual effort to label $n+m$ examples, can we classify the remaining examples better than anomaly detection?

In order to address this question, an accuracy measure must be defined for our learning scenario. This can be done using the fact that only the ranking of examples according to their scores – and not the score values themselves – matters for the computation of a ROC curve (Cortes and Mohri, 2004). The ranking in our experiment can be defined as follows: the first n examples are ranked according to their anomaly scores, the next m examples are ranked according to their order of inclusion during the active learning phase, and the remaining examples are ranked according to their classification scores.

The ROC curves for active learning with the two heuristics and for anomaly detection are shown in Figure 9. One can easily see a different behavior exhibited by the two active learning rules. The "largest positive" rule attains the highest true positive rate during the active learning phase, but does not perform significantly better than anomaly detection during the classification phase ($i > 60$). On the contrary, the "near boundary" rule is close or worse than anomaly detection during the learning phase but exhibit a sharp increase of the true positive rate after moving to classification mode. Its accuracy then remains consistently better than anomaly detection for a considerable false positive interval (until FP = 0.3 for the linear kernel and until FP = 0.9 for the RBF kernel). Similar behavior

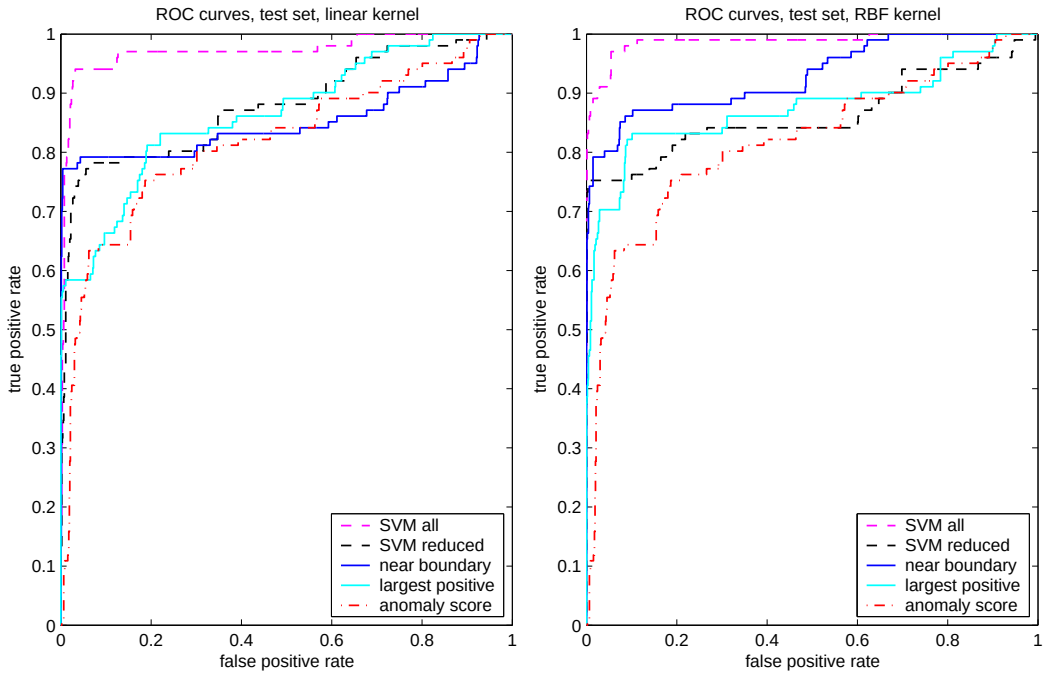


Figure 10: linear kernel, n=10 and m=50

of the two heuristics in the active learning phase was also observed by Warmuth et al. (2003). Yet the "near boundary" heuristic is obviously more suitable for classification, since it explores the boundary region and not merely the region of positive examples.

Another interesting insight can be gained by investigating the behavior of active learning on test data. In this case, supervised learning can also be drawn into comparison. In particular, we consider a full SVM (using a training set size of 1000 examples as opposed to only 60 examples used in the active learning) and a reduced SVM. The latter is obtained from a full SVM by finding a hyperplane closest to a full SVM hyperplane subject to 1-norm regularization over expansion coefficients (cf. Schölkopf et al. (1999)). The regularization constant is chosen such that a reduced SVM has approximately the same number of support vectors as the solution obtain by active learning (in our case the value $\lambda = 2.5$ resulted in about 30 support vectors). Thus one can compare active learning with supervised learning given equal complexity of solutions.

The ROC curves of active learning, supervised learning and anomaly detection on test data are shown in Figure 10. It can be observed that the "near-boundary" heuristic of active learning attains a solution which is at least as good (for FP ≤ 0.4) as a reduced SVM for the linear kernel and significantly better for the RBF kernel. This shows that active learning does a very good job at discovering the necessary structure of a solution – it picks a better representation within a desired complexity since it is using a learning-related criterion to select an interesting representation instead of a merely geometric one.

7. Discussion and Conclusions

Online learning algorithms have proved to be essential when dealing with (a) very large (see e.g. LeCun et al. (1998); Bordes et al. (2005); Tsang et al. (2005)) or (b) non-stationary data (see e.g. Robbins and Munro (1951); Murata (1992); Murata et al. (1997, 2002)). While classical neural networks (e.g. LeCun et al. (1998); Saad (1998)) have a well established online learning toolbox for optimization, incremental learning techniques for Support Vector Machines have been only recently developed (Cauwenberghs and Poggio, 2001; Tax and Laskov, 2003; Martin, 2002; Ma et al., 2003; Ma and Perkins, 2003).

The current paper contributes two-fold to the field of incremental SVM learning. The convergence analysis of the algorithm has been performed showing that immediate cycling of the algorithm is impossible provided a kernel matrix is positive semi-definite. Furthermore, we propose a better scheme for organization of memory and arithmetic operations in exact incremental SVM using the gaxpy-type updates of the sensitivity vector. As it is demonstrated by our experiments, the new design results in major constant improvement in the running time of the algorithm.

The achieved performance gains open wide possibilities for application of incremental SVM to various practical problems. We have presented exemplary applications to two possible scenarios: learning with limited resources and active learning. Potential applications of incremental SVM learning include, among others, drug discovery, intrusion detection, network surveillance, monitoring of non-stationary time series etc. Our implementation is available free of charge for academic use at http://www.mind-ids.org/Software.

It is interesting to compare exact incremental learning to recently proposed alternative approaches to online learning. The recent work of Bordes et al. (2005) presents an online algorithm for L1 SVM, in which a very close approximation of the exact solution is built online before the last gap is bridged in the REPROCESS phase in an offline fashion. This algorithm has been shown to scale well to several hundred thousand examples, however its online solution is not as accurate as the exact solution. It has been observed (cf. Fig. 9 in Bordes et al. (2005)) that the REPROCESS phase may result in major improvement of the test error and may come at a high price in comparison with the online phase, depending on a data set. Another recent algorithm, the Core Vector Machine of Tsang et al. (2005), is based on the L2 formulation of an SVM and has be shown to scale to several million of examples. The idea of this algorithm is to approximate a solution to an L2 SVM by a solution to the two-class Maximal Enclosing Ball problem, for which several efficient online algorithms are known. While scalability results of CVM are very impressive, the approximation of the exact solution can likewise in higher test errors.

The major limilation of the exact incremental learning is its memory requirement, since the set of support vectors must be retained in memory during the entire learning. Due to this limitation, the algorithm is unlikely to be scalable beyond tens of thousands examples; however, for data sizes within this limit it offers an advantage of immediate availablility of the exact solution (crucial in e.g learning of non-stationary problems) and reversibility.

Future work will include further investigation of properties of incremental SVM such as numerical stability and their utility for tracking the values of generalization bounds. A relationship with parametric optimization needs to be further clarified. Extensions to advance learning modes, such as learning for structured domains and semi-supervised learning, are being considered.

Acknowledgments

The authors are grateful to David Tax and Christian Zillober for fruitful discussions on various topics of mathematical optimization that contributed to the development of main ideas of this paper. René Gerstenberger provided valuable help in the profiling experiments. This work was partially supported by *Bundesministerium für Bildung und Forschung* under the project MIND (FKZ 01-SC40A), by *Deutsche Forschungsgemeinschaft* under the project MU 987/2-1, and by the IST Programme of the European Community under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

B.. Blankertz, G. Dornhege, C. Schäfer, R. Krepki, J. Kohlmorgen, K.-R. Müller, V. Kunzmann, F. Losch, and G. Curio. BCI bit rates and error detection for fast-pace motor commands based on single-trial EEG analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11:127–131, 2003.

A. Bordes, S. Ertekin, J. Wesdon, and L. Bottou. Fast kernel classifiers for online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 409–415. MIT Press, 2001.

S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kaufmann, 2002. ISBN 1-55860-754-4.

C.-C. Chang and C.-J. Lin. Libsvm: Introduction and benchmarks. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2000.

R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *Proc. NIPS'2003*, 2004.

E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. *Applications of Data Mining in Computer Security*, chapter A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data. Kluwer, 2002.

G. H. Golub and C. F. van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, London, 3rd edition, 1996.

T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. In T. G. Diettrich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Inf. Proc. Systems (NIPS 01)*, pages 785–792, 2001.

F. Klee and G. J. Minty. How good is the simplex algorithm? In O. Sisha, editor, *Inequalities III*, pages 159–175. Academic Press, 1972.

P. Laskov. Feasible direction decomposition algorithms for training support vector machines. *Machine Learning*, 46:315–349, 2002.

P. Laskov, C. Schäfer, and I. Kotenko. Intrusion detection in unlabeled data with quarter-sphere support vector machines. In *Proc. DIMVA*, pages 71–82, 2004.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In G. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524, pages 9–53, Heidelberg, New York, 1998. Springer LNCS.

N. Littlestone, P. M. Long, and M. K. Warmuth. On-line learning of linear functions. Technical Report CRL-91-29, University of California at Santa Cruz, October 1991.

J. Ma and S. Perkins. Time-series novelty detection using one-class Support Vector Machines. In *IJCNN*, 2003. to appear.

J. Ma, J. Theiler, and S. Perkins. Accurate online support vector regression. http://nis-www.lanl.gov/˜jt/Papers/aosvr.pdf, 2003.

M. Martin. On-line Support Vector Machines for function approximation. Technical report, Universitat Politècnica de Catalunya, Departament de Llengatges i Sistemes Informàtics, 2002.

N. Murata. *A statistical study on the asymptotic theory of learning*. PhD thesis, University of Tokyo (In Japanese), 1992.

N. Murata, M. Kawanabe, A. Ziehe, K.-R. Müller, and S.-I. Amari. On-line learning in changing environments with applications in supervised and unsupervised learning. *Neural Networks*, 15 (4-6):743–760, 2002.

N. Murata, K.-R. Müller, A. Ziehe, and S. i. Amari. Adaptive on-line learning in changing environments. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 599. The MIT Press, 1997.

G. Orr and K.-R. Müller, editors. *Neural Networks: Tricks of the Trade*, volume 1524. Springer LNCS, 1998.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

L. Ralaivola and F. d'Alché Buc. Incremental support vector machine learning: A local approach. *Lecture Notes in Computer Science*, 2130:322–329, 2001.

H. Robbins and S. Munro. A stochastic approximation method. *Ann. Math. Stat.*, 22:400–407, 1951.

S. Rüping. Incremental learning with support vector machines. Technical Report TR-18, Universität Dortmund, SFB475, 2002.

D. Saad, editor. *On-line learning in neural networks*. Cambridge University Press, 1998.

B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5): 1000–1017, September 1999.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

N. A. Syed, H. Liu, and K. K. Sung. Incremental learning with support vector machines. In *SVM workshop, IJCAI*, 1999.

D. Tax and R. Duin. Data domain description by support vectors. In M. Verleysen, editor, *Proc. ESANN*, pages 251–256, Brussels, 1999. D. Facto Press.

D. M. J. Tax and P. Laskov. Online SVM learning: from classification to data description and back. In C. et al. Molina, editor, *Proc. NNSP*, pages 499–508, 2003.

I. Tsang, J. Kwok, and P.-M. Cheung. Core Vector Machines: fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

M. K. Warmuth, J. Liao, G. Rätsch, M. Mathieson, S. Putta, and C. Lemmem. Support Vector Machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, 43(2):667–673, 2003.

Journal of Machine Learning Research 7 (2006) 1937-1962 Submitted 6/05; Revised 2/06; Published 10/06

A Simulation-Based Algorithm for Ergodic Control of Markov Chains Conditioned on Rare Events

Shalabh Bhatnagar SHALABH@CSA.IISC.ERNET.IN
Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560 012, India

Vivek S. Borkar BORKAR@TIFR.RES.IN
School of Technology and Computer Science
Tata Institute of Fundamental Research, Homi Bhabha Road
Mumbai 400 005, India

Madhukar Akarapu MADHUKAR.AKARAPU@ORACLE.COM
Oracle India Pvt Ltd.
Bangalore - 560 029, India

Editor: Shie Mannor

Abstract

We study the problem of long-run average cost control of Markov chains conditioned on a rare event. In a related recent work, a simulation based algorithm for estimating performance measures associated with a Markov chain conditioned on a rare event has been developed. We extend ideas from this work and develop an adaptive algorithm for obtaining, online, optimal control policies conditioned on a rare event. Our algorithm uses three timescales or step-size schedules. On the slowest timescale, a gradient search algorithm for policy updates that is based on one-simulation simultaneous perturbation stochastic approximation (SPSA) type estimates is used. Deterministic perturbation sequences obtained from appropriate normalized Hadamard matrices are used here. The fast timescale recursions compute the conditional transition probabilities of an associated chain by obtaining solutions to the multiplicative Poisson equation (for a given policy estimate). Further, the risk parameter associated with the value function for a given policy estimate is updated on a timescale that lies in between the two scales above. We briefly sketch the convergence analysis of our algorithm and present a numerical application in the setting of routing multiple flows in communication networks.

Keywords: Markov decision processes, optimal control conditioned on a rare event, simulation based algorithms, SPSA with deterministic perturbations, reinforcement learning

1. Introduction

Markov decision processes (MDPs) (Bertsekas, 2001; Puterman, 1994), form a general framework for studying problems of control of stochastic dynamic systems (SDS). Many times, one encounters situations involving control of SDS conditioned on a rare event of asymptotically zero probability. This could be, for example, a problem of damage control when faced with a catastrophic event. For instance, in the setting of a large communication network such as the internet, one may be interested in obtaining optimal flow and congestion control or routing strategies in a subnetwork given that an extremal event such as a link failure has occurred in another remote subnetwork. Our

objective in this paper is to consider a problem of this nature wherein a rare event is specifically defined to be the time average of a function of the MDP and its associated control-valued process exceeding a threshold that is larger than its mean. We consider the infinite horizon long-run average cost criterion for our problem and devise an algorithm based on policy iteration for the same.

Research on developing simulation based methods for control of SDS has gathered momentum in recent times. These largely go under the names of neuro-dynamic programming (NDP) or reinforcement learning (RL), (see, for example, Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998), and are applicable in the case of systems for which model information is not known or computationally forbiddingly expensive, but output data obtained either through a real system or a simulated one is available. Our problem does not share this last feature, but we do borrow certain algorithmic paradigms from this literature. Before we proceed further, we first review some representative recent work along these lines. In Baxter and Bartlett (2001), an algorithm for long-run average cost MDPs is presented. The average cost gradient is approximated using that associated with a corresponding infinite horizon discounted cost MDP problem. The variance of the estimates however increases rapidly as the discount factor is brought closer to one. In Baxter et al. (2001), certain variants based on the algorithm in Baxter and Bartlett (2001) are presented and applications on some experimental settings shown.

In Cao and Guo (2004), a perturbation analysis (PA) type approach is used to obtain the performance gradient based on sample path analysis. In Cao (1998), a PA-based method is proposed for solving long-run average cost MDPs. This requires keeping track of the regeneration epochs of the underlying process for any policy and aggregating data over these. The above epochs can however be very infrequent in most real life systems. In Marbach and Tsitsiklis (2001), the average cost gradient is computed by assuming that sample path gradients of performance and transition probabilities are known in functional form. Amongst other RL-based approaches, the temporal difference (TD) and Q-learning, (see Sutton and Barto, 1998; Watkins and Dayan, 1992, respectively), have been popular in recent times. These are based on value function approximations. A parallel development is that of actor-critic algorithms based on the classical policy iteration algorithm in dynamic programming. Note that the classical policy iteration algorithm proceeds via two nested loops—an outer loop in which the policy improvement step is performed and an inner loop in which the policy evaluation step for the policy prescribed by the outer loop is conducted. The respective operations in the two loops are performed one-after-the-other in a cyclic manner. The inner loop can in principle take a long time to converge, making the overall procedure slow in practice. In Konda and Borkar (1999), certain simulation-based algorithms that use multi-timescale stochastic approximation are proposed. The idea is to use coupled stochastic recursions driven by different step-size schedules or timescales. The recursion corresponding to policy evaluation is run on the faster timescale while that corresponding to policy improvement is run on the slower one. Thus while both recursions proceed simultaneously, the algorithm converges to the optimal policy. The algorithms of Konda and Borkar (1999) (as with those described in the previous paragraph) are for finite state and finite action MDPs, under both the discounted and long-run average cost criteria. A variant of the above algorithms for the case of finite state but compact (non-discrete) action sets, in the setting of infinite horizon discounted cost MDPs is presented by Bhatnagar and Kumar (2004), and performs gradient search in the space of stationary deterministic policies using a simultaneous perturbation stochastic approximation (SPSA) gradient estimate.

Standard SPSA (Spall, 1992) uses two simulations for estimating the performance/cost gradient regardless of the dimension N of the parameter vector, unlike Kiefer-Wolfowitz (K-W) based esti-

mates that require $(N+1)$ simulations for the same. This it done by randomly perturbing all parameter components at each update epoch. The original SPSA algorithm of Spall (1992) is, however, a one-timescale Robbins-Monro variant for parameter optimization and is not directly applicable when the cost to be optimized is for instance the long-run average of a running cost function, viz., the objective function for a given parameter value is derived only after viewing the entire sample path / trajectory of the system for that parameter value. Perturbation analysis (PA) schemes (see Chong and Ramadge, 1994; Ho and Cao, 1991), that were proposed for problems such as these use largely one simulation, however, they require certain constraining regularity conditions on the system dynamics and cost functions in order to allow for an interchange between the 'gradient' and 'expectation' operators. Moreover, many of these schemes update parameters only at certain regeneration epochs of the underlying process, making them slow in practice. In Bhatnagar and Borkar (1997, 1998), certain two-timescale stochastic approximation algorithms were introduced as alternatives to PA type schemes. These do not require constraining regularity conditions like PA, while they also update parameters at certain deterministic epochs. The key in the above algorithms is the use of two-timescale stochastic approximation, whereby on the faster timescale, data corresponding to a given parameter update is aggregated and on the slower timescale, the parameter is updated. These algorithms, however, use K-W estimates. In Bhatnagar et al. (2001), variants that use SPSA estimates were proposed and were found to show significantly improved performance. In Spall (1997), a one-simulation (one-timescale) variant of the original SPSA algorithm was proposed, which however does not show good performance because of the presence of an 'additional' bias term in its gradient estimate whose contribution to overall bias tends to be high. In Bhatnagar et al. (2003), it was observed in a similar setting by Bhatnagar and Borkar (1997), Bhatnagar and Borkar (1998) and Bhatnagar et al. (2001), respectively, that the use of deterministic perturbation sequences (instead of randomized) derived using normalized Hadamard matrices significantly alleviates this problem in the case of one-simulation SPSA with the latter subsequently showing good performance. It was shown that perturbation sequences derived using normalized Hadamard matrices satisfy the desired properties on such sequences that result in all bias terms getting cancelled at regular intervals. Further, the space of perturbations derived as above has a cardinality of $2^{\log_2(N+1)}$ as against 2^N when randomized perturbations are used (the perturbation vectors in both spaces being $\{\pm 1\}^N$-valued). To sum up, the use of normalized Hadamard matrix based perturbations in the setting as described above has the inherent advantage that one may use a fast one-simulation SPSA based algorithm that updates all parameter components at each update epoch (the epochs themselves being deterministically spaced). In particular, the algorithms of Bhatnagar et al. (2003) update the parameter once every L epochs for a given, arbitrarily chosen integer L while working with a more general class of systems than what the PA based methods allow.

The works cited above represent some recent developments in the general area of simulation based optimization and control of SDS. We now review some of the work that is more directly related to the problem we study in this paper. In Borkar et al. (2004), a simulation-based algorithm for estimating performance measures of a Markov chain conditioned on a rare event of zero probability has been developed. This is based on the result that the transition probabilities of the Markov chain conditioned on a rare event as above are the same as those of another irreducible chain on the same state space whose transition probabilities are absolutely continuous w.r.t. those of the former chain. The calculation of these calls for the solution of an associated *multiplicative Poisson equation*, an object familiar from risk-sensitive control and large deviations theory (see Kontoyiannis and Meyn, 2003; Balaji and Meyn, 2000). The simulation based algorithm of Borkar et al. (2004) recursively

obtains the solution to this multiplicative Poisson equation and uses the same to learn, online, the new transition probabilities. In Ahamed et al. (2006), a reinforcement learning based importance sampling scheme for estimating expectations associated with rare events has also been proposed.

A related paper by Rubinstein (1997), in which a simulation based technique for optimizing certain performance measures in discrete event systems conditioned on rare events is presented. The problem there is formulated as a constrained optimization problem with an importance sampling estimate in the objective function that is obtained by assuming the underlying processes to be regenerative. The constraint there corresponds to the occurrence of the given rare event. The above problem is then solved as a two-stage stochastic programming problem. Our work is fundamentally different from that of Rubinstein (1997) in many ways. First, we consider the problem of obtaining an optimal control policy conditioned on a rare event and not just one of optimizing certain performance metrics within a parameterized class as with Rubinstein (1997). Next, even though we assume that our underlying process for any given stationary policy is ergodic Markov and hence regenerative, we do not use the regenerative structure per se in obtaining estimates of performance as Rubinstein (1997) does. For the latter, one needs in particular to keep track of regeneration epochs of the underlying process that can be very infrequent in the case of most systems. Finally, we use a stochastic approximation based recursive procedure that incorporates reinforcement learning type estimates, unlike (as already mentioned) Rubinstein (1997) who formulates the problem as a stochastic program.

Our work can be viewed as an extension of Borkar et al. (2004) that addresses the important problem of optimal control of a Markov chain conditioned on a rare event. In our framework, the results of Borkar et al. (2004) correspond to policy evaluation for a *fixed* stationary deterministic policy. We develop and use a simulation-based algorithm to find the *optimal* randomized policy 'on top of' the algorithm of Borkar et al. (2004). Our algorithm uses three timescales or step-size schedules and iterates in the space of stationary randomized policies. The policy itself, however, is updated on the slowest timescale. The value function updates for finding the solution to the multiplicative Poisson equation for a given policy, based on which the transition probabilities of an associated chain are obtained, are performed on the fastest timescale. The risk parameter associated with the multiplicative Poisson equation is updated on a timescale that is faster than the one on which policy is updated, but slower than that on which value function is updated. Finally, there is another recursion that is used for averaging the cost function with the latter average used in the policy update step. This proceeds on the fastest scale as well (same as the one on which the value function is updated). We show in the analysis that the difference in timescales of the various recursions results in the desired algorithmic behavior. For policy updates, we use a one-simulation SPSA based recursion with normalized Hadamard matrices (Bhatnagar et al., 2003). Finally, we present numerical experiments using our algorithm in the setting of routing multiple flows in communication networks conditioned on a rare event. We observe that our algorithm exhibits good performance in this setting. It must be noted here that adaptive importance sampling (IS) schemes require storage of transition probabilities and our algorithm is no different in this regard. Thus it may not be applicable (as is also the case with other IS methods) in scenarios that involve very large state spaces for which storage of such information is not possible. Nevertheless, feature based methods as in RL may still be applied for ease of computation in the case of problems with state and action spaces that are moderately large but for which storage of vectors of the size of state space is not a major concern. Further, in many cases such as queuing networks, the transition probabilities are easy to compute and transitions easy to simulate using simple local dynamic laws. In such scenarios, storage of

transition probability matrices may also not be a major concern as these are known to be highly sparse.

The rest of the paper is organized as follows: Section 2 presents the problem formulation and gives the basic results. Section 3 presents the simulation-based algorithm. Its convergence analysis is also briefly sketched here. The numerical results are presented in Section 4. Finally, Section 5 presents the concluding remarks.

2. Problem Formulation and Basic Results

Consider a Markov decision process (MDP) $\{X_n,\ n \geq 0\}$ on a finite state space $S = \{1,2,\ldots,s\}$. For $X_n = i$, $i \in S$, let $A(i)$ be the set of feasible controls or actions. We assume $A(i)$ has the form $A(i) = \{a_i^1, a_i^2, \ldots, a_i^{N_i}\}$. Let $A = \cup_{i\in S} A(i)$ denote the action space (which is also finite). Let $\{Z_n,\ n \geq 0\}$ denote the associated control-valued sequence such that $Z_n \in A(X_n)\ \forall n$. Suppose $p(i,j,a)$ denotes the transition probability from state i to state j under action $a \in A(i)$. Then the evolution of $\{X_n\}$ is governed by

$$Pr(X_{n+1} = j \mid X_n = i, Z_n = a, X_{n-1} = i_{n-1}, Z_{n-1} = a_{n-1}, \ldots, X_0 = i_0, Z_0 = a_0) = p(i,j,a),$$

for any $i_0,\ldots,i_{n-1}$, i, j, $a_0,\ldots,a_{n-1}$, a, in appropriate sets.

A sequence of functions $\pi = \{\mu_1, \mu_2, \ldots\}$ with each $\mu_n : S \to A$, $n \geq 1$, is said to be an admissible policy if $\mu_n(i) \in A(i), \forall i \in S$. This corresponds to the control choice $Z_n = \mu_n(X_n)\ \forall n$. An admissible policy $\pi = \{\mu_1, \mu_2, \ldots\}$ with each $\mu_n = \mu$, $n \geq 1$, is said to be a stationary deterministic policy (SDP). By a common abuse of notation, we simply refer to μ itself as the SDP. By a randomized policy (RP) ψ, we mean a sequence $\psi = \{\phi_1, \phi_2, \ldots\}$ with each $\phi_n : S \to \mathcal{P}(A)$, $n \geq 1$. Here $\mathcal{P}(A)$ is the set of all probability vectors on A such that for each $i \in S$, $n \geq 1$, $\phi_n(i) \in \mathcal{P}(A(i))$, with $\mathcal{P}(A(i))$ being the set of all probability vectors on $A(i)$. A stationary randomized policy (SRP) is an RP ψ for which $\phi_n(i) = \phi\ \forall n \geq 1$. By an abuse of notation, we refer to ϕ itself as the SRP. The $a-$th component of $\phi(i)$, $\phi(i)(a)$ is the probability of choosing action a when in state i. Thus this corresponds to picking Z_n with probability distribution $\phi(X_n)$ at time n, independent of all other random variables realized till n. We make

Assumption (A) Under any SDP μ, the process $\{X_n\}$ forms an irreducible Markov chain.

Let $E_\mu[\cdot]$ denote the expectation w.r.t. the stationary distribution of $\{X_n\}$ under SDP μ. Let $g : S \times A \to \mathcal{R}$ be a given function such that $E_\mu[g(X_n, \mu(X_n))] < \alpha < \infty$ for a given constant α, for every SDP μ. The rare event that we consider corresponds to

$$\lim_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} g(X_m, \mu(X_m)) \geq \alpha.$$

The choice of the function $g(\cdot,\cdot)$ and α will be, in practice, dictated by the application. For example, in reliability, one may want to look at the stationary probability of crossing a very large threshold, say, N. Then $g(X_m, \mu(X_m))$ can be chosen to be $I\{X_m \geq N\}$, where $I\{\cdot\}$ is the indicator function and α could be a convenient upper bound on the stationary expectation.

Let $h : S \times A \times S \to \mathcal{R}$ denote the cost function that we assume is bounded. For any SDP μ, let for any (initial state) $X_0 \in S$,

$$J(\mu) = \lim_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} h(X_m, \mu(X_m), X_{m+1})$$

be the long-run average cost. Let D be the set of all possible stationary deterministic policies. The aim is to find

$$\mu^* = \arg\min_{\mu \in D} J(\mu),$$

conditioned on the rare event $\lim_{n\to\infty} \frac{1}{n}\sum_{m=0}^{n-1} g(X_m, \mu(X_m)) \geq \alpha$, $\forall \mu \in D$. Let $p^{\mu,*}(i,j) = \lim_{n\to\infty} P(X_1 = j \mid X_0 = i, Z_0 = \mu(i), \frac{1}{n}\sum_{m=0}^{n-1} g(X_m, \mu(X_m)) \geq \alpha)$ denote the transition probabilities under SDP μ conditioned on a rare event (as defined above). We now present the basic results for a given SDP μ. These have been directly adapted from Borkar et al. (2004) for a fixed SDP and are stated here for the sake of completeness. Some of these results are also available in the context of risk sensitive control of Markov chains (see, for instance, Balaji and Meyn, 2000; Hernández-Hernández and Marcus, 1996; Kontoyiannis and Meyn, 2003). We briefly explain the risk sensitive control problem in order to put things in perspective. Suppose (that instead of the original) the aim is simply to find an SDP μ that minimizes $J_\zeta(\mu)$ defined by

$$J_\zeta(\mu) = \lim_{n\to\infty} \frac{1}{n} \ln\left(E\left[\exp(\sum_{m=0}^{n-1} \zeta g(X_m, \mu(X_m))) \right] \right),$$

where ζ denotes the risk parameter. Note above that the cost considered in this setting is given by the function g and not h. The cases $\zeta > 0$ and $\zeta < 0$ correspond to the risk-averse and risk-preferring cases, respectively. For a given μ, $J_\zeta(\mu)$ and ρ_ζ^μ are a solution (see Balaji and Meyn, 2000; Hernández-Hernández and Marcus, 1996) to the multiplicative Poisson equation: For $i \in S$,

$$V_\zeta^\mu(i) = \frac{\exp(\zeta g(i,\mu(i)))}{\rho_\zeta^\mu} \sum_j p(i,j,\mu(i)) V_\zeta^\mu(j), \;\; i \in S, \tag{1}$$

where $V_\zeta^\mu(\cdot)$ is a bounded function (that is unique up to a multiplicative constant). It turns out that ρ_ζ^μ corresponds to $\exp(J_\zeta(\mu))$ or that $J_\zeta(\mu) = \ln \rho_\zeta^\mu$. Note that solution of this equation is an eigenvalue problem for the positive matrix $[[\exp(\zeta g(i,\mu(i)))p(i,j,\mu(i))]]_{i,j\in S}$ with V_ζ^μ and ρ_ζ^μ corresponding to its Perron-Frobenius eigenvector and eigenvalue respectively.

For the problem considered in this paper, as shown by Borkar et al. (2004), the multiplicative Poisson equation also arises via the conditional transition probabilities $p^{\mu,*}(i,j)$ (for given SDP μ), see (2) below. In fact, for any given $i \in S$, upon summing over all $j \in S$ on both sides of (2), one obtains the multiplicative Poisson Equation (1). For any SDP μ and risk parameter ζ, $J_\zeta(\mu) = \ln \rho_\zeta^\mu$ corresponds to the infinite horizon risk-sensitive cost. As in Borkar et al. (2004), we fix the choice of $V_\zeta^\mu(\cdot)$ by setting $V_\zeta^\mu(i_0) = \rho_\zeta^\mu$ for a given $i_0 \in S$ in order to obtain unique $V_\zeta^\mu(i)$ $\forall i \in S$.

Theorem 1 (Borkar et al., 2004)

(a) The map $\zeta \to \rho_\zeta^\mu$ is convex for each SDP μ and there exists a unique $\zeta_*^\mu \stackrel{\triangle}{=} \arg\max_{\zeta\geq 0}(\zeta\alpha - \ln(\rho_\zeta^\mu))$ for any μ.

(b) $p^{\mu,*}(i,j)$, $i,j \in S$ is given by

$$p^{\mu,*}(i,j) = \frac{\exp(\zeta_*^\mu g(i,\mu(i)))p(i,j,\mu(i))V_*^\mu(j)}{\rho_*^\mu V_*^\mu(i)}. \tag{2}$$

(c) The regular conditional law of the MDP $\{X_m, m \geq 0\}$ under SDP μ, conditioned on the event $\{X_0 = x, \frac{1}{n}\sum_{k=0}^{n-1} g(X_k, \mu(X_k)) \geq \alpha\}$ converges to the law of a Markov chain starting at x with transition probabilities $p^{\mu,*}(\cdot,\cdot)$.

In the above, $\rho_*^\mu \stackrel{\triangle}{=} \rho_{\zeta_*^\mu}^\mu$ and $V_*^\mu \stackrel{\triangle}{=} V_{\zeta_*^\mu}^\mu$, respectively. It can be shown as in Lemma 2 of Borkar et al. (2004) using a generalization of Theorem 6.3 of Kontoyiannis and Meyn (2003) that as $n \to \infty$,

$$P_x(\frac{1}{n}\sum_{m=0}^{n-1} g(X_m, \mu(X_m)) \geq \alpha_n) \sim \frac{V_*^\mu(x)\exp(-n(\zeta_*^\mu\alpha - \ln(\rho_*^\mu)))\exp(k\zeta_*^\mu)}{\zeta_*^\mu\sqrt{2\pi n\lambda_*^\mu}}$$

where $\alpha_n = \alpha - \frac{k}{n}$ and $\lambda_*^\mu = \sqrt{\frac{\partial^2 \ln \rho_\zeta^\mu}{\partial \zeta^2}\,|_{\zeta=\zeta^*}}$. The result in Theorem 1(b) follows in a straightforward manner from the above. Thus the transition probabilities $p^{\mu,*}(\cdot,\cdot)$ depend on the risk parameter ζ_*^μ given in Theorem 1(a).

For a given $\zeta > 0$ and SDP μ, let $\{X_n^{\zeta,\mu}, n \geq 0\}$ represent a Markov chain on S with (suitably normalized) transition probabilities

$$p^{\mu,\zeta}(i,j) \stackrel{\triangle}{=} \frac{\exp(\zeta g(i,\mu(i)))p(i,j,\mu(i))V_\zeta^\mu(j)}{\rho_\zeta^\mu V_\zeta^\mu(i)}, \;\; i,j \in S.$$

In particular, we consider here the corresponding risk-averse case ($\zeta > 0$). The risk-preferring case ($\zeta < 0$) is easier to handle and is not considered in this paper. In view of Assumption (A), $\{X_n^{\zeta,\mu}\}$ is irreducible. Let $\eta_\zeta^\mu(\cdot)$ denote its unique stationary distribution. We now have the following lemma whose proof follows as in Proposition 4.9 of Kontoyiannis and Meyn (2003).

Lemma 1 $\frac{\partial \ln(\rho_\zeta^\mu)}{\partial \zeta} = \sum_{i \in S} \eta_\zeta^\mu(i) g(i,\mu(i))$.

In classical Markov decision theory, one is minimizing expectation and not conditional expectation of the ergodic cost and one can prove that it suffices to consider only SDPs. Such a result is not proved here, so it is our *choice* to restrict to these. Finally, in principle, the requirement that the rare event condition hold for all SDPs μ (see the problem definition above) is not strictly needed in order for the theory to go through. However, one expects this to be true in typical applications. In the next section, we present an adaptive algorithm for finding optimal μ and ζ by building on the basic results of Theorem 1 and Lemma 1.

3. The Adaptive Algorithm

Given an SRP $\phi : S \to \mathcal{P}(A)$, one can identify $\phi(i)$ with a parameter vector $\theta_i = (\theta_i^1, \ldots, \theta_i^{N_i-1})^T$, where $\theta_i^j \geq 0$ are the probabilities of picking actions a_i^j, $j = 1, \ldots, N_i - 1$. Thus $\sum_{j=1}^{N_i-1} \theta_i^j \leq 1$. Further, $\theta_i^{N_i}$ (the probability of selecting action $a_i^{N_i}$) is directly obtained from the above representation of $\phi(i)$ as $\theta_i^{N_i} = 1 - \sum_{j=1}^{N_i-1} \theta_i^j$. Let $\theta = (\theta_1 \ldots, \theta_s)^T = (\theta_1^1, \ldots, \theta_1^{N_1-1}, \theta_2^1, \ldots, \theta_2^{N_2-1}, \ldots, \theta_s^1, \ldots, \theta_s^{N_s-1})^T$. Let $p^{\theta_i}(i,j)$, $i,j \in S$, be defined by $p^{\theta_i}(i,j) = \theta_i^1 p(i,j,a_i^1) + \ldots + \theta_i^{N_i} p(i,j,a_i^{N_i})$. Thus $p^{\theta_i}(i,j)$

correspond to the transition probabilities of the resulting Markov chain under SRP ϕ. Suppose $g^{\theta_i}(i) = \theta_i^1 g(i,a_i^1) + \ldots + \theta_i^{N_i} g(i,a_i^{N_i})$ and $h^{\theta_i}(i,j) = \theta_i^1 h(i,a_i^1,j) + \ldots + \theta_i^{N_i} h(i,a_i^{N_i},j)$, respectively, denote the expected values of the function $g(\cdot,\cdot)$ and the single-stage cost $h(\cdot,\cdot,\cdot)$ under SRP ϕ. Define three step-size sequences $\{a(n)\}$, $\{b(n)\}$ and $\{c(n)\}$ satisfying

Assumption (B)

$$\sum_n a(n) = \sum_n b(n) = \sum_n c(n) = \infty, \ \sum_n (a(n)^2 + b(n)^2 + c(n)^2) < \infty, \tag{3}$$

$$c(n) = o(b(n)), \ b(n) = o(a(n)). \tag{4}$$

Examples of $\{a(n)\}$, $\{b(n)\}$ and $\{c(n)\}$ that satisfy (3)-(4) are $a(n) = \frac{1}{n^{3/5}}$, $b(n) = \frac{1}{n^{4/5}}$, $c(n) = \frac{1}{n}$, and $a(n) = \frac{\log n}{n}$, $b(n) = \frac{1}{n}$, $c(n) = \frac{1}{n \log n}$, respectively. Let

$$T_i = \{x_i \stackrel{\triangle}{=} (x_i^1, \ldots, x_i^{N_i-1})^T \mid x_i^j \geq 0, \ j = 1, \ldots, N_i - 1, \text{ and } \sum_{j=1}^{N_i-1} x_i^j \leq 1\}$$

denote the policy simplex in state i onto which, after each policy update recursion, the vector of probabilities corresponding to the first $N_i - 1$ actions is projected. The probability $x_i^{N_i}$ of selecting the N_i-th action in state i is then set according to $x_i^{N_i} = 1 - \sum_{j=1}^{N_i-1} x_i^j$.

For any $i \in S$, let $\triangle_i^j(n)$, $j = 1, \ldots, N_i - 1$, $n \geq 0$, be ± 1-valued variables. These shall constitute the perturbations in SPSA type gradient estimates. Exact values of these for any given n are obtained using a normalized Hadamard matrix based construction as in Bhatnagar et al. (2003) (see below). Let $\triangle_i(n) = (\triangle_i^1(n), \ldots, \triangle_i^{N_i-1}(n))^T$ denote the vector of perturbations at the nth epoch. In general, an $m \times m$ ($m \geq 2$) matrix H is said to be a Hadamard matrix of order m if its entries belong to $\{1, -1\}$ and $H^T H = mI_m$, where I_m is the $m \times m$ identity matrix. A Hadamard matrix is said to be normalized if all the elements in its first column are 1. The construction used by Bhatnagar et al. (2003) that we also use here is the following:

- For $k = 1$, let
$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
- For general $k > 1$,
$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}.$$

For an $(N_i - 1)$-dimensional parameter vector as above, the order of the Hadamard matrix used is $M_i = 2^{\lceil log_2(N_i) \rceil}$. It is easy to see that $N_i - 1 < M_i$. Next form a matrix $\hat{H}_i$ in the following manner: Remove the first column from the normalized Hadamard matrix constructed above. Next pick any $(N_i - 1)$ of the remaining $(M_i - 1)$ columns and all M_i rows to form the new matrix. If only $(N_i - 1)$ columns remain after deleting the first column above, then pick all the remaining columns. Thus $\hat{H}_i$ is an $M_i \times (N_i - 1)$ matrix. Let the M_i rows of this matrix be represented by $\hat{H}_i(1)$, ..., $\hat{H}_i(M_i)$, respectively. Finally, the perturbation sequence $\triangle_i(n)$ is cyclically moved through the sequence

$\{\hat{H}_i(1), \ldots, \hat{H}_i(M_i)\}$ of vectors by setting $\triangle_i(n) = \hat{H}_i(n \bmod M_i + 1)$. In what follows, we present an adaptive single simulation stochastic approximation based algorithm that performs asynchronous updates. Suppose $\nu_i(n)$ denotes the number of times that state i is visited by the MDP $\{X_m\}$ in n epochs. Then, one can write, $\nu_i(n) = \sum_{m=1}^{n} I\{X_m = i\}$. We generate new $\triangle_i(n)$ only for those instants n for which state i is visited by the chain, that is, $X_n = i$. For all other instants, $\theta_i(n)$ and $\triangle_i(n)$ are held fixed. Let $\triangle_i(n)^{-1}$ denote the vector $\triangle_i(n)^{-1} = (\frac{1}{\triangle_i^1(n)}, \ldots, \frac{1}{\triangle_i^{N_i-1}(n)})^T$. We now present our algorithm.

3.1 The Algorithm

Suppose $\delta > 0$ is a given constant and $\Gamma_i : \mathcal{R}^{N_i-1} \rightarrow \mathcal{R}^{N_i-1}$ be the projection from $\mathcal{R}^{N_i-1}$ to the simplex T_i. Let $\theta_i(n)$, $n \geq 0$ denote the nth update of θ_i. Let $\bar{\theta}_i(n) = \Gamma_i(\theta_i(n) + \delta\triangle_i(n))$, where $\triangle_i(n)$, $n \geq 0$ are obtained using normalized Hadamard matrices as explained earlier. We analogously denote $\bar{\theta}_i(n)$ as the vector $\bar{\theta}_i(n) = (\bar{\theta}_i^1(n), \ldots, \bar{\theta}_i^{N_i-1}(n))^T$ and let $\bar{\theta}_i^{N_i}(n) = 1 - \sum_{j=1}^{N_i-1} \bar{\theta}_i^j(n)$. The simulated MDP $\{X_n\}$ is governed by the perturbed randomized policy in the following manner: If $X_n = i$, then an action from the set $A(i)$ is selected according to the randomized policy $\bar{\theta}_i(n)$. Let $Y_i(n)$, $n \geq 0$ be quantities defined via the recursions below that are used for averaging the cost function. Let $V_n(i)$, $i \in S$ denote the nth update of value function and ζ_n the nth update of the risk parameter, respectively. We also let $\theta_i^j(0) = \frac{1}{N_i}$, $\forall j = 1, \ldots, N_i$, $i \in S$, implying that the simulation is started with a policy that assigns equal weightage to every feasible action in each state. Other initial values for the same could be selected as well. The algorithm is described as follows:

The Algorithm

- *Step 0 (Initialize): Fix* $\theta_i(0) \stackrel{\triangle}{=} (\theta_i^1(0), \ldots \theta_i^{N_i-1}(0))^T$, $i \in S$, *as the vectors of initial probabilities for selecting actions in states* i *with* $\theta_i^{N_i}(0) = 1 - \sum_{j=1}^{N_i-1} \theta_i^j$. *Fix integers* L *and (large)* P *arbitrarily. Fix a (small) constant* $\delta > 0$. *Set* $n := 0$ *and* $m := 0$. *Generate* $M_i \times M_i$, *normalized Hadamard matrices* (H_i) *where* $M_i = 2^{\lceil \log_2(N_i) \rceil}$, $i \in S$. *Let* $\hat{H}_i$, $i \in S$, *be* $M_i \times N_i$ *matrices formed from* H_i *by choosing any* N_i *of its columns other than the first and let* $\hat{H}_i(p)$, $p = 1, ..., M_i$ *denote the* M_i *rows of* $\hat{H}_i$. *Now set* $\Delta_i(0) := \hat{H}_i(1)$, $\forall i \in S$. *Set* $\bar{\theta}_i(0) = \Gamma_i(\theta_i(0) + \delta\Delta_i(0))$, $i \in S$ *as the initial value of the perturbed randomized policy. Alternatively, denote* $\bar{\theta}_i(0) = (\bar{\theta}_i^1(0), \ldots, \bar{\theta}_i^{N_i-1}(0))$ *and let* $\bar{\theta}_i^{N_i}(0) = 1 - \sum_{j=1}^{N_i-1} \theta_i^j(0)$. *Obtain initial transition probabilities* $p^{\bar{\theta}_i(0)}(i, j)$, $i, j \in S$ *by setting* $p^{\bar{\theta}_i(0)}(i, j) = \bar{\theta}_i^1(0)p(i, j, a_i^1) + \ldots + \bar{\theta}_i^{N_i}(0)p(i, j, a_i^{N_i})$. *Set* $p_0^{\bar{\theta}_i(0)}(i, j) \stackrel{\triangle}{=} p^{\bar{\theta}_i(0)}(i, j)$ *as the transition probabilities of the new Markov chain. Set* $g^{\bar{\theta}_i(0)}(i) = \bar{\theta}_i^1(0)g(i, a_i^1) + \ldots + \bar{\theta}_i^{N_i}(0)g(i, a_i^{N_i})$ *and* $h^{\bar{\theta}_i(0)}(i, j) = \bar{\theta}_i^1(0)h(i, a_i^1, j) + \ldots + \bar{\theta}_i^{N_i}(0)h(i, a_i^{N_i}, j)$, *respectively. Set* $V_0(i)$, $\forall i \in S$ *as the initial estimates of the cost-to-go function. Also, set* $\zeta_0 = 0$. *Fix a state* $i_0 \in S$ *to be a given reference state and set* $Y_i(0) = 0, \forall i \in S$.

- *Step 1: For all states $X_{nL+m} = i \in S$, simulate the corresponding next states X_{nL+m+1} according to transition probabilities $p^{\bar{\theta}_i(n)}(i,\cdot)$. For all $i \in S$, perform the following updates:*

$$V_{nL+m+1}(i) = V_{nL+m}(i) + a(\nu_i(n))I\{X_{nL+m} = i\}\times$$

$$\left(\frac{\exp(\zeta_{nL+m}g^{\bar{\theta}_i(n)}(i))}{V_{nL+m}(i_0)}V_{nL+m}(X_{nL+m+1})\frac{p^{\bar{\theta}_i(n)}(i,X_{nL+m+1})}{p_n^{\bar{\theta}_i(n)}(i,X_{nL+m+1})} - V_{nL+m}(i)\right) \tag{5}$$

$$\zeta_{nL+m+1} = \zeta_{nL+m} + b(n)\left(\alpha - g^{\bar{\theta}_{X_{nL+m+1}}(n)}(X_{nL+m+1})\right) \tag{6}$$

$$Y_i(nL+m+1) = Y_i(nL+m) + a(\nu_i(n))I\{X_{nL+m} = i\}\times$$

$$\left(h^{\bar{\theta}_i(n)}(i,X_{nL+m+1})\left(\frac{p^{\bar{\theta}_i(n)}(i,X_{nL+m+1})}{p_n^{\bar{\theta}_i(n)}(i,X_{nL+m+1})}\right) - Y_i(nL+m)\right) \tag{7}$$

If $m = L-1$, set $nL := (n+1)L$, $m := 0$ and go to Step 2;

else, set $m := m+1$ and repeat Step 1.

- *Step 2: For all $i \in S$,*

$$\theta_i(n+1) = \Gamma_i\left(\theta_i(n) - c(\nu_i(n))I\{X_{nL} = i\}\frac{Y_i(nL)\triangle_i(\nu_i(n))^{-1}}{\delta}\right). \tag{8}$$

Set $n := n+1$. If $n = P$, go to Step 3;

else, for all $i \in S$, set $\Delta_i(n) := \hat{H}(n \bmod M_i + 1)$ as the new Hadamard matrix generated perturbation. Set $\bar{\theta}_i(n) = (\Gamma_i(\theta_i(n) + \delta\Delta_i(n))$, $i \in S$ as the new perturbed randomized policy. For all $i,j \in S$, set $p^{\bar{\theta}_i(n)}(i,j), = \bar{\theta}_i^1(n)p(i,j,a_i^1) + \ldots + \bar{\theta}_i^{N_i}(n)p(i,j,a_i^{N_i})$. Set $g^{\bar{\theta}_i(n)}(i) = \bar{\theta}_i^1(n)g(i,a_i^1) + \ldots + \bar{\theta}_i^{N_i}(n)g(i,a_i^{N_i})$ and $h^{\bar{\theta}_i(n)}(i,j) = \bar{\theta}_i^1(n)h(i,a_i^1,j) + \ldots + \bar{\theta}_i^{N_i}(n)h(i,a_i^{N_i},j)$, respectively. Finally, for all $i,j \in S$, update estimates $p_n^{\bar{\theta}_i(n)}(i,j)$ of the transition probabilities for the new chain according to

$$p_n^{\bar{\theta}_i(n)}(i,j) = \frac{\exp(\zeta_{nL}g(i,\bar{\theta}_i(n)))}{V_{nL}(i)V_{nL}(i_0)}p^{\bar{\theta}_i(n)}(i,j)V_{nL}(j).$$

Normalize $p_n^{\bar{\theta}_i(n)}(i,j)$ such that $p_n^{\bar{\theta}_i(n)}(i,j) \geq 0, \ \forall i,j$ and $\sum_{j\in S} p_n^{\bar{\theta}_i(n)}(i,j) = 1, \forall i$.

Go to Step 1.

- *Step 3 (termination): Terminate algorithm and output $\bar{\theta}_i(P)$, $i \in S$ as the final randomized policy.*

Remark 1: As we did in the algorithm, and because we found it useful in the experiments, we update the slowest timescale recursion (8) every (given) $L \geq 1$ visits to state i, $i \in S$, and keep the randomized policy fixed in between. This, in effect, amounts to an additional averaging over and above that resulting from the use of different step-size schedules (see also Bhatnagar et al., 2001, 2003) for certain simulation based parametric optimization algorithms that use a similar 'additional' averaging. As observed by Spall (1997) and also Bhatnagar et al. (2003), the one-simulation SPSA algorithms that use randomized perturbation sequences do not show good performance because of

the presence of extra bias terms in the gradient estimates of these. As described in Section 1 (see also the discussion after Eq.(15) below), the use of normalized Hadamard matrices significantly improves performance since all bias terms get cancelled after regular deterministic intervals that are, in general, also significantly shorter in duration as compared to the case when randomized perturbations are used. Finally, even though we present our algorithm for the case when the number of iterations P is fixed apriori, it can be easily modified to allow for stopping criteria based on desired accuracy levels, a scenario that we consider in our numerical experiments in Section 4. The convergence analysis that follows carries through for this case with minor modifications.

3.2 Sketch of Convergence Analysis

The convergence analysis uses the following basic principle of two timescale, or more generally multiple timescale, stochastic approximation (Borkar, 1997): Each iteration in such a scheme can be analyzed separately by treating other iteration(s) on slower timescale(s) as quasi-static, that is, freezing the parameter(s) updated by the latter; while treating other iteration(s) on faster timescale(s) as quasi-equilibrated, that is, averaging the parameter(s) updated by the latter w.r.t. their equilibrium behavior, arrived at similarly by treating all slower components as constants and all faster components as equilibrated. For simplicity of presentation, we show here the analysis for the case corresponding to $L = 1$. The extension to the general case is straightforward (see Bhatnagar et al., 2001, 2003). Let us first consider the synchronous version of the algorithm. Recursions (5)-(8) can be written as follows: For all $i \in S$,

$$V_{n+1}(i) = V_n(i) + a(n)\left(\frac{\exp(\zeta_n g^{\bar{\theta}_i(n)}(i))}{V_n(i_0)} V_n(X_{n+1})\left(\frac{p^{\bar{\theta}_i(n)}(i,X_{n+1})}{p_n^{\bar{\theta}_i(n)}(i,X_{n+1})}\right) - V_n(i)\right), \tag{9}$$

$$\zeta_{n+1} = \zeta_n + b(n)\left(\alpha - g^{\bar{\theta}_{X_{n+1}}(n)}(X_{n+1})\right), \tag{10}$$

$$Y_i(n+1) = Y_i(n) + a(n)\left(h^{\bar{\theta}_i(n)}(i,X_{n+1})\left(\frac{p^{\bar{\theta}_i(n)}(i,X_{n+1})}{p_n^{\bar{\theta}_i(n)}(i,X_{n+1})}\right) - Y_i(n)\right), \tag{11}$$

$$\theta_i(n+1) = \Gamma_i\left(\theta_i(n) - c(n)\frac{Y_i(n)\triangle_i(n)^{-1}}{\delta}\right). \tag{12}$$

Iteration (9):

It can be shown that iteration (9) for fixed ζ_n and $\bar{\theta}_i(n)$ viz., $\zeta_n \equiv \zeta$ and $\bar{\theta}_i(n) \equiv \bar{\theta}_i$, respectively, asymptotically tracks the trajectories of the ordinary differential equation (ODE): For $i \in S$,

$$\dot{x}_t(i) = \frac{\exp(\zeta g^{\bar{\theta}_i}(i))}{x_t(i_0)} \sum_{j\in S} p^{\bar{\theta}_i}(i,j)x_t(j) - x_t(i). \tag{13}$$

The ODE (13) has a unique asymptotically stable fixed point in the positive quadrant (which is invariant under the ODE) which corresponds to the solution to the multiplicative Poisson equation. To see how this comes by, we use the fact that

$$E\left[\frac{\exp(\zeta g^{\bar{\theta}_i}(i))}{V_n(i_0)} V_n(X_{n+1})\left(\frac{p^{\bar{\theta}_i}(i,X_{n+1})}{p_n^{\bar{\theta}_i}(i,X_{n+1})}\right) \mid X_n = i\right] = \frac{\exp(\zeta g^{\bar{\theta}_i}(i))}{V_n(i_0)} \sum_{j\in S} p^{\bar{\theta}_i}(i,j)V_n(j).$$

Thus (9) can be rewritten as

$$\begin{aligned} V_{n+1}(i) &= V_n(i) \\ &+ a(n)\left(\frac{\exp(\zeta g^{\bar{\theta}_i}(i))}{V_n(i_0)}\sum_{j\in S} p^{\bar{\theta}_i}(i,j)V_n(j)) - V_n(i)\right) \\ &+ a(n)\left(\frac{\exp(\zeta_n g^{\bar{\theta}_i(n)}(i))}{V_n(i_0)}V_n(X_{n+1})\left(\frac{p^{\bar{\theta}_i(n)}(i,X_{n+1})}{p_n^{\theta_i(n)}(i,X_{n+1})}\right) - \frac{\exp(\zeta g^{\bar{\theta}_i}(i))}{V_n(i_0)}\sum_{j\in S} p^{\bar{\theta}_i}(i,j)V_n(j))\right). \end{aligned}$$

This is seen as a noisy discretization of the ODE (13) with decreasing stepsize $a(n)$ and a 'martingale difference' or 'noise' error term. The contribution to the net error due to the former vanishes asymptotically because $a(n) \to 0$ and so does the contribution of the latter 'almost surely' following a standard martingale argument. This is a commonly used technique in reinforcement learning based algorithms (see Konda and Borkar, 1999; Bhatnagar and Kumar, 2004), with the idea being to replace conditional averages by evaluation at actual or simulated transitions and, then exploit the incremental nature of stochastic approximation scheme to do the averaging for you.

Iteration (10):

The iteration (10) is a stochastic gradient scheme that, for fixed $\bar{\theta}_i(n) \equiv \bar{\theta}_i$, can be seen, from the first part of Theorem 1 and Lemma 1, to asymptotically track the point ζ_*^{θ} corresponding to the given policy above (using again martingale type arguments and the latter part of (3) on $\{b(n)\}$ now).

Note from (4) that $c(n) = o(b(n))$ and $c(n) = o(a(n))$, respectively. This implies that recursions (9) and (10), respectively, proceed on faster timescales as compared to (12). Moreover, since $b(n) = o(a(n))$ as well, (9) proceeds on a faster scale than (10). Using standard analysis of multi-timescale stochastic approximations (Borkar, 1997), one can show that the iterations (10) and (12) appear to be quasi-static when viewed from the timescale on which (9) is updated. Moreover, when viewed from either of the timescales on which (10) or (12) are updated, the recursion (9) appears to be essentially equilibrated. Similarly, when viewed from the timescale on which (10) is performed, the recursion (9) appears to be equilibrated while, as already stated, (12) appears to be quasi-static. The above justifies selecting time-invariant quantities $\zeta_n \equiv \zeta$ and $\bar{\theta}_i(n) \equiv \bar{\theta}_i$ (resp. $\bar{\theta}_i(n) \equiv \bar{\theta}_i$) in the convergence analysis of recursion (9) (resp. (10)).

Iteration (11):

The iteration (11) proceeds on the fastest timescale $\{a(n)\}$ as well and is merely used to perform averaging of the cost function. The updates from this recursion are then used in the gradient estimate for average cost in the slow timescale recursion (12).

Iteration (12):

Iteration (12) does policy update. Note that here one is interested in finding the minimizing policy parameters (i.e., the probabilities) for the long-run average cost albeit conditioned on the rare event. Thus one is interested in finding the gradient of the average cost. This is achieved by our slow timescale iteration as explained below.

For a bounded, continuous $v_i(\cdot) : \mathcal{R}^{N_i-1} \to \mathcal{R}^{N_i-1}$, define

$$\bar{\Gamma}_i(v_i(y)) = \lim_{\eta\downarrow 0}\left(\frac{\Gamma_i(y+\eta v_i(y)) - \Gamma_i(y)}{\eta}\right).$$

Suppose $\theta = (\theta_1^1, \ldots, \theta_1^{N_1-1}, \ldots, \theta_s^1, \ldots, \theta_s^{N_s-1})^T$ be a given SRP. Let $\hat{J}(\theta)$ denote the long-run average cost under SRP θ. Let $\nabla_i^j \hat{J}(\theta)$ denote the derivative of $\hat{J}(\theta)$ w.r.t. θ_i^j, $j = 1, \ldots, N_i - 1$, and let $\nabla_i \hat{J}(\theta)$ correspond to $\nabla_i \hat{J}(\theta) = (\nabla_i^1 \hat{J}(\theta), \ldots, \nabla_i^{N_i-1} \hat{J}(\theta))^T$. The policy update can be shown to track (in the limits as $P \to \infty$ and $\delta \to 0$) the trajectories of the ODE: For $i \in S$,

$$\dot{\theta}_i(t) = \overline{\Gamma}_i(-\nabla_i \hat{J}(\theta)). \tag{14}$$

The proof broadly proceeds as follows. A standard analysis of (11), see for instance, Bhatnagar and Borkar (1998), Bhatnagar et al. (2001), using the fact that the chain under each stationary policy is irreducible (and hence positive recurrent) shows that

$$\| Y_i(n) - \hat{J}(\overline{\theta}(n)) \| \to 0 \text{ as } n \to \infty.$$

Here $\overline{\theta}(n) = (\overline{\theta}_1(n), \ldots, \overline{\theta}_s(n))^T$. Suppose for all $i \in S$, $\theta_i(n) \in T_i^0$, where T_i^0 corresponds to the interior of the simplex T_i. Then for δ sufficiently small, $\theta_i(n) + \delta\triangle_i(n) \in T_i^0$ as well. Hence $\overline{\theta}_i(n) = \Gamma_i(\theta_i(n) + \delta\triangle_i(n)) = \theta_i(n) + \delta\triangle_i(n)$. Moreover, since $c(n) \to 0$ as $n \to \infty$, $\| \hat{J} \| < \infty$ and $\delta > 0$, one can ensure by choosing n large enough that

$$\Gamma_i \left(\theta_i(n) - c(n) \frac{\hat{J}(\overline{\theta}(n))\triangle_i(n)^{-1}}{\delta} \right) = \theta_i(n) - c(n) \frac{\hat{J}(\theta(n) + \delta\triangle(n))\triangle_i(n)^{-1}}{\delta}.$$

Using a Taylor series expansion of $\hat{J}(\theta(n) + \delta\triangle(n))$ around $\theta(n)$, one obtains

$$\hat{J}(\theta(n) + \delta\triangle(n)) = \hat{J}(\theta(n)) + \delta \sum_{l=1}^{s} \sum_{j=1}^{N_l - 1} \triangle_l^j(n) \nabla_l^j \hat{J}(\theta(n)) + O(\delta^2).$$

For a given $k \in \{1, \ldots, N_i - 1\}$,

$$\frac{\hat{J}(\theta(n) + \delta\triangle(n))}{\delta\triangle_i^k(n)} = \frac{\hat{J}(\theta(n))}{\delta\triangle_i^k(n)} + \nabla_i^k \hat{J}(\theta(n)) + \sum_{j=1, j\neq k}^{N_i - 1} \frac{\triangle_i^j(n) \nabla_i^j \hat{J}(\theta(n))}{\triangle_i^k(n)}$$

$$+ \sum_{l=1, l\neq i}^{s} \sum_{j=1}^{N_l - 1} \frac{\triangle_l^j(n) \nabla_l^j \hat{J}(\theta(n))}{\triangle_i^k(n)} + O(\delta). \tag{15}$$

The first term in the RHS above corresponds to the 'additional' bias term, described earlier, whose overall contribution to bias depends on the magnitude of δ and the frequency with which $\triangle_i^k(n)$ change sign as a function of n, for all k and i. It can be shown as in Theorem 2.5 of Bhatnagar et al. (2003) that for any $n \geq 0$, $\sum_{m=n}^{n+M_i} \frac{1}{\triangle_i^k(m)} = 0$, $\forall k = 1, \ldots, N_i$, and $\sum_{m=n}^{n+M_i} \frac{\triangle_i^j(m)}{\triangle_i^k(m)} = 0$, $\forall j \neq k$, $j, k \in \{1, \ldots, N_i\}$, respectively. Note that because of the use of Hadamard matrices, M_i is typically small, as a result of which the bias contributed by the above terms is not significant in general.

One can also show in a similar manner as Corollary 2.6 of Bhatnagar et al. (2003) that

$$\| \sum_{m=n}^{n+\bar{M}} \sum_{l=1, l\neq i}^{s} \sum_{j=1}^{N_l - 1} \frac{c(m)}{c(n)} \frac{\triangle_l^j(m) \nabla_l^j \hat{J}(\theta(m))}{\triangle_i^k(m)} \| \to 0 \text{ as } n \to \infty,$$

where $\bar{M} = \max(M_1, \ldots, M_s)$. (Recall that M_i is the number of rows in the $\hat{H}_i$, $i = 1, \ldots, s$, matrix defined earlier.) Thus (12) can be seen to be analogous to the recursion

$$\theta_i(n+1) = \Gamma_i(\theta_i(n) - c(n)(\nabla_i \hat{J}(\theta(n)) + \xi_1(n) + O(\delta))), \tag{16}$$

where $\xi_1(n) = o(n)$. In general, one can write $\Gamma_i(\theta_i(n) + \delta\triangle_i(n)) = \theta_i(n) + \delta\triangle_i(n) + \delta r_i(n)$ where $r_i(n)$ correspond to error terms because of the projection operator, such that $\| r_i(n) \| \leq \| \triangle_i(n) \|$ with equality only when $r_i(n) = -\triangle_i(n)$. In the latter case,

$$\| \sum_{m=n}^{n+M_i} \frac{c(m)}{c(n)} \frac{\hat{J}(\theta(m))}{\delta \triangle_i^k(m)} \| \rightarrow 0 \text{ as } n \rightarrow \infty, \ \forall \delta > 0. \tag{17}$$

Finally, we consider the case of any other $\theta_i(n)$ lying on the boundary of T_i. Suppose the correction term $r_i(n) \stackrel{\triangle}{=} (r_i^1(n), \ldots, r_i^{N_i-1}(n))^T$, $i \in S$. Now $\exists\ j \in \{1, \ldots, N_i - 1\}$ for which if sign of $\triangle_i^j(n)$ is such that the vector $\theta_i(n) + \delta\triangle_i(n)$ points outwards from the boundary, then $r_i^j(n) = -\triangle_i^j(n)$. For simplicity, suppose all other $\triangle_i^l(n)$ are such that components $\theta_i^l(n) + \delta\triangle_i^l(n)$ lie inside their respective regions. Then again one can see that (16) is valid. Also, for $k = j$, (17) continues to hold. Now the function $\hat{J}(\cdot)$ itself serves as a Liapunov function for the ODE (14) which has $K \stackrel{\triangle}{=} \{\theta \in T_1 \times T_2 \times \cdots \times T_s \mid \bar{\Gamma}_i(\nabla_i \hat{J}(\theta)) = 0\ \forall i \in S\}$ as its asymptotically stable fixed points. A standard argument now shows that the iterations (12) converge to K almost surely in the limits as $P \rightarrow \infty$ and $\delta \rightarrow 0$. The equilibria for the projected gradient scheme here correspond to Kuhn-Tucker points with the stable ones being local minima. By 'avoidance of traps' results, see Borkar (2003), Brandiere (1998), the scheme converges to one of these with probability one. (Strictly speaking, this requires some additional conditions on the noise component of the iterations that can be ensured by adding independent noise if necessary. Most often, as here, it is empirically observed that the existing noise suffices.)

For the asynchronous case that we actually work with, the step-size sequences are $\{a(\nu_i(n))\}$, $\{b(\nu_i(n))\}$ and $\{c(\nu_i(n))\}$, respectively, and the parameters corresponding to state i are updated only at instants when the MDP $\{X_n\}$ under the running policy visits state i. It can be shown as in Borkar (1998), Borkar (2001), Borkar (2002), and Borkar and Meyn (2002), respectively, that the iterate (5) for fixed ζ and $\bar{\theta}$ as before, asymptotically tracks trajectories of the (combined) ODE

$$\dot{x}_t = \Pi(t) \begin{pmatrix} \frac{\exp(\zeta g^{\bar{\theta}_1}(1))}{x_t(i_0)} \sum_{j \in S} p^{\bar{\theta}_1}(1,j) x_t(j) - x_t(1) \\ \cdot \\ \cdot \\ \cdot \\ \frac{\exp(\zeta g^{\bar{\theta}_s}(s))}{x_t(i_0)} \sum_{j \in S} p^{\bar{\theta}_s}(s,j) x_t(j) - x_t(s) \end{pmatrix}.$$

Here $\Pi(t)$ is an $s \times s$ scaling matrix which is a positive scalar in $[0,1]$ times the identity matrix under some additional technical conditions on the stepsize sequence, see $(i) - (iv)$, pp. 842 of Borkar (1998). Hence this ODE is a time-scaled version of the synchronous ODE. One thus obtains the same result here as before with the only difference being that the convergence to the desired limit points can now become slower as compared to the synchronous case. We now present our numerical results.

4. Numerical Results

The problem of routing multiple flows in communication networks has been well studied during the last few decades (Bertsekas and Gallager, 1991) with several approaches having been proposed for static and dynamic optimization of routing. In Tsitsiklis and Bertsekas (1986) as well as Bertsekas and Gallager (1991), gradient based projection algorithms for optimal routing have been studied. More recently, in Marbach et al. (2000), Nowe et al. (1998) and Varadarajan et al. (2003), reinforcement learning techniques have also been applied to the problem of routing. We consider here an application of our algorithm to finding optimal routes for flows in communication networks, conditioned on a rare event. The basic setting is shown in Fig. 1.

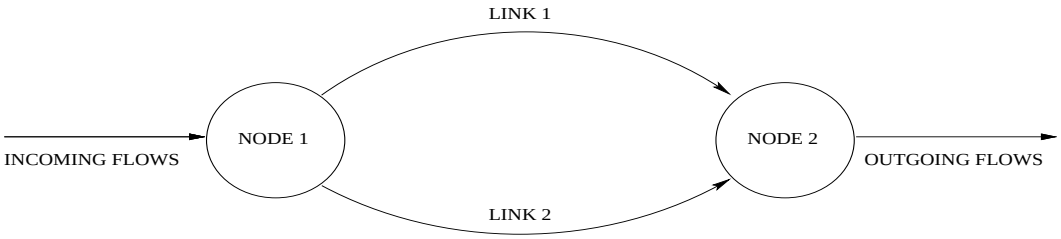


Figure 1: The Model

Nodes A and B are connected via two links. We assume that the system is slotted with time slots of equal length. Customers/flows arrive at the beginning of time slots at A, and have to be sent to B. There are two routes R_1 and R_2 from A to B. An arrival occurs with a certain probability (p) in a given time slot independent of others. At the beginning of a time slot, decision on whether to route all arrivals (that occur in the time slot) onto R_1 or R_2 is made by a controller (at Node A). Thus, all new arrivals at the beginning of a time slot are routed either to R_1 or R_2. However, we also assume that both R_1 and R_2 can accommodate at most M customers (or flows) at any given instant. All flows that cannot be accommodated in a given slot immediately leave the system. Suppose each flow at any given instant (or a slot boundary) finishes service w.p. q_1 on R_1 and w.p. q_2 on R_2, respectively, independent of other flows. Further, if a flow does not finish service in a time slot, its service extends to the next slot independently of the number of flows in either route and the number of slots the given flow has been in service for. The above process is repeated again in subsequent slots. Thus the number of slots that a customer is in service at node j, $j = 1,2$ equals i with probability $(1-q_j)^{i-1}q_j$, for $i \geq 1$. Let $X_n^{(1)}$ (resp. $X_n^{(2)}$) denote the number of flows on R_1 (resp. R_2) in time slot n. Let $\{A(n)\}$ with $A(n) \in \{a_1, a_2\}$ $\forall n \geq 1$, denote the associated action-valued process, where a_i corresponds to the action of routing new flows in a time slot on the route R_i, $i = 1,2$. Then under a given SDP, $\{X_n\}$, where $X_n = (X_n^{(1)}, X_n^{(2)})$, $n \geq 0$, forms a discrete time Markov chain with state transition equation given by

$$\begin{pmatrix} X_{n+1}^{(1)} \\ X_{n+1}^{(2)} \end{pmatrix} = \begin{pmatrix} \min[X_n^{(1)} - Q_1(n) + I\{A(n) = a_1\}B(n), M] \\ \min[X_n^{(2)} - Q_2(n) + I\{A(n) = a_2\}B(n), M] \end{pmatrix},$$

where the departures from routes R_1 and R_2 during time slot n are denoted as $Q_1(n)$ and $Q_2(n)$, respectively, and satisfy $0 \leq Q_j(n) \leq N_j(n)$, $j = 1,2$. Also, $B(n)$ denotes the number of new arrivals at Node A, at the beginning of time slot $(n+1)$. Note that since there are only two actions associated with each state here, the parameter vector $\theta_i(n)$ of the randomized policy is simply $\theta_i(n) = \theta_i^1(n)$. The simplex T_i associated with each state here corresponds to the interval $[0,1]$ $\forall i$. The projection map Γ_i is thus defined by $\Gamma_i(x) = \max(0, \min(x,1))$ $\forall i$. Also, $\bar{\theta}_i(n) = \Gamma_i(\theta_i^1(n) + \delta \triangle_i^1(n))$. The sequences $\{\triangle_i^1(n),\, n \geq 0\}$, $i \in S$ are generated using normalized Hadamard matrices. These turn out to be simply $\triangle_i^1(n) = (-1)^n$. The step-sizes are chosen as $a(n) = b(n) = c(n) = 1$, $n = 0,1$, and for $n \geq 2$,

$$a(n) = \frac{\log(n)}{n}, b(n) = \frac{1}{n}, c(n) = \frac{1}{n\log(n)}.$$

The single-stage cost in state i under policy $\bar{\theta}_i(n)$ is given by $h^{\bar{\theta}_i(n)}(i, X_{n+1}) = |X_{n+1}^{(1)} - N_1| + |X_{n+1}^{(2)} - N_2|$, where N_1 and N_2 are given thresholds and (as before) $X_{n+1} = (X_{n+1}^{(1)}, X_{n+1}^{(2)})$ corresponds to the state at the next instant. The cost function thus aims to keep the number of flows along R_1 to be near threshold N_1 and those along R_2 to be near N_2 for some $0 \leq N_1, N_2 \leq M$. Here the parameters N_1 and N_2 may be set arbitrarily. Note that since all new arrivals in a time slot are routed to either R_1 or R_2, N_1 and N_2 should be judiciously chosen. A value of N_1 or N_2 close to zero would lead to under-utilization while a value close to M would result in leaving less room for accommodating future flows on the corresponding route. The latter is required, for instance, in cases where there are different categories of traffic flows in the network each having a possibly different pay off (a scenario not considered in this paper). Any other choice for the cost function may be used as well.

The function $g^{\cdot}(\cdot)$ used for defining the rare event is given as $g^{\bar{\theta}_{X_n}}(X_n) = I\{X_n^{(2)} > N\}$, where N is another (given integer) threshold. Thus $g^{\cdot}(\cdot)$ equals one if $X_n^{(2)} \in \{N+1, \ldots, M\}$ and is zero otherwise. The long-run average $\lim_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} g^{\bar{\theta}_{X_m}}(X_m)$ in this case corresponds to the stationary probability of the number of flows at the second node exceeding N. For any given SDP, the latter quantity would depend on the resulting transition probability matrix for the process $\{X_n\}$ under that SDP. We consider two different settings for our experiments that we refer to as settings (a) and (b), respectively. The input parameters for the two settings are given in Table 1 below.

Note that in the algorithm in Section 3.1, the number of iterations P is fixed apriori. However, for obtaining more accurate estimates, we use a different stopping criterion for the algorithm that is based on an accuracy parameter ε as explained below and not one based on a fixed value of P. For a given $\varepsilon > 0$, let k_ε be the transition number of the Markov chain at which the estimate of $\rho_\zeta^{\mu^*} \equiv V_\zeta^{\mu^*}(i_0)$ converges to within ε of its previous value 100 times in succession. We let the value of ε to be 5×10^{-9} for setting (a) and 5×10^{-8} for setting (b), respectively. The above values of ε (for the two settings) will in fact be denoted as $\bar{\varepsilon}$. More experiments using other values of ε are subsequently discussed.

In Figs. 2 and 4, we show the optimal policies $\theta^*(\cdot)$ for the two settings. The corresponding value functions are shown in Figs. 3 and 5. We observed from the optimal policies in both settings that for states (i_1, i_2), for given i_1, the value of $\theta^*(\cdot)$, that is, the probability of selecting action a_1, on the whole seems to increase, starting from a low value, as i_2 is increased from 0 to M. Thus, in general, for low values of i_2, for given i_1, the preferred action is a_2 (i.e., to route customers on the second link) while for higher values of i_2, the preferred action becomes a_1. This is along expected

Input Parameter	Setting (a)	Setting (b)
Link Capacity, M	10	20
N_i	$N_1 = 3, N_2 = 5$	$N_1 = 6, N_2 = 12$
N	7	13
α	0.25	0.25
Arrival probability, p	0.65	0.85
Departure probability, q_i	$q_1 = 0.7$, $q_2 = 0.52$	$q_1 = 0.7$, $q_2 = 0.52$
δ	0.01	0.01
L	11	11
$\bar{\varepsilon}$	5.000000e-09	5.000000e-08
n (see Equation (18))	50	150
ζ_0	0	0
$V_0(i), \ \forall i \in S$	1	1
$Y_i(0), \ \forall i \in S$	0	0
Initial policy $\forall i \in S$	$\theta_i^1(0) = \theta_i^2(0) = 0.5$	$\theta_i^1(0) = \theta_i^2(0) = 0.5$
Reference state, i_0	$(2,2)$	$(2,2)$

Table 1: Input Parameters for the two settings

lines given the form of the associated cost function. The value function $V^*(\cdot)$ (in both settings) takes low values for low values of (i_1, i_2) and gradually increases (overall) when either i_1 or i_2 is increased. What is more interesting, however, is that there is a step-increase in these values as soon as the set of rare event states is reached and it stays high over those states. This is not surprising since the conditional probabilities of the rare event states will be higher as we are conditioning on the rare event.

In Table 2, values of various performance metrics under the optimal policy are shown. Note that ζ^* corresponds to the converged value of the risk parameter obtained from the recursion (6). The quantities $E^{\theta_X^*}[X^{(1)}]$ and $E^{\theta_X^*}[X^{(2)}]$ denote the mean numbers of flows on the two routes. These, in general, depend on the parameters p, q_1, q_2, M and θ^*, and in the present case, can be seen to be less than the thresholds N_1 and N_2, in either setting. The mean cost $E^{\theta_X^*}[h^{\theta_i}(i, X^{(1)}, X^{(2)})]$, is higher in Setting (b) as compared to Setting (a) since the values of thresholds N_1 and N_2 in the former setting are higher.

Next, we performed some additional experiments along similar lines as Borkar et al. (2004) and Bucklew (1990), to estimate the rare event probability $\hat{p}_n$ (see below) under the optimal policy for both settings. Note that even though our main aim is to obtain the optimal policy (above), the additional experiments provide insight on the choice of the accuracy parameter ε and its effect on computational performance. We define

$$\hat{p}_n = P_x\left(\frac{1}{n}\sum_{m=0}^{n-1} g^{\theta_{X_m}^*}(X_m) \geq \alpha\right). \tag{18}$$

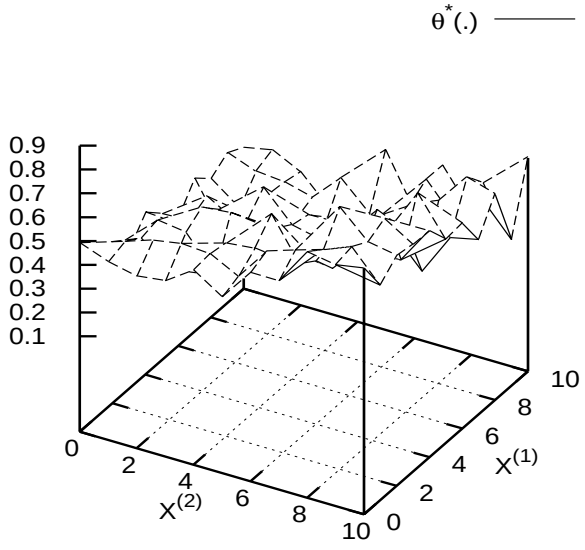


Figure 2: Setting (a): Optimal Policy $\theta^*(\cdot)$

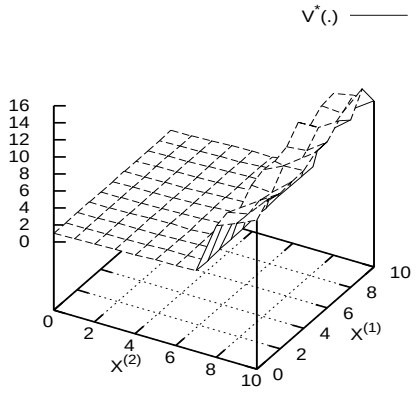


Figure 3: Setting (a): Value Function $V^*(\cdot)$

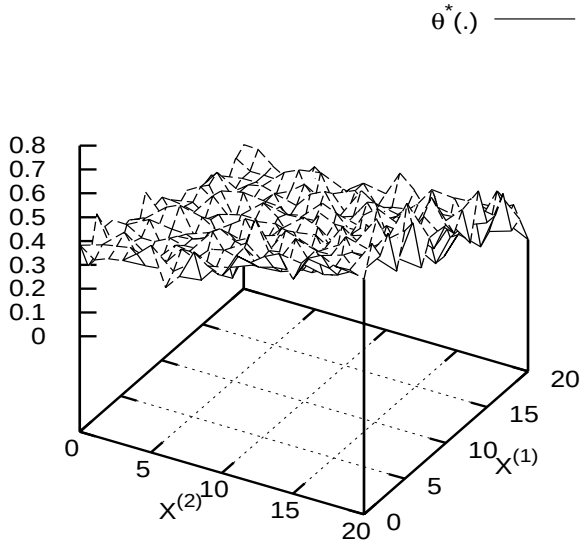


Figure 4: Setting (b): Optimal Policy $\theta^*(\cdot)$

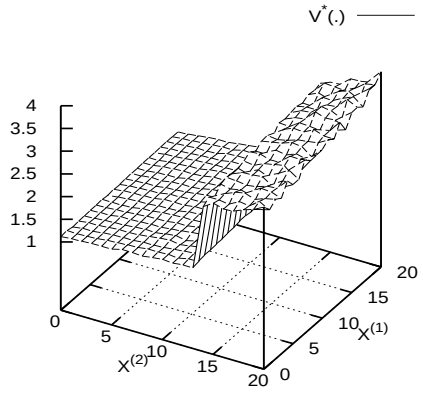


Figure 5: Setting (b): Value Function $V^*(\cdot)$

Performance Metric	Setting (a)	Setting (b)
ζ^*	1.652923e+00	7.370684e-01
$\zeta^*\alpha - \ln(\rho_{\zeta^*})$	2.456064e-01	5.742653e-02
$E^{\theta^*_X}[X^{(1)}]$	1.092038e+00	2.836020e+00
$E^{\theta^*_X}[X^{(2)}]$	4.183547e+00	8.720516e+00
$E^{\theta^*_X}[h^{\theta_i}(i,X^{(1)},X^{(2)})]$	5.488044e+00	1.096857e+01

Table 2: Performance under optimal policy

The values of n are described in Table 1 for the two settings. An importance sampling estimator for this probability is the average of the i.i.d. samples

$$I\{\frac{1}{n}\sum_{m=0}^{n-1} g^{\theta^*}(X_m) \geq \alpha\} \frac{p^{\theta^*_{X_0}}(X_0,X_1)p^{\theta^*_{X_1}}(X_1,X_2)\cdots p^{\theta^*_{X_{n-2}}}(X_{n-2},X_{n-1})}{p_*^{\theta^*_{X_0}}(X_0,X_1)p_*^{\theta^*_{X_1}}(X_1,X_2)\cdots p_*^{\theta^*_{X_{n-2}}}(X_{n-2},X_{n-1})}.$$

In practice, one is able to obtain the above estimate only upto a certain specified degree of accuracy as obtained from the quantity ε (see above). There is however a tradeoff involved in the choice of ε. The variance of the estimates tends to be high if ε is not chosen to be small enough, which may affect their accuracy. On the other hand, as the value of ε is decreased beyond a point, the amount of computational effort required increases rapidly.

We run the algorithm for different values of ε. For each value of ε, we obtain an estimate $p_*^{\varepsilon}(\cdot,\cdot)$ of $p_*^{\theta^*}(\cdot,\cdot)$ that is then used to generate i.i.d. samples for the estimate of the rare event probability $\hat{p}_n$ (see above). The mean and variance of the rare event probability are then determined using the batch means method. The simulation is terminated when the 95% confidence interval, see Law and Kelton (2000), of probability lies within 5% of its estimated mean value. Let T_ε denote the total computational effort involved in terms of the number of simulated transitions of the MDP that are generated during this process. We show in Figs. 6 and 8, plots of k_ε, T_ε and $(k_\varepsilon + T_\varepsilon)$ as functions of ε for settings (a) and (b), respectively. The total computational effort (in terms of $(k_\varepsilon + T_\varepsilon)$) is found to be the least for $\varepsilon \equiv \varepsilon^* = 5 \times 10^{-5}$ in setting (a) and for $\varepsilon \equiv \varepsilon^* = 10^{-4}$ in setting (b), respectively. Also, Figs. 7 and 9 show the plots of the rare event probability $\hat{p}_n$ (described in the figures as p_ε) obtained for different accuracy levels ε. The values of ε in the above figures are shown on the log scale for convenience.

In Table 3, we describe the values of the various parameters and metrics obtained for the rare event probability experiments. The quantities k_{ε^*}, T_{ε^*} and $(k_{\varepsilon^*} + T_{\varepsilon^*})$, respectively, correspond to the case when $\varepsilon = \varepsilon^*$ is chosen for both settings. Also $\bar{\varepsilon} = 5 \times 10^{-9}$ (resp. $\bar{\varepsilon} = 5 \times 10^{-8}$) is the lowest value of ε for which the simulations were run for setting (a) (resp. setting (b)). This level of accuracy was obtained in about 1.18×10^{10} iterations in setting (a) and about 3.05×10^9 iterations in setting (b). As stated previously, the value of $\bar{\varepsilon}$ is used as the accuracy parameter in the earlier experiments (cf. Figs. 2 to 5 and Table 2). In Table 3, p_{ε^*} (resp. $p_{\bar{\varepsilon}}$) corresponds to the value of $\hat{p}_n$ obtained when $\varepsilon = \varepsilon^*$ (resp. $\varepsilon = \bar{\varepsilon}$). Note that these values are much lower for setting (a) than for setting (b) (see also Figs. 7 and 9). As a consequence of the above, the values of k_{ε^*} and T_{ε^*} are seen to be much less for setting (b) as compared to the corresponding values of these for setting (a).

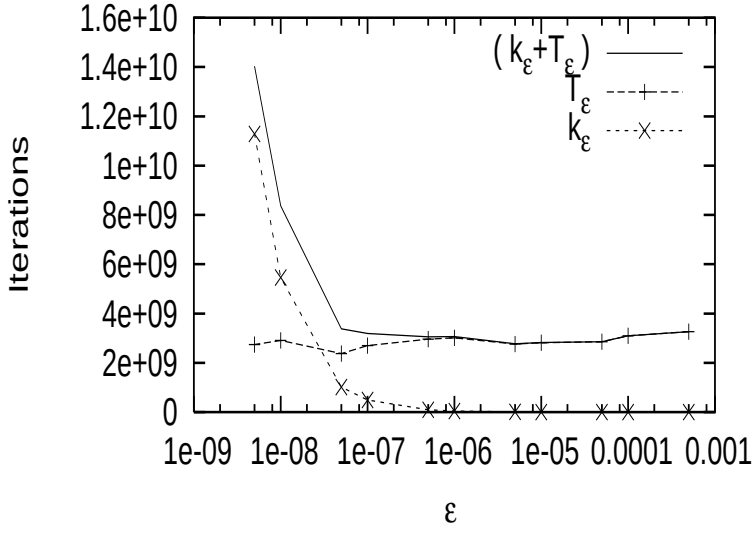


Figure 6: Setting (a): Plot of $k_\varepsilon, T_\varepsilon$ and $(k_\varepsilon + T_\varepsilon)$ w.r.t. ε

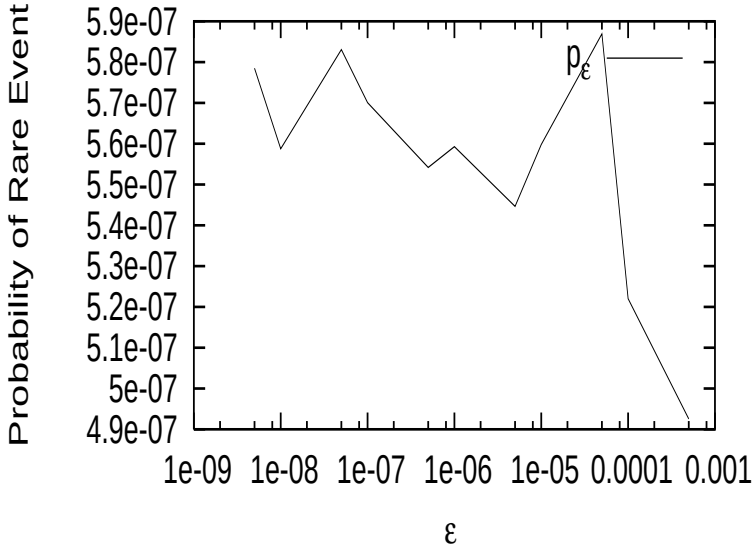


Figure 7: Setting (a): Variation of p_ε with ε

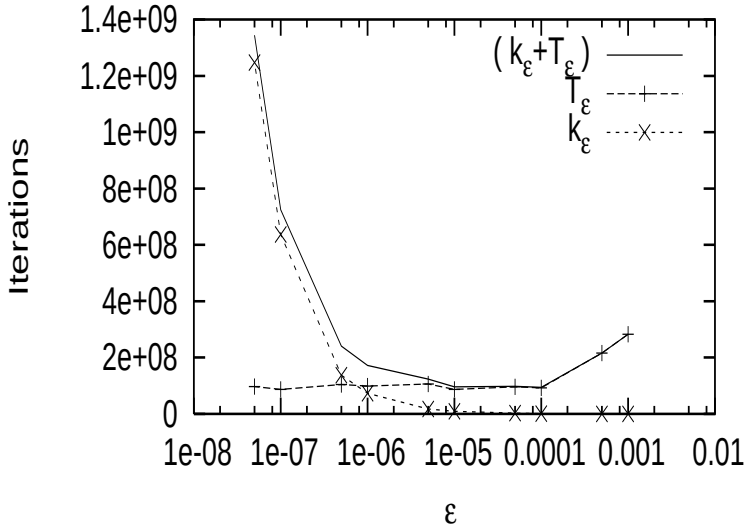


Figure 8: Setting (b): Plot of $k_\varepsilon, T_\varepsilon$ and $(k_\varepsilon + T_\varepsilon)$ w.r.t. ε

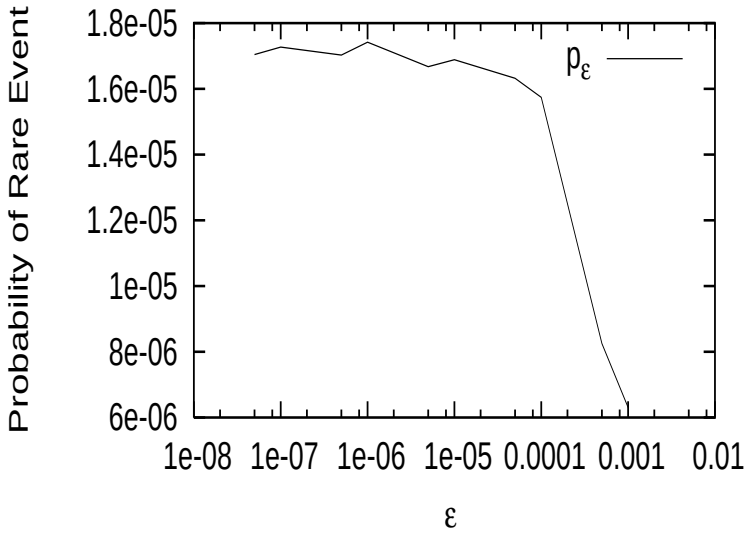


Figure 9: Setting (b): Variation of p_ε with ε

Parameters/Performance Metrics	Setting (a)	Setting (b)
$k_{\bar{\varepsilon}}$	11287258742	1247427803
$p_{\bar{\varepsilon}}$	5.785067e-07	1.704158e-05
ε^*	5.000000e-05	1.000000e-04
k_{ε^*}	9292162	1197983
T_{ε^*}	2760999897	92719997
$(k_{\varepsilon^*} + T_{\varepsilon^*})$	2770292059	93917980
p_{ε^*}	5.446732e-07	1.574290e-05

Table 3: Rare Event Probability Experiments

5. Conclusions

We developed an adaptive simulation based stochastic approximation algorithm for ergodic control of Markov chains conditioned on a rare event of zero probability. Our algorithm uses coupled recursions that are driven by different timescales. We briefly sketched the convergence analysis of our algorithm and presented numerical experiments on a setting involving routing multiple flows in communication networks. The results obtained demonstrate the usefulness of the proposed algorithm in obtaining optimal policies conditioned on a rare event and in estimating the rare event probability. The numerical setting considered here was, however, a simple setting designed to demonstrate the usefulness of the proposed algorithm. More complex settings involving, say, networks with multiple nodes and more routes with large numbers of flows on each should be tried in order to study the scalability of the proposed algorithm. The SPSA technique, in general, is known to be highly scalable as has been demonstrated through several applications over the last decade. In the simulation based optimization framework, SPSA based multi-timescale algorithms have been found to perform well computationally in the case of high-dimensional parameter settings studied in Bhatnagar et al. (2001) and Bhatnagar et al. (2003) (by more than an order of magnitude over related K-W based algorithms). Implementations involving such high-dimensional settings (along the lines described above) need to be studied for the proposed algorithm in the setting of this paper. Recently, in Bhatnagar (2005), certain Newton-based multiscale SPSA algorithms that estimate both the gradient and Hessian of the average cost have been developed in the simulation optimization setting. Similar algorithms for the setting considered here may also be developed.

One may extend these ideas further by applying these for optimal control conditioned on multiple rare events. For problems with large action spaces, one may consider suitable parameterizations of the policy space. One may also use feature based methods for problems with moderately large state spaces. Our adaptive algorithm can be used to derive optimal parameterized policies using features in place of states. It must be noted here that adaptive importance sampling techniques require storage of transition probabilities and our algorithm is no different in this regard. Hence it cannot directly be applied in the case of problems with very large state spaces where storage of such information itself is computationally infeasible. However, in many cases such as queuing networks, the transition probabilities are easy to compute and transitions easy to simulate using simple local dynamic laws. Further, storage of transition probability matrices may not be a major concern in such scenarios since these are known to be highly sparse. Developing similar algorithms in general scenarios involving very large state spaces would be an interesting research direction to pursue.

Acknowledgments

The first author was supported by grant number SR/S3/EE/43/2002-SERC-Engg. from the Department of Science and Technology, Government of India. The second author was supported by grant number III.5(157)/99-ET from the Department of Science and Technology, Government of India.

References

T. P. I. Ahamed, V. S. Borkar, and S. Juneja. Adaptive importance sampling technique for markov chains using stochastic approximation. *Operations Research*, 54(3):489–504, 2006.

S. Balaji and S. P. Meyn. Multiplicative ergodicity and large deviations for an irreducible markov chain. *Stochastic Processes and their Appl.*, 90:123–144, 2000.

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

J. Baxter, P. L. Bartlett, and L. Weaver. Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:351–381, 2001.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, USA, 2001.

D. P. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, New Jersey, USA, 1991.

D. P. Bertsekas and J. Tsitsiklis. *Neuro-dynamic Programming*. Athena Scientific, Boston, MA, USA, 1996.

S. Bhatnagar. Adaptive multivariate three-timescale stochastic approximation algorithms for simulation based optimization. *ACM Transactions on Modelling and Computer Simulation*, 15(1): 74–107, 2005.

S. Bhatnagar and V. S. Borkar. Multiscale stochastic approximation for parametric optimization of hidden Markov models. *Prob. Engg. and Info. Sci.*, 11:509–522, 1997.

S. Bhatnagar and V. S. Borkar. A two time scale stochastic approximation scheme for simulation based parametric optimization. *Prob. Engg. and Info. Sci.*, 12:519–531, 1998.

S. Bhatnagar, M. C. Fu, S. I. Marcus, and S. Bhatnagar. Two timescale algorithms for simulation optimization of hidden Markov models. *IIE Transactions*, 33(3):245–258, 2001.

S. Bhatnagar, M. C. Fu, S. I. Marcus, and I-J. Wang. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modelling and Computer Simulation*, 13(2):180–209, 2003.

S. Bhatnagar and S. Kumar. A simultaneous perturbation stochastic approximation based actor-critic algorithm for markov decision processes. *IEEE Transactions on Automatic Control*, 49(4): 592–598, 2004.

V. S. Borkar. Stochastic approximation with two timescales. *System and Control Letters*, 29:291–294, 1997.

V. S. Borkar. Asynchronous stochastic approximations. *SIAM J. Control and Optimization*, 36: 840–851, 1998.

V. S. Borkar. A sensitivity formula for risk-sensitive cost and the actor-critic algorithm. *System and Control Letters*, 44:339–346, 2001.

V. S. Borkar. Q-learning for risk-sensitive control. *Mathematics of Operations Research*, 27:294–311, 2002.

V. S. Borkar. Avoidance of traps in stochastic approximation. *System and Control Letters*, 50:1–9, 2003.

V. S. Borkar, S. Juneja, and A. A. Kherani. Performance analysis conditioned on rare events: an adaptive simulation scheme. *Communications in Information and Systems*, 3:259–278, 2004.

V. S. Borkar and S. P. Meyn. Risk-sensitive optimal control for markov decision processes with monotone cost. *Mathematics of Operations Research*, 27:192–209, 2002.

O. Brandiere. Some pathological traps for stochastic approximation. *SIAM J. Contr. and Optim.*, 36:1293–1314, 1998.

J. Bucklew. *Large Deviations Techniques in Decision, Simulation and Estimation*. John Wiley, New York, 1990.

X.-R. Cao. The relations among potentials, perturbation analysis, and markov decision processes. *Discrete Event Dynamic Systems*, 8:71–87, 1998.

X.-R. Cao and X. Guo. A unified approach to markov decision problems and performance sensitivity analysis with discounted and average criteria: multichain cases. *Automatica*, 40:1749–1759, 2004.

E. K. P. Chong and P. J. Ramadge. Stochastic optimization of regenerative systems using infinitesimal perturbation analysis. *IEEE Trans. Auto. Cont.*, 39(7):1400–1410, 1994.

D. Hernández-Hernández and S. I. Marcus. Risk sensitive control of markov processes in countable state space. *Systems and Control Letters*, 29:147–155, 1996.

Y. C. Ho and X. R. Cao. *Perturbation Analysis of Discrete Event Dynamical Systems*. Kluwer, Boston, 1991.

V. R. Konda and V. S. Borkar. Actor-critic like learning algorithms for markov decision processes. *SIAM Journal on Control and Optimization*, 38(1):94–123, 1999.

I. Kontoyiannis and S. P. Meyn. Spectral theory and limit theorems for geometrically ergodic markov processes. *Annals of Applied Probability*, 13:304–362, 2003.

A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, 2000.

P. Marbach, O. Mihatsch, and J. N. Tsitsiklis. Call admission control and routing in integrated services networks using neuro-dynamic programming. *IEEE J. Selected Areas in Communications*, 18(2):197–208, 2000.

P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46:191–209, 2001.

A. Nowe, K. Steenhaut, M. Fakir, and K. Veerbeck. Q-learning for adaptive load based routing. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, California, USA, 1998. San Diego.

M. Puterman. *Markov Decision Processes*. Wiley Inter-science, 1994.

R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 19(1):89–112, 1997.

J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

J. C. Spall. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica*, 33:109–112, 1997.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

J. N. Tsitsiklis and D. P. Bertsekas. Distributed asynchronous optimal routing in data networks. *IEEE Transactions on Automatic Control*, 31:325–332, 1986.

S. Varadarajan, N. Ramakrishnan, and M. Thirunavukkarasu. Reinforcing reachable routes. *Computer Networks*, 43(3):389–416, 2003.

C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Journal of Machine Learning Research 7 (2006) 1963-2001 Submitted 3/05; Revised 7/06; Published 10/06

Learning Spectral Clustering, With Application To Speech Separation

Francis R. Bach FRANCIS.BACH@MINES.ORG
Centre de Morphologie Mathématique
Ecole des Mines de Paris
35, rue Saint Honoré
77300 Fontainebleau, France

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA

Editor: Yoshua Bengio

Abstract

Spectral clustering refers to a class of techniques which rely on the eigenstructure of a similarity matrix to partition points into disjoint clusters, with points in the same cluster having high similarity and points in different clusters having low similarity. In this paper, we derive new cost functions for spectral clustering based on measures of error between a given partition and a solution of the spectral relaxation of a minimum normalized cut problem. Minimizing these cost functions with respect to the partition leads to new spectral clustering algorithms. Minimizing with respect to the similarity matrix leads to algorithms for learning the similarity matrix from fully labelled data sets. We apply our learning algorithm to the blind one-microphone speech separation problem, casting the problem as one of segmentation of the spectrogram.

Keywords: spectral clustering, blind source separation, computational auditory scene analysis

1. Introduction

Spectral clustering has many applications in machine learning, exploratory data analysis, computer vision and speech processing. Most techniques explicitly or implicitly assume a metric or a similarity structure over the space of configurations, which is then used by clustering algorithms. The success of such algorithms depends heavily on the choice of the metric, but this choice is generally not treated as part of the learning problem. Thus, time-consuming manual feature selection and weighting is often a necessary precursor to the use of spectral methods.

Several recent papers have considered ways to alleviate this burden by incorporating prior knowledge into the metric, either in the setting of K-means clustering (Wagstaff et al., 2001; Xing et al., 2003; Bar-Hillel et al., 2003) or spectral clustering (Yu and Shi, 2002; Kamvar et al., 2003). In this paper, we consider a complementary approach, providing a general framework for learning the similarity matrix for spectral clustering from examples. We assume that we are given sample data with known partitions and are asked to build similarity matrices that will lead to these partitions when spectral clustering is performed. This problem is motivated by the availability of such data sets for at least two domains of application: in vision and image segmentation, databases of hand-labelled segmented images are now available (Martin et al., 2001), while for the blind separation

of speech signals via partitioning of the time-frequency plane (Brown and Cooke, 1994), training examples can be created by mixing previously captured signals.

Another important motivation for our work is the need to develop spectral clustering methods that are robust to irrelevant features. Indeed, as we show in Section 4.5, the performance of current spectral methods can degrade dramatically in the presence of such irrelevant features. By using our learning algorithm to learn a diagonally-scaled Gaussian kernel for generating the similarity matrix, we obtain an algorithm that is significantly more robust.

Our work is based on cost functions $J_1(W,E)$ and $J_2(W,E)$ that characterize how close the eigenstructure of a similarity matrix W is to a partition E. We derive these cost functions in Section 2. As we show in Section 2.5, minimizing those cost functions with respect to the partition E leads to new clustering algorithms that take the form of weighted K-means algorithms. Minimizing them with respect to W yields a theoretical framework for learning the similarity matrix, as we show in Section 3. Section 3.3 provides foundational material on the approximation of the eigensubspace of a symmetric matrix that is needed for Section 4, which presents learning algorithms for spectral clustering.

We highlight one other aspect of the problem here—the major computational challenge involved in applying spectral methods to domains such as vision or speech separation. Indeed, in image segmentation, the number of pixels in an image is usually greater than hundreds of thousands, leading to similarity matrices of potential huge sizes, while, for speech separation, four seconds of speech sampled at 5.5 kHz yields 22,000 samples and thus a naive implementation would need to manipulate similarity matrices of dimension at least $22,000 \times 22,000$. Thus a major part of our effort to apply spectral clustering techniques to speech separation has involved the design of numerical approximation schemes that exploit the different time scales present in speech signals. In Section 4.4, we present numerical techniques that are appropriate for generic clustering problems, while in Section 6.3, we show how these techniques specialize to speech.

2. Spectral Clustering and Normalized Cuts

In this section, we present our spectral clustering framework. Following Shi and Malik (2000) and Gu et al. (2001), we derive the spectral relaxation through normalized cuts. Alternative frameworks, based on Markov random walks (Meila and Shi, 2002), on different definitions of the normalized cut (Meila and Xu, 2003), or on constrained optimization (Higham and Kibble, 2004), lead to similar spectral relaxations.

2.1 Similarity Matrices

Spectral clustering refers to a class of techniques for clustering that are based on pairwise similarity relations among data points. Given a data set I of P points in a space $\mathcal{X}$, we assume that we are given a $P \times P$ "similarity matrix" W that measures the similarity between each pair of points: $W_{pp'}$ is large when points indexed by p and p' are preferably in the same cluster, and is small otherwise. The goal of clustering is to organize the data set into disjoint subsets with high intra-cluster similarity and low inter-cluster similarity.

Throughout this paper we always assume that the elements of W are nonnegative ($W \geqslant 0$) and that W is symmetric ($W = W^\top$). Moreover, we make the assumption that the diagonal elements of W are strictly positive. In particular, contrary to most work on kernel-based algorithms, our theoretical framework makes no assumptions regarding the positive semidefiniteness of the matrix

(a symmetric matrix W is positive semidefinite if and only if for all vectors $u \in \mathbb{R}^P$, $u^\top W u \geqslant 0$). If in fact the matrix is positive semidefinite this can be exploited in the design of efficient approximation algorithms (see Section 4.4). But the spectral clustering algorithms presented in this paper are not limited to positive semidefinite matrices.

A classical similarity matrix for clustering in $\mathbb{R}^d$ is the diagonally-scaled Gaussian similarity, defined between pairs of points $(x,y) \in \mathbb{R}^d \times \mathbb{R}^d$ as:

$$W(x,y) = \exp(-(x-y)^\top \mathrm{Diag}(\alpha)(x-y)),$$

where $\alpha \in \mathbb{R}^d$ is a vector of positive parameters, and $\mathrm{Diag}(\alpha)$ denotes the $d \times d$ diagonal matrix with diagonal α. It is also very common to use such similarity matrices after transformation to a set of "features," where each feature can depend on the entire data set $(x_i)_{i=1,\ldots,P}$ or a subset thereof (see, for example, Shi and Malik, 2000, for an example from computational vision and see Section 5 of the current paper for examples from speech separation).

In the context of *graph partitioning* where data points are vertices of an undirected graph and W_{ij} is defined to be one if there is an edge between i and j, and zero otherwise, W is often referred to as an "affinity matrix" (Chung, 1997).

2.2 Normalized Cuts

We let $V = \{1,\ldots,P\}$ denote the index set of all data points. We wish to find R disjoint clusters, $A = (A_r)_{r\in\{1,\ldots,R\}}$, where $\bigcup_r A_r = V$, that optimize a certain cost function. In this paper, we consider the R-way normalized cut, $C(A,W)$, defined as follows (Shi and Malik, 2000; Gu et al., 2001). For two subsets A, B of V, define the total weight between A and B as $W(A,B) = \sum_{i\in A}\sum_{j\in B} W_{ij}$. Then the normalized cut is equal to:

$$C(A,W) = \sum_{r=1}^{R} \frac{W(A_r, V\backslash A_r)}{W(A_r,V)}. \tag{1}$$

Noting that $W(A_r,V) = W(A_r,A_r) + W(A_r,V\backslash A_r)$, we see that the normalized cut is small if for all r, the weight between the r-th cluster and the remaining data points is small compared to the weight within that cluster. The normalized cut criterion thus penalizes unbalanced partitions, while non-normalized criteria do not and often lead to trivial solutions (e.g., a cluster with only one point) when applied to clustering. In addition to being more immune to outliers, the normalized cut criterion and the ensuing spectral relaxations have a simpler theoretical asymptotic behavior when the number of data points tend to infinity (von Luxburg et al., 2005).

Let e_r be the indicator vector in $\mathbb{R}^P$ for the r-th cluster, that is, $e_r \in \{0,1\}^P$ is such that e_r has a nonzero component only for points in the r-th cluster. Knowledge of $E = (e_1,\ldots,e_R) \in \mathbb{R}^{P\times R}$ is equivalent to knowledge of $A = (A_1,\ldots,A_R)$ and, when referring to partitions, we will use the two formulations interchangeably. A short calculation reveals that the normalized cut is then equal to:

$$C(E,W) = \sum_{r=1}^{R} \frac{e_r^\top (D-W) e_r}{e_r^\top D e_r},$$

where D denotes the diagonal matrix whose i-th diagonal element is the sum of the elements in the i-th row of W, that is, $D = \mathrm{Diag}(W\mathbf{1})$, where $\mathbf{1}$ is defined as the vector in $\mathbb{R}^P$ composed of ones. Since we have assumed that all similarities are nonnegative, the matrix $L = D - W$, usually

referred to as the "Laplacian matrix," is a positive semidefinite matrix (Chung, 1997). In addition, its smallest eigenvalue is always zero, with eigenvector $\mathbf{1}$. Also, we have assumed that the diagonal of W is strictly positive, which implies that D is positive definite. Finally, in the next section, we also consider the normalized Laplacian matrix defined as $\widetilde{L} = I - D^{-1/2}WD^{-1/2}$. This matrix is also positive definite with zero as its smallest eigenvalue, associated with eigenvector $D^{1/2}\mathbf{1}$.

Minimizing the normalized cut is an NP-hard problem (Shi and Malik, 2000; Meila and Xu, 2003). Fortunately, tractable relaxations based on eigenvalue decomposition can be found.

2.3 Spectral Relaxation

The following proposition, which extends a result of Shi and Malik (2000) for two clusters to an arbitrary number of clusters, gives an alternative description of the clustering task, and leads to a spectral relaxation:

Proposition 1 *For all partitions E into R clusters, the R-way normalized cut $C(W,E)$ is equal to $R - \mathrm{tr}\, Y^\top D^{-1/2}WD^{-1/2}Y$ for any matrix $Y \in \mathbb{R}^{P\times R}$ such that:*

(a) the columns of $D^{-1/2}Y$ are piecewise constant with respect to the clusters E,

(b) Y has orthonormal columns ($Y^\top Y = I$).

Proof The constraint (a) is equivalent to the existence of a matrix $\Lambda \in \mathbb{R}^{R\times R}$ such that $D^{-1/2}Y = E\Lambda$. The constraint (b) is thus written as $I = Y^\top Y = \Lambda^\top E^\top DE\Lambda$. The matrix $E^\top DE$ is diagonal, with elements $e_r^\top De_r$ and is thus positive and invertible. The $R \times R$ matrix $M = (E^\top DE)^{1/2}\Lambda$ satisfies $M^\top M = I$, that is, M is orthogonal, which implies $I = MM^\top = (E^\top DE)^{1/2}\Lambda\Lambda^\top(E^\top DE)^{1/2}$.

This immediately implies that $\Lambda\Lambda^\top = (E^\top DE)^{-1}$. Thus we have:

$$\begin{aligned} R - \mathrm{tr}\, Y^\top (D^{-1/2}WD^{-1/2})Y &= R - \mathrm{tr}\,\Lambda^\top E^\top D^{1/2}(D^{-1/2}WD^{-1/2})D^{1/2}E\Lambda \\ &= R - \mathrm{tr}\,\Lambda^\top E^\top WE\Lambda \\ &= R - E^\top WE\Lambda\Lambda^\top = \mathrm{tr}\, E^\top WE(E^\top DE)^{-1} \\ &= C(W,E), \end{aligned}$$

which completes the proof. ■

By removing the constraint (a), we obtain a relaxed optimization problem, whose solutions involve the eigenstructure of $D^{-1/2}WD^{-1/2}$ and which leads to the classical lower bound on the optimal normalized cut (Zha et al., 2002; Chan et al., 1994). The following proposition gives the solution obtained from the spectral relaxation[1]:

Proposition 2 *The maximum of $\mathrm{tr}\, Y^\top D^{-1/2}WD^{-1/2}Y$ over matrices $Y \in \mathbb{R}^{P\times R}$ such that $Y^\top Y = I$ is the sum of the R largest eigenvalues of $D^{-1/2}WD^{-1/2}$. It is attained at all Y of the form $Y = UB_1$ where $U \in \mathbb{R}^{P\times R}$ is any orthonormal basis of the R-th principal subspace of $D^{-1/2}WD^{-1/2}$ and B_1 is an arbitrary orthogonal matrix in $\mathbb{R}^{R\times R}$.*

Proof Let $\widetilde{W} = D^{-1/2}WD^{-1/2}$. The proposition is equivalent to the classical variational characterization of the sum of the R largest eigenvalues $\lambda_1(\widetilde{W}) \geqslant \cdots \geqslant \lambda_R(\widetilde{W})$ of $\widetilde{W}$—a result known as Ky Fan's theorem (Overton and Womersley, 1993):

$$\lambda_1(\widetilde{W}) + \cdots + \lambda_R(\widetilde{W}) = \max\{\mathrm{tr}\, Y^\top \widetilde{W}Y, Y \in \mathbb{R}^{P\times R}, Y^\top Y = I\},$$

1. Tighter relaxations that exploit the nonnegativity of cluster indicators can be obtained (Xing and Jordan, 2003). These lead to convex relaxations, but their solution cannot be simply interpreted in terms of eigenvectors.

where the maximum is attained for all matrices Y of the form $Y = UB_1$, where $U \in \mathbb{R}^{P\times R}$ is any orthonormal basis of the R-th principal subspace of $\widetilde{W}$ and B_1 is an arbitrary orthogonal matrix in $\mathbb{R}^{R\times R}$. Note that the R-th principal subspace is uniquely defined if and only if $\lambda_R \neq \lambda_{R+1}$ (i.e., there is a positive eigengap). ■

The solutions found by this relaxation will not in general be piecewise constant, that is, they will not in general satisfy constraint (a) in Proposition 1, and thus the relaxed solution has to be projected back to the constraint set defined by (a), an operation we refer to as "rounding," due to the similarity with the rounding performed after a linear programming relaxation of an integer programming problem (Bertsimas and Tsitsiklis, 1997).

2.4 Rounding

Our rounding procedure is based on the minimization of a metric between the relaxed solution and the entire set of discrete allowed solutions. Different metrics lead to different rounding schemes. In this section, we present two different metrics that take into account the known invariances of the problem.

2.4.1 COMPARISON OF SUBSPACES

Solutions of the relaxed problem are defined up to an orthogonal matrix, that is, $Y_{eig} = UB_1$, where $U \in \mathbb{R}^{P\times R}$ is any orthonormal basis of the R-th principal subspace of M and B_1 is an arbitrary orthogonal matrix. The set of matrices Y that correspond to a partition E and that satisfy constraints (a) and (b) are of the form $Y_{part} = D^{1/2}E(E^\top DE)^{-1/2}B_2$, where B_2 is an arbitrary orthogonal matrix.

Since both matrices are defined up to an orthogonal matrix, it makes sense to compare the *subspaces* spanned by their columns. A common way to compare subspaces is to compare the orthogonal projection operators on those subspaces (Golub and Loan, 1996), that is, to compute the Frobenius norm between $Y_{eig}Y_{eig}^\top = UU^\top$ and the orthogonal projection operator $\Pi_0(W,E)$ on the subspace spanned by the columns of $D^{1/2}E = D^{1/2}(e_1,\ldots,e_r)$, equal to:

$$\begin{aligned}\Pi_0(W,E) &= Y_{part}Y_{part}^\top \\ &= D^{1/2}E(E^\top DE)^{-1}E^\top D^{1/2} \\ &= \sum_r \frac{D^{1/2}e_r e_r^\top D^{1/2}}{e_r^\top De_r}.\end{aligned}$$

We thus define the following cost function:

$$J_1(W,E) = \tfrac{1}{2}\left\|U(W)U(W)^\top - \Pi_0(W,E)\right\|_F^2. \tag{2}$$

Other cost functions could be derived using different metrics between linear subspaces, but as shown in Section 2.5, the Frobenius norm between orthogonal projections has the appealing feature that it leads to a weighted K-means algorithm.[2]

2. Another natural possibility followed by Yu and Shi (2003) is to compare directly U (or a normalized version thereof) with the indicator matrix E, up to an orthogonal matrix R, which then has to be estimated. This approach leads to an alternating minimization scheme similar to K-means.

Using the fact that both $U(W)U(W)^\top$ and $\Pi_0(W,E)$ are orthogonal projection operators on linear subspaces of dimension R, we have:

$$\begin{aligned} J_1(W,E) &= \frac{1}{2}\mathrm{tr}\,U(W)U(W)^\top + \frac{1}{2}\mathrm{tr}\,\Pi_0(W,E)\Pi_0(W,E)^\top - \mathrm{tr}\,U(W)U(W)^\top\Pi_0(W,E) \\ &= \frac{R}{2} + \frac{R}{2} - \mathrm{tr}\,U(W)U(W)^\top\Pi_0(W,E) \\ &= R - \sum_r \frac{e_r^\top D^{1/2}U(W)U(W)^\top D^{1/2}e_r}{e_r^\top De_r}. \end{aligned}$$

Note that if the similarity matrix W has rank equal to R, then our cost function $J_1(W,E)$ is exactly equal to the normalized cut $C(W,E)$.

2.4.2 NORMALIZATION OF EIGENVECTORS

By construction of the orthonormal basis U of the R-dimensional principal subspace of $\widetilde{W} = D^{-1/2}WD^{-1/2}$, the P R-dimensional rows $u_1,\ldots,u_P \in \mathbb{R}^R$ are already *globally* normalized, that is, they satisfy $U^\top U = \sum_{i=1}^P u_i u_i^\top = I$. Additional renormalization of those eigenvectors has proved worthwhile in clustering applications (Scott and Longuet-Higgins, 1990; Weiss, 1999; Ng et al., 2002), as can be seen in the idealized situation in which the similarity is zero between points that belong to different clusters and strictly positive between points in the same clusters. In this situation, the eigenvalue 1 has multiplicity R, and $D^{1/2}E$ is an orthonormal basis of the principal subspace. Thus, any basis U of the principal subspace has rows which are located on orthogonal *rays* in $\mathbb{R}^R$, where the distance from the i-th row u_i to the origin is simply $D_{ii}^{1/2}$. By normalizing each row by the value $D_{ii}^{1/2}$ or by its norm $\|u_i\|$, the rows become orthonormal points in $\mathbb{R}^R$ (in the idealized situation) and thus are trivial to cluster. Ng et al. (2002) have shown that when the similarity matrix is "close" to this idealized situation, the properly normalized rows tightly cluster around an orthonormal basis.

We also define an alternative cost function by removing the scaling introduced by D; that is, we multiply U by $D^{-1/2}$ and re-orthogonalize to obtain: $V = D^{1/2}U(U^\top D^{-1}U)^{-1/2}$, where we use any of the matrix square roots of $U^\top D^{-1}U$ (our framework is independent of the chosen square root). Note that this is equivalent to considering the generalized eigenvectors (i.e., vectors $x \neq 0$ such that $Wx = \lambda Dx$ for a certain λ). We thus define the following cost function:

$$\begin{aligned} J_2(W,E) &= \frac{1}{2}\left\| V(W)V(W)^\top - E(E^\top E)^{-1}E^\top \right\|_F^2 \\ &= \frac{1}{2}\left\| V(W)V(W)^\top - \sum_{i=1}^r \frac{e_r e_r^\top}{e_r^\top e_r} \right\|_F^2. \end{aligned} \tag{3}$$

Our two cost functions characterize the ability of the matrix W to produce the partition E when using its eigenvectors. Minimizing with respect to E leads to new clustering algorithms that we now present. Minimizing with respect to the matrix W for a given partition E leads to algorithms for learning the similarity matrix, as we show in Section 3 and Section 4.

In practice, the two cost functions lead to very similar results. The first cost function has closest ties to the normalized cut problem since it is equal to the normalized cut for similarity matrices of rank R, while the second cost function leads to better theoretical learning bounds as shown in Section 3.2.

2.5 Spectral Clustering Algorithms

In this section, we provide a variational formulation of our two cost functions. Those variational formulations lead naturally to K-means and weighted K-means algorithms for minimizing those cost functions with respect to the partition. While K-means is often used heuristically as a post-processor for spectral clustering (Ng et al., 2002; Meila and Shi, 2002), our approach provides a mathematical foundation for the use of K-means.

2.6 Weighted K-means

The following theorem, inspired by the spectral relaxation of K-means presented by Zha et al. (2002), shows that the cost function can be interpreted as a weighted distortion measure:

Theorem 3 *Let W be a similarity matrix and let $U = (u_1, \ldots, u_P)^\top$, where $u_p \in \mathbb{R}^R$, be an orthonormal basis of the R-th principal subspace of $D^{-1/2}WD^{-1/2}$, and $d_p = D_{pp}$ for all p. For any partition $E \equiv A$, we have*

$$J_1(W,E) = \min_{(\mu_1,\ldots,\mu_R)\in\mathbb{R}^{R\times R}} \sum_r \sum_{p\in A_r} d_p \|u_p d_p^{-1/2} - \mu_r\|^2.$$

Proof Let $D(\mu,A) = \sum_r \sum_{p\in A_r} d_p \|u_p d_p^{-1/2} - \mu_r\|^2$. Minimizing $D(\mu,A)$ with respect to μ is a decoupled least-squares problem and we get:

$$\begin{aligned}
\min_\mu D(\mu,A) &= \textstyle\sum_r \sum_{p\in A_r} u_p^\top u_p - \sum_r \|\sum_{p\in A_r} d_p^{1/2} u_p\|^2 / (\sum_{p\in A_r} d_p) \\
&= \textstyle\sum_p u_p^\top u_p - \sum_r \sum_{p,p'\in A_r} d_p^{1/2} d_{p'}^{1/2} u_p^\top u_{p'} / (e_r^\top D e_r) \\
&= R - \textstyle\sum_r e_r^\top D^{1/2} U U^\top D^{1/2} e_r / (e_r^\top D e_r) = J_1(W,E).
\end{aligned}$$

■

This theorem has an immediate algorithmic implication—to minimize the cost function $J_1(W,E)$ with respect to the partition E, we can use a weighted K-means algorithm. The resulting algorithm is presented in Figure 1.

For the second cost function, we have a similar theorem, which leads naturally to the K-means algorithm presented in Figure 2:

Theorem 4 *Let W be a similarity matrix and let U be an orthonormal basis of the R-th principal subspace of $D^{-1/2}WD^{-1/2}$, and $V = D^{1/2}U(U^\top DU)^{-1/2}$. For any partition $E \equiv A$, we have*

$$J_2(W,E) = \min_{(\mu_1,\ldots,\mu_R)\in\mathbb{R}^{R\times R}} \sum_r \sum_{p\in A_r} \|v_p - \mu_r\|^2.$$

The rounding procedures that we propose in this paper are similar to those in other spectral clustering algorithms (Ng et al., 2002; Yu and Shi, 2003). Empirically, all such rounding schemes usually lead to similar partitions. The main advantage of our procedure—which differs from the others in being derived from a cost function—is that it naturally leads to an algorithm for learning the similarity matrix from data, presented in Section 3.

Input: Similarity matrix $W \in \mathbb{R}^{P \times P}$.

Algorithm:

1. Compute first R eigenvectors U of $D^{-1/2}WD^{-1/2}$ where $D = \mathrm{diag}(W\mathbf{1})$.
2. Let $U = (u_1, \ldots, u_P)^\top \in \mathbb{R}^{P \times R}$ and $d_p = D_{pp}$.
3. Initialize partition A.
4. Weighted K-means: While partition A is not stationary,
   a. For all r, $\mu_r = \sum_{p \in A_r} d_p^{1/2} u_p / \sum_{p \in A_r} d_p$
   b. For all p, assign p to A_r where $r = \arg\min_{r'} \|u_p d_p^{-1/2} - \mu_{r'}\|$

Output: partition A, distortion measure $\sum_r \sum_{p \in A_r} d_p \|u_p d_p^{-1/2} - \mu_r\|^2$

Figure 1: Spectral clustering algorithm that minimizes $J_1(W,E)$ with respect to E with weighted K-mean. See Section 2.6 for the initialization of the partition A.

Input: Similarity matrix $W \in \mathbb{R}^{P \times P}$.

Algorithm:

1. Compute first R eigenvectors U of $D^{-1/2}WD^{-1/2}$ where $D = \mathrm{diag}(W1)$.
2. Let $V = D^{1/2}U(U^\top DU)^{-1/2}$
3. Let $V = (v_1, \ldots, v_P)^\top \in \mathbb{R}^{P \times R}$
4. Initialize partition A.
5. Weighted K-means: While partition A is not stationary,
   a. For all r, $\mu_r = \frac{1}{|A_p|} \sum_{p \in A_r} u_p$
   b. For all p, assign p to A_r where $r = \arg\min_{r'} \|u_p - \mu_{r'}\|$

Output: partition A, distortion measure $\sum_r \sum_{p \in A_r} \|u_p - \mu_r\|^2$

Figure 2: Spectral clustering algorithm that minimizes $J_2(W,E)$ with respect to E with K-means. See Section 2.6 for the initialization of the partition A.

Initialization The K-means algorithm can be interpreted as a coordinate descent algorithm and is thus subject to problems of local minima. Thus good initialization is crucial for the practical success of the algorithm in Figure 1.

A similarity matrix W is said *perfect with respect to a partition* E with R clusters if the cost functions $J_1(W,E)$ and $J_2(W,E)$ are exactly equal to zero. This is true in at least two potentially distinct situations: (1) when the matrix W is block-constant, where the block structure follows the partition E, and, as seen earlier, (2) when the matrix W is such that the similarity between points in different clusters is zero, while the similarity between points in the same clusters is strictly positive (Weiss, 1999; Ng et al., 2002).

In both situations, the R cluster centroids are orthogonal vectors, and Ng et al. (2002) have shown that when the similarity matrix is "close" to the second known type of perfect matrices, those centroids are close to orthogonal. This lead to the following natural initialization of the partition A for the K-means algorithm in Figure 1 and Figure 2 (Ng et al., 2002): select a point u_p at random, and successively select $R-1$ points whose directions are most orthogonal to the previously chosen points; then assign each data point to the closest of the R chosen points.

2.7 Variational Formulation for the Normalized Cut

In this section, we show that there is a variational formulation of the normalized cut similar to Theorem 3 for positive semidefinite similarity matrices, that is, for matrices that can be factorized as $W = GG^\top$ where $G \in \mathbb{R}^{P\times M}$, where $M \leqslant P$. Indeed we have the following theorem, whose proof is almost identical to the proof of Theorem 3:

Theorem 5 *If $W = GG^\top$ where $G \in \mathbb{R}^{P\times M}$, then for any partition E, we have:*

$$C(W,E) = \min_{(\mu_1,\ldots,\mu_R)\in\mathbb{R}^{R\times R}} \sum_r \sum_{p\in A_r} d_p \|g_p d_p^{-1} - \mu_r\|^2 + R - \mathrm{tr} D^{-1/2} W D^{-1/2}.$$

This theorem shows that for positive semidefinite matrices, the normalized cut problem is equivalent to the minimization of a weighted distortion measure. However, the dimensionality of the space involved in the distortion measure is equal to the rank of the similarity matrices, and thus can be very large (as large as the number of data points). Consequently, this theorem does not lead straightforwardly to an efficient algorithm for minimizing normalized cuts, since a weighted K-means algorithm in very high dimensions is subject to severe local minima problems (see, for example, Meila and Heckerman, 2001). See Dhillon et al. (2004) for further algorithms based on the equivalence between normalized cuts and weighted K-means.

3. Cost Functions for Learning the Similarity Matrix

Given a similarity matrix W, the steps of a spectral clustering algorithms are (1) normalization, (2) computation of eigenvalues, and (3) partitioning of the eigenvectors using (weighted) K-means to obtain a partition E. In this section, we assume that the partition E is given, and we develop a theoretical framework and a set of algorithms for learning a similarity matrix W.

It is important to note that if if we put no constraints on W, then there is a trivial solution, namely any perfect similarity matrix with respect to the partition E, in particular, any matrix that is block-constant with the appropriate blocks. For our problem to be meaningful, we thus must consider a setting in which there are several data sets to partition and we have a parametric form

for the similarity matrix. The objective is to learn parameters that generalize to unseen data sets with a similar structure. We thus assume that the similarity matrix is a function of a vector variable $\alpha \in \mathbb{R}^F$, and develop a method for learning α.

Given a distance between partitions, a naive algorithm would simply minimize the distance between the true partition E and the output of the spectral clustering algorithm. However, the K-means algorithm that is used to cluster eigenvectors is a non continuous map and the naive cost function would be non continuous and thus hard to optimize. In this section, we first show that the cost function we have presented is an upper bound of the naive cost function; this upper bound has better differentiability properties and is amenable to gradient-based optimization. The function that we obtain is a function of eigensubspaces and we provide numerical algorithms to efficiently minimize such functions in Section 3.3.

3.1 Distance Between Partitions

Let $E = (e_r)_{r=1,\ldots,R}$ and $F = (f_s)_{s=1,\ldots,S}$ be two partitions of P data points with R and S clusters, represented by the indicator matrices of sizes $P \times R$ and $P \times S$, respectively. We use the following distance between the two partitions (Hubert and Arabie, 1985):

$$\begin{aligned} d(E,F) &= \frac{1}{\sqrt{2}} \left\| E(E^\top E)^{-1}E^\top - F(F^\top F)^{-1}F^\top \right\| \\ &= \frac{1}{\sqrt{2}} \left\| \sum_r \frac{e_r e_r^\top}{e_r^\top e_r} - \sum_s \frac{f_s f_s^\top}{f_s^\top f_s} \right\|_F \\ &= \frac{1}{\sqrt{2}} \left(R + S - 2 \sum_{r,s} \frac{(e_r^\top f_s)^2}{(e_r^\top e_r)(f_s^\top f_s)} \right)^{1/2}. \end{aligned} \tag{4}$$

The term $e_r^\top f_s$ simply counts the number of data points which belong to the r-th cluster of E and the s-th cluster of F. The function $d(E,F)$ is a distance for partitions, that is, it is nonnegative and symmetric, it is equal to zero if and only if the partitions are equal, and it satisfies the triangle inequality. Moreover, if F has S clusters and E has R clusters, we have $0 \leqslant d(E,F) \leqslant (\frac{R+S}{2} - 1)^{1/2}$. In simulations, we compare partitions using the squared distance.

3.2 Cost Functions as Upper Bounds

We let $E_1(W)$ denote the clustering obtained by minimizing the cost function $J_1(W,E)$ with respect to E, and let $E_2(W)$ denote the clustering obtained by minimizing the cost function $J_2(W,E)$. The following theorem shows that our cost functions are upper bounds on the distance between a partition and the output of the spectral clustering algorithm:

Theorem 6 *Let* $\eta(W) = \max_p D_{pp} / \min_p D_{pp} \geqslant 1$. *If* $E_1(W) = \arg\min_R J_1(W,E)$ *and* $E_2(W) = \arg\min_R J_2(W,E)$, *then for all partitions* E, *we have:*

$$d(E,E_1(W))^2 \leqslant 4\eta(W) J_1(W,E) \tag{5}$$

$$d(E,E_2(W))^2 \leqslant 4 J_2(W,E). \tag{6}$$

Proof Given a similarity matrix W, following Section 2.4, we have

$$J_2(W,E) = \frac{1}{2} \| V(W)V(W)^\top - E(E^\top E)^{-1}E^\top \|_F^2,$$

where we let denote $V(W) = D^{1/2}U(W)(U(W)^\top DU(W))^{-1/2}$ and $U(W)$ is an orthonormal basis of the R-th principal subspace of $D^{-1/2}WD^{-1/2}$. With that definition, for all partitions E, we have:

$$\begin{aligned} d(E,E_2(W)) &= \frac{1}{\sqrt{2}}\left\|E(E^\top E)^{-1}E^\top - E(W)(E(W)^\top E(W))^{-1}E(W)^\top\right\| \\ &\leqslant \frac{1}{\sqrt{2}}\left\|E(E^\top E)^{-1}E^\top - V(W)V(W)^\top\right\| \\ &\qquad + \frac{1}{\sqrt{2}}\left\|V(W)V(W)^\top - E(W)(E(W)^\top E(W))^{-1}E(W)^\top\right\| \\ &= (J_2(E,W))^{1/2} + \min_E (J_2(E,W))^{1/2} \\ &\leqslant 2(J_2(E,W))^{1/2}, \end{aligned}$$

which proves Eq. (6). In order to prove Eq. (5), we define a distance between partitions that is scaled by D, that is:

$$d_D(E,F) = \frac{1}{\sqrt{2}}\left\|D^{1/2}E(E^\top DE)^{-1}E^\top D^{1/2} - D^{1/2}F(F^\top DF)^{-1}F^\top D^{1/2}\right\|.$$

Following the same steps as above, we can prove:

$$d_D(E,E_1(W)) \leqslant 2(J_1(E,W))^{1/2}.$$

Finally, in order to obtain Eq. (5), we use Lemma 9 in Appendix A. ■

The previous theorem shows that minimizing our cost functions is equivalent to minimizing an upper bound on the true cost function. This bound is tight at zero, consequently, if we are able to produce a similarity matrix W with small $J_1(W,E)$ or $J_2(W,E)$ cost, then the matrix will provably lead to partition that is close to E. Note that the bound in Eq. (5) contains a constant term dependent on W and is thus weaker than the bound in Eq. (6) which does not. In Section 3.4, we compare our cost functions to previously proposed cost functions.

3.3 Functions of Eigensubspaces

Our cost functions, as defined in Eq. (2) and Eq. (3), depend on the R-th principal eigensubspace, that is, the subspace spanned by the first R eigenvectors, $U \in \mathbb{R}^{P\times R}$, of $\widetilde{W} = D^{-1/2}WD^{-1/2}$. In this section, we review classical properties of eigensubspaces, and present optimization techniques to minimize functions of eigensubspaces. In this section, we focus mainly on the cost function $J_1(W,E)$ which is defined in terms of the projections onto the principal subspace of $\widetilde{W} = D^{-1/2}WD^{-1/2}$. The extensions of our techniques to the alternative cost function $J_2(W,E)$ is straightforward. In this section, we first assume that all considered matrices are positive semidefinite, so that all eigenvalues are nonnegative, postponing the treatment of the general case to Section 3.3.5.

3.3.1 PROPERTIES OF EIGENSUBSPACES

Let $\mathcal{M}_{P,R}$ be the set of symmetric matrices such that there is a positive gap between the R-th largest eigenvalue and the $(R+1)$-th largest eigenvalue. The set $\mathcal{M}_{P,R}$ is open (Magnus and Neudecker, 1999), and for any matrix in $\mathcal{M}_{P,R}$, the R-th principal subspace $E_R(M)$ is uniquely defined and the orthogonal projection $\Pi_R(M)$ on that subspace is an unique identifier of that subspace. If $U_R(M)$ is

an orthonormal basis of eigenvectors associated with the R largest eigenvalues, we have $\Pi_R(M) = U_R(M)U_R(M)^\top$, and the value is independent of the choice of the basis $U_R(M)$. Note that the R-th eigensubspace is well defined even if some eigenvalues larger than the R-th eigenvalue coalesce (in which case, the R eigenvectors are not well defined but the R-th principal eigensubspace is).

The computation of eigenvectors and eigenvalues is a well-studied problem in numerical linear algebra (see, for example, Golub and Loan, 1996). The two classical iterative techniques to obtain a few eigenvalues of a symmetric matrix are the *orthogonal iterations* (a generalization of the power method for one eigenvalue) and the *Lanczös method*.

The method of orthogonal iterations starts with a random matrix V in $\mathbb{R}^{P\times R}$, successively multiplies V by the matrix M and orthonormalizes the result with the QR decomposition. For almost all V, the orthogonal iterations converge to the principal eigensubspace, and the convergence is linear with rate $\lambda_{R+1}(M)/\lambda_R(M)$, where $\lambda_1(M) \geqslant \cdots \geqslant \lambda_{R+1}(M)$ are the $R+1$ largest eigenvalues of M. The complexity of performing q steps of the orthogonal iterations is qR times the complexity of the matrix-vector product with the matrix M. If M has no special structure, the complexity is thus $O(qRP^2)$. As discussed in Section 4.4, if special structure is present in M it is possible to reduce this to linear in P. The number of steps to obtain a given precision depends directly on the multiplicative eigengap $\varepsilon_R(M) = \lambda_{R+1}(M)/\lambda_R(M) \leqslant 1$; indeed this number of iterations is $O\left(\frac{1}{1-\varepsilon_R(M)}\right)$.

The Lanczös method is also an iterative method, one which makes better use of the available information to obtain more rapid convergence. Indeed the number of iterations is only $O\left(\frac{1}{(1-\varepsilon_R(M))^{1/2}}\right)$, that is, the square root of the number of iterations for the orthogonal iterations (Golub and Loan, 1996). Note that it is usual to perform subspace iterations on more than the desired number of eigenvalues in order to improve convergence (Bathe and Wilson, 1976).

Finally, in our setting of learning the similarity matrix, we can speed up the eigenvalue computation by initializing the power or Lanczös method with the eigensubspace of previous iterations. Other techniques are also available that can provide a similar speed-up by efficiently tracking the principal subspace of slowly varying matrices (Comon and Golub, 1990; Edelman et al., 1999).

3.3.2 EXACT DIFFERENTIAL

The following proposition shows that the function $\Pi_R(M)$ is continuous and differentiable on $\mathcal{M}_{P,R}$ (for a proof see Appendix B).

Proposition 7 *The function $\Pi_R(M)$, the orthogonal projection on the R-th principal eigensubspace of R, is an infinitely differentiable function on $\mathcal{M}_{P,R}$ and for any differentiable path $M(t)$ of symmetric matrices with values in $\mathcal{M}_{P,R}$ such that $M(0) = M$, the derivative $\frac{d\Pi_R(M(t))}{dt}$ at $t = 0$ is equal to $UN^\top + NU^\top$, where N is the unique solution of the linear system*

$$MN - NU^\top MU = -(I - UU^\top)M'(0)U \text{ and } U^\top N = 0, \tag{7}$$

and where U is any orthonormal basis of the R-th principal subspace of M. The value of the derivative is independent of the chosen orthonormal basis.

The linear system Eq. (7) has PR equations and PR unknowns. It turns out that this system is a positive definite system, with a condition number that is upper bounded by $1/(1-\varepsilon_R(M))$. Thus solving this system using the conjugate gradient method takes a number of iterations proportional to $\frac{1}{(1-\varepsilon_R(M))^{1/2}}$, that is, the complexity of obtaining one derivative is the same as that of computing the first R eigenvectors with the Lanczös method.

Note that if all the first R eigenvalues of M are distinct, the system Eq. (7) decouples into R smaller systems that characterize the differential of a single eigenvector (Magnus and Neudecker, 1999; Cour et al., 2005). However, the condition number of such systems might be very large if some eigenvalues are close. We thus advocate the use of the full system, for which the complexity of each iteration of conjugate gradient is the same complexity as the sum of the decoupled algorithms, but for which the condition number is better behaved. We hereby follow the classical rule of thumb of numerical linear algebra, namely that eigensubspaces are better behaved than individual eigenvectors (Edelman et al., 1999).

3.3.3 APPROXIMATION OF EIGENSUBSPACE AND ITS DIFFERENTIAL

When learning the similarity matrix, the cost function and its derivatives are computed many times and it is thus worthwhile to use an efficient approximation of the eigensubspace as well as its differential. A very natural solution is to stop the iterative methods for computing eigenvectors at a fixed iteration q. The following proposition shows that for the method of power iterations, for almost all starting matrix $V \in \mathbb{R}^{P\times R}$, the projection obtained by early stopping is an infinitely differentiable function:

Proposition 8 *Let $V \in \mathbb{R}^{P\times R}$ be such that $\eta = \max_{u\in E_R(M)^\perp, v\in \mathrm{range}(V)} \cos(u,v) < 1$. Then if we let $V_q(M)$ denote the results of q orthogonal iterations, the function $V_q(M)V_q(M)^\top$ is infinitely differentiable in a neighborhood of M, and we have: $\|V_q(M)V_q(M)^\top - \Pi_R(M)\|_2 \leqslant \frac{\eta}{(1-\eta^2)^{1/2}}(|\lambda_{R+1}(M)|/|\lambda_R(M)|)^q$.*

Proof Golub and Loan (1996) show that for all q, M^qV always has rank R. When only the projection on the column space is sought, the result of the orthogonal iterations does not depend on the chosen method of orthonormalization (usually the QR decomposition), and the final result is theoretically equivalent to orthonormalizing at the last iteration. Thus $V_q(M)V_q(M)^\top = M^qV(V^\top M^{2q}V)^{-1}V^\top M^q$. $V_q(M)V_q(M)^\top$ is C^∞ since matrix inversion and multiplication are C^∞. The bound is proved in Golub and Loan (1996) for the *QR* orthogonal iterations, and since the subspaces computed by the two methods are the same, the bound also holds here. The derivative can easily be computed using the chain rule. ■

Note that numerically taking powers of matrices without care can lead to disastrous results (Golub and Loan, 1996). By using successive *QR* iterations, the computations can be made stable and the same technique can be used for the computation of the derivatives.

3.3.4 POTENTIALLY HARD EIGENVALUE PROBLEMS

In most of the literature on spectral clustering, it is taken for granted that the eigenvalue problem is easy to solve. It turns out that in many situations, the (multiplicative) eigengap is very close to one, making the eigenvector computation difficult (examples are given in the following section).

When the eigengap is close to one, a large power is necessary for the orthogonal iterations to converge. In order to avoid those situations, we regularize the approximation of the cost function based on the orthogonal iterations by a term which is large when the matrix $D^{-1/2}WD^{-1/2}$ is expected to have a small eigengap, and small otherwise. We use the function $n(W) = \mathrm{tr}\,W/\mathrm{tr}\,D$, which is always between 0 and 1, and is equal to 1 when W is diagonal (and thus has no eigengap).

We thus use the cost function defined as follows. Let $V \in \mathbb{R}^{P\times R}$ be defined as $D^{1/2}F$, where the r-th column of F is the indicator matrix of a random subset of the r-th cluster normalized by

the number of points in that cluster. This definition of W ensures that when W is diagonal, the cost function is equal to $R-1$, that is, if the power iterations are likely not to converge, then the value is the maximum possible true value of the cost.

Let $B(W)$ be an approximate orthonormal basis of the projections on the R-th principal subspace of $D^{-1/2}WD^{-1/2}$, based on orthogonal iterations starting from V.[3]

The cost function that we use to approximate $J_1(W,E)$ is

$$F_1(W,E) = \frac{1}{2}\left\|B(W)B(W)^\top - \Pi_0(W,E)\right\|_F^2 - \kappa\log(1-n(W)).$$

We define also $C(W) = D^{1/2}B(W)(B(W)^\top DB(W))^{-1/2}$. The cost function that we use to approximate $J_2(W,E)$ is then

$$F_2(W,E) = \frac{1}{2}\left\|C(W)C(W)^\top - \Pi_0(E)\right\|_F^2 - \kappa\log(1-n(W)).$$

3.3.5 NEGATIVE EIGENVALUES

The spectral relaxation in Proposition 2 involves the largest eigenvalues of the matrix $\widetilde{W} = D^{-1/2}WD^{-1/2}$. The vector $D^{1/2}\mathbf{1}$ is an eigenvector with eigenvalue 1; since we have assumed that W is pointwise nonnegative, 1 is the largest eigenvalue of $\widetilde{W}$. Given any symmetric matrices (not necessarily positive semidefinite) orthogonal iterations will converge to eigensubspaces corresponding to eigenvalues which have largest magnitude, and it may well be the case that some negative eigenvalues of $\widetilde{W}$ have larger magnitude than the largest (positive) eigenvalues, thus preventing the orthogonal iterations from converging to the desired eigenvectors. When the matrix W is positive semidefinite this is not possible. However, in the general case, eigenvalues have to be shifted so that they are all nonnegative. This is done by adding a multiple of the identity matrix to the matrix $\widetilde{W}$, which does not modify the eigenvectors but simply potentially change the signs of the eigenvalues. In our context adding exactly the identity matrix is sufficient to make the matrix positive; indeed, when W is pointwise nonnegative, then both $D+W$ and $D-W$ are *diagonally dominant* with nonnegative diagonal entries, and are thus positive semidefinite (Golub and Loan, 1996), which implies that $-I \preccurlyeq \widetilde{W} \preccurlyeq I$, and thus $I+\widetilde{W}$ is positive semidefinite.

3.4 Empirical Comparisons Between Cost Functions

In this section, we study the ability of the various cost functions we have proposed to track the gold standard error measure in Eq. (4) as we vary the parameter α in the similarity matrix $W_{pp'} = \exp(-\alpha\|x_p - x_{p'}\|^2)$. We study the cost functions $J_1(W,E)$ and $J_2(W,E)$ as well as their approximations based on the power method presented in Section 3.3.3. We also present results for two existing approaches, one based on a Markov chain interpretation of spectral clustering (Meila and Shi, 2002) and one based on the alignment (Cristianini et al., 2002) of $D^{-1/2}WD^{-1/2}$ and Π_0. Our experiment is based on the simple clustering problem shown in Figure 3(a). This apparently simple toy example captures much of the core difficulty of spectral clustering—nonlinear separability and thinness/sparsity of clusters (any point has very few near neighbors belonging to the same cluster, so that the weighted graph is sparse). In particular, in Figure 3(b) we plot the eigengap of the similarity

3. The matrix $D^{-1/2}WD^{-1/2}$ always has the same largest eigenvalue 1 with eigenvector $D^{1/2}\mathbf{1}$ and we could consider instead the $(R-1)$th principal subspace of $D^{-1/2}WD^{-1/2} - D^{1/2}\mathbf{1}\mathbf{1}^\top D^{1/2}/(\mathbf{1}^\top D\mathbf{1})$.

matrix as a function of α, noting that for all optimum values of α, this gap is very close to one, and thus the eigenvalue problem is hard to solve. Worse, for large values of α, the eigengap becomes so small that the eigensolver starts to diverge. It is thus essential to prevent our learning algorithm from yielding parameter settings that lead to a very small eigengap. In Figure 3(e), we plot our approximation of the cost function based on the power method, and we see that, even without the additional regularization presented in Section 3.3.4, our approximate cost function avoids a very small eigengap. The regularization presented in Section 3.3.4 strengthens this behavior.

In Figure 3(c) and (d), we plot the four cost functions against the gold standard. The gold standard curve shows that the optimal α lies above 2.5 on a log scale, and as seen in Figure 3(c) and (e), the minima of the new cost function and its approximation lie among these values. As seen in Figure 3(d), on the other hand, the alignment and Markov-chain-based cost functions show a poor match to the gold standard, and yield minima far from the optimum.

The problem with the latter cost functions is that these functions essentially measure the distance between the similarity matrix W (or a normalized version of W) and a matrix T which (after permutation) is block-diagonal with constant blocks. Spectral clustering does work with matrices which are close to block-constant; however, one of the strengths of spectral clustering is its ability to work effectively with similarity matrices which are not block-constant, and which may exhibit strong variations among each block.

Indeed, in examples such as that shown in Figure 3, the optimal similarity matrix is very far from being block diagonal with constant blocks. Rather, given that data points that lie in the same ring are in general far apart, the blocks are very sparse—not constant and full. Methods that try to find constant blocks cannot find the optimal matrices in these cases. In the language of spectral graph partitioning, where we have a weighted graph with weights W, each cluster is a connected but very sparse graph. The power W^q corresponds to the q-th power of the graph; that is, the graph in which two vertices are linked by an edge if and only if they are linked by a path of length no more than q in the original graph. Thus taking powers can be interpreted as "thickening" the graph to make the clusters more apparent, while not changing the eigenstructure of the matrix (taking powers of symmetric matrices only changes the eigenvalues, not the eigenvectors). Note that other clustering approaches based on taking powers of similarity matrices have been studied by Tishby and Slonim (2001) and Szummer and Jaakkola (2002); these differ from our approach in which we only take powers to approximate the cost function used for learning the similarity matrix.

4. Algorithms for Learning the Similarity Matrix

We now turn to the problem of learning the similarity matrix from data. We assume that we are given one or more sets of data for which the desired clustering is known. The goal is to design a "similarity map," that is, a mapping from data sets of elements in $\mathcal{X}$ to the space of symmetric matrices with nonnegative elements. In this paper, we assume that this space is parameterized. In particular, we consider diagonally-scaled Gaussian kernel matrices (for which the parameters are the scales of each dimension), as well as more complex parameterized matrices for the segmentation of line drawings in Section 4.6 and for speech separation in Section 5. In general we assume that the similarity matrix is a function of a vector variable $\alpha \in \mathbb{R}^F$. We also assume that the parameters are in one-to-one correspondence with the features; setting one of these parameters to zero is equivalent to ignoring the corresponding feature.

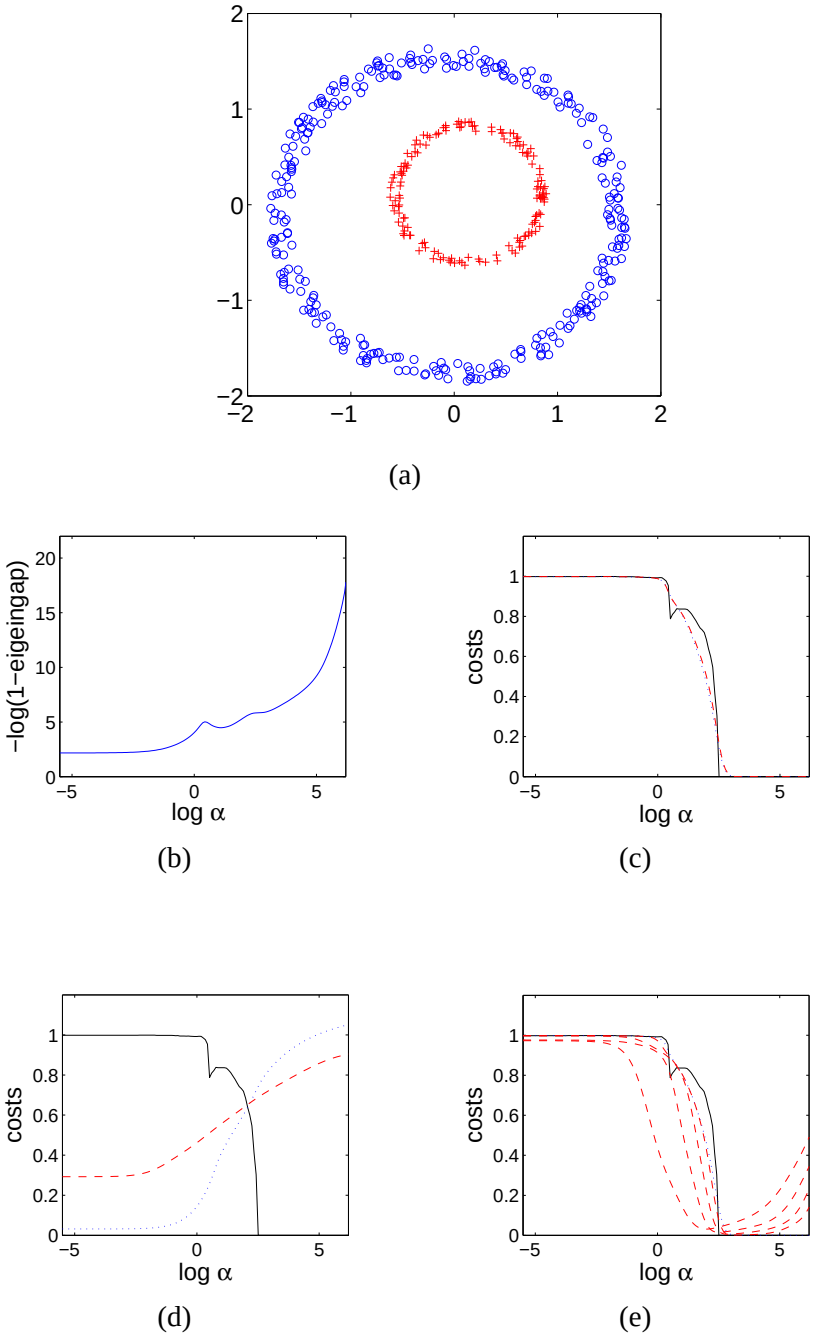


Figure 3: Empirical comparison of cost functions. (a) Data with two clusters (red crosses and blue circles). (b) Eigengap of the similarity matrix as a function of α. (c) Gold standard clustering error (black solid), spectral cost function J_1 (red dotted) and J_2 (blue dashed). (d) Gold standard clustering error (black solid), the alignment (red dashed), and a Markov-chain-based cost, divided by 20 (blue dotted). (e) Approximations based on the power method, with increasing power q: 2 4 16 32.

4.1 Learning Algorithm

We assume that we are given several related data sets with known partitions and our objective is to learn parameters of similarity matrices adapted to the overall problem. This "supervised" setting is not uncommon in practice. In particular, as we show in Section 5, labelled data sets are readily obtained for the speech separation task by artificially combining separately-recorded samples. Note also that in the image segmentation domain, numerous images have been hand-labelled and a data sets of segmented natural images is available (Martin et al., 2001).

More precisely, we assume that we are given N data sets $\mathcal{D}_n$, $n \in \{1,\ldots,N\}$, of points in $\mathcal{X}$. Each data set $\mathcal{D}_n$ is composed of P_n points x_{np}, $p \in \{1,\ldots,P_n\}$. Each data set is segmented; that is, for each n we know the partition E_n. For each n and each α, we have a similarity matrix $W_n(\alpha)$. The cost function that we use is $H(\alpha) = \frac{1}{N}\sum_n F(W_n(\alpha), E_n) + C\sum_{f=1}^{F} |\alpha_f|$. The ℓ_1 penalty serves as a feature selection term, tending to make the solution sparse. The learning algorithm is the minimization of $H(\alpha)$ with respect to $\alpha \in \mathbb{R}^F$, using the method of steepest descent.

Given that the complexity of the cost function increases with q, we start the minimization with small q and gradually increase q up to its maximum value. We have observed that for small q, the function to optimize is smoother and thus easier to optimize—in particular, the long plateaus of constant values are less pronounced. In some cases, we may end the optimization with a few steps of steepest descent using the cost function with the true eigenvectors, that is, for $q = \infty$; this is particularly appropriate when the eigengaps of the optimal similarity matrices happen to be small.

4.2 Related Work

Several other frameworks aim at learning the similarity matrices for spectral clustering or related procedures. Closest to our own work is the algorithm of Cour et al. (2005) which optimizes directly the eigenvectors of the similarity matrix, rather than the eigensubpaces, and is applied to image segmentation tasks. Although differently motivated, the frameworks of Meila and Shi (2002) and Shental et al. (2003) lead to similar convex optimization problems. The framework of Meila and Shi (2002) directly applies to spectral clustering, but we have shown in Section 3.4 that the cost function, although convex, may lead to similarity matrices that do not perform well. The probabilistic framework of Shental et al. (2003) is based on the model granular magnet of Blatt et al. (1997) and applies recent graphical model approximate inference techniques to solve the intractable inference required for the clustering task. Their framework leads to a convex maximum likelihood estimation problem for the similarity parameters, which is based on the same approximate inference algorithms. Among all those frameworks, ours has the advantage of providing theoretical bounds linking the cost function and the actual performance of spectral clustering.

4.3 Testing Algorithm

The output of the learning algorithm is a vector $\alpha \in \mathbb{R}^F$. In order to cluster previously unseen data sets, we compute the similarity matrix W and use the algorithm of Figure 1 or Figure 2. In order to further enhance testing performance, we also adopt an idea due to Ng et al. (2002)—during testing, we vary the parameter α along a direction β. That is, for small λ we set the parameter value to $\alpha + \beta\lambda$ and perform spectral clustering, selecting λ such that the (weighted) distortion obtained after application of the spectral clustering algorithm of Figure 1 or Figure 2 is minimal.

In our situation, there are two natural choices for the direction of search. The first is to use $\beta = \alpha/\|\alpha\|$, that is, we hold fixed the direction of the parameter but allow the norm to vary. This is natural for diagonally-scaled Gaussian kernel matrices. The second solution, which is more generally applicable, is to used the gradient of the individual cost functions, that is, let $G_n = \frac{dF(W_n(\alpha),E_n)}{d\alpha} \in \mathbb{R}^F$. If we neglect the effect of the regularization, at optimality, $\sum_n G_n = 0$. We take the unit-norm direction such that $\sum_n (\beta^\top G_n)^2$ is maximum, which leads to choosing β as the largest eigenvector of $\sum_n G_n G_n^\top$.

4.4 Handling Very Large Similarity Matrices

In applications to vision and speech separation problems, the number of data points to cluster can be enormous: indeed, even a small 256×256 image leads to more than $P = 60,000$ pixels while 3 seconds of speech sampled at 5 kHz leads to more than $P = 15,000$ spectrogram samples. Thus, in such applications, the full matrix W, of size $P \times P$, cannot be stored in main memory. In this section, we present approximation schemes for which the storage requirements are linear in P, for which the time complexity is linear in P, and which enable matrix-vector products to be computed in linear time. See Section 6.3 for an application of each of these methods to speech separation.

For an approximation scheme to be valid, we require that the approximate matrix $\widetilde{W}$ is symmetric, with nonnegative elements, and has a strictly positive diagonal (to ensure in particular that D has a strictly positive diagonal). The first two techniques can be applied generally, while the last method is specific to situations in which there is natural one-dimensional structure, such as in speech or motion segmentation.

4.4.1 SPARSITY

In applications to vision and related problems, most of the similarities are local, and most of the elements of the matrix W are equal to zero. If $Q \leqslant P(P+1)/2$ is the number of elements less than a given threshold (note that the matrix is symmetric so just the upper triangle needs to be stored), the storage requirement is linear in Q, as is the computational complexity of matrix-vector products. However, assessing which elements are equal to zero might take $O(P^2)$. Note that when the sparsity is low, that is, when Q is large, using a sparse representation is unhelpful; only when the sparsity is expected to be high is it useful to consider such an option.

Thus, before attempting to compute all the significant elements (i.e., all elements greater than the threshold) of the matrix, we attempt to ensure that the resulting number of elements Q is small enough. We do so by selecting S random elements of the matrix and estimating from those S elements the proportion of significant elements, which immediately yields an estimate of Q.

If the estimated Q is small enough, we need to compute those Q numbers. However, although the total number of significant elements can be efficiently estimated, the indices of those significant elements cannot be obtained in less than $O(P^2)$ time without additional assumptions. A particular example is the case of diagonally-scaled Gaussian kernel matrices, for which the problem of computing all non-zero elements is equivalent to that of finding pairs of data points in an Euclidean space with distance smaller than a given threshold. We can exploit classical efficient algorithms to perform this task (Gray and Moore, 2001).

If W is an element-wise product of similarity matrices, only a subset of which have a nice structure, we can still use these techniques, albeit with the possibility of requiring more than Q elements of the similarity matrix to be computed.

4.4.2 LOW-RANK NONNEGATIVE DECOMPOSITION

If the matrix W is not sparse, we can approximate it with a low-rank matrix. Following Fowlkes et al. (2001), it is computationally efficient to approximate each column of W by a linear combination of a set of randomly chosen columns: if I is the set of columns that are selected and J is the set of remaining columns, we approximate each column w_j, $j \in J$, as a combination $\sum_{i \in I} H_{ij} w_i$. In the Nyström method of Fowlkes et al. (2001), the coefficient matrix H is chosen so that the squared error on the rows in indexed by I is minimum, that is, H is chosen so that $\sum_{k \in I} (w_j(k) - \sum_{i \in I} H_{ij} w_i(k))^2$. Since W is symmetric, this only requires knowledge of the columns indexed by I. The solution of this convex quadratic optimization problem is simply $H = W(I,I)^{-1} W(I,J)$, where for any sets A and B of distinct indices $W(A,B)$ is the (A,B) block of W. The resulting approximating matrix is symmetric and has a rank equal to the size of I.

When the matrix W is positive semidefinite, then the approximation remains positive semidefinite. However, when the matrix W is element-wise nonnegative, which is the main assumption in this paper, then the approximation might not be and this may lead to numerical problems when applying the techniques presented in this paper. In particular the approximated matrix D might not have a strictly positive diagonal. The following low-rank nonnegative decomposition has the advantage of retaining a pointwise nonnegative decomposition, while being only slightly slower. We use this decomposition in order to approximate the large similarity matrices, and the required rank is usually in the order of hundreds; this is to be contrasted with the approach of Ding et al. (2005), which consists in performing a nonnegative decomposition with very few factors in order to potentially obtain directly cluster indicators.

We first find the best approximation of $A = W(I,J)$ as VH, where $V = W(I,I)$ and H is element-wise nonnegative. This can be done efficiently using algorithms for nonnegative matrix factorization (Lee and Seung, 2000). Indeed, starting from a random positive H, we perform the following iteration until convergence:

$$\forall i, j,\ H_{ij} \leftarrow \frac{\sum_k V_{ki} A_{kj} / (VH)_{kj}}{\sum_k V_{ki}}. \tag{8}$$

The complexity of the iteration in Eq. (8) is $O(M^2 P)$, and empirically we usually find that we require a small number of iterations before reaching a sufficiently good solution. Note that the iteration yields a monotonic decrease in the following divergence:

$$D(A||VH) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{(VH)_{ij}} - A_{ij} + (VH)_{ij} \right).$$

We approximate $W(J,J)$ by symmetrization,[4] that is, $W(J,I)H + H^\top W(I,J)$. In order to obtain a better approximation, we ensure that the diagonal of $W(J,J)$ is always used with its true (i.e., not approximated) value. Note that the matrices H found by nonnegative matrix factorization are usually sparse.

The storage requirement is $O(MP)$, where M is the number of selected columns. The complexity of the matrix-vector products is $O(MP)$. Empirically, the average overall complexity of obtaining the decomposition is $O(M^2 P)$.

4. For a direct low-rank symmetric nonnegative decomposition algorithm, see Ding et al. (2005).

4.4.3 LOW-RANK BAND DECOMPOSITION

There are situations in between the two previous cases, that is, the matrix W is not sparse enough and it cannot be well approximated by a low-rank matrix. When there is a natural one-dimensional structure, such as in audio or video, then we can use the potential "bandedness" structure of W. The matrix W is referred to as *band-diagonal with bandwidth B*, if for all i, j, $|i-j| \geqslant B \Rightarrow W_{ij} = 0$. The matrix has then at most $O(BP)$ non zero elements.

We can use the bandedness of the problem to allow the rank M to grow linearly with P, while retaining linear time complexity. We assume that the M columns are sampled uniformly. Let $C = P/M$ be the average distance between two successive sampled columns. For the low-rank approximation to make sense, we require that $B \gg C$ (otherwise, $W(I,I)$ is close to being diagonal and carries no information). Moreover, the approximation is only useful if $M \ll P$, that is, the rank M is significantly smaller than P, which leads to the requirement $C \gg 1$. This approximating scheme is thus potentially useful when C is between 1 and B.

For the Nyström technique, if the sampling of columns of I is uniform, then $W(I,I)$ is expected to be band-diagonal with bandwidth B/C, and thus inverting $W(I,I)$ takes time $O(M(B/C)^2) = O(B^2/C^3 \times P)$. The inverse is also band-diagonal with the same bandwidth. The storage and the matrix multiplications are then $O(B/C \times P)$, that is, everything is linear in P, while the rank M is allowed to grow with P.

In summary, if we require a nonnegative decomposition, it is possible to adapt the iteration Eq. (8) using band matrix techniques, to obtain a linear complexity in P, even though the rank M grows with P.

4.5 Simulations on Toy Examples

We performed simulations on synthetic data sets involving two-dimensional data sets similar to that shown in Figure 3, where there are two rings whose relative distance is constant across samples (but whose relative orientation has a random direction). We add D irrelevant dimensions of the same magnitude as the two relevant variables. The goal is thus to learn the diagonal scale $\alpha \in \mathbb{R}^{D+2}$ of a Gaussian kernel that leads to the best clustering on unseen data. We learn α from N sample data sets ($N=1$ or $N=10$), and compute the clustering error of our algorithm with and without adaptive tuning of the norm of α during testing (cf. Section 4.3) on ten previously unseen data sets. We compare to an approach that does not use the training data: α is taken to be the vector of all ones and we again search over the best possible norm during testing (we refer to this method as "no learning"). We report results in Table 1. Without feature selection, the performance of spectral clustering degrades very rapidly when the number of irrelevant features increases, while our learning approach is very robust, even with only one training data set.

4.6 Simulations on Line Drawings

In this section, we consider the problem of segmenting crossing line drawings in the plane. In Section 4.6.1 we describe the features that we used. Section 4.6.2 discusses the construction of the parameterized similarity matrices and Section 4.6.3 presents our experimental results. The general setup of the experiments is that we learn the parameters on a training set of images and test on unseen images. In one of the experiments, we focus on drawings whose segmentations are ambiguous; in this case we learn two different parameterized similarity matrices with two different sets of hand-

D	no learning	learning w/o tuning		learning with tuning	
		N=1	N=10	N=1	N=10
0	0	15.5	10.5	0	0
1	60.8	37.7	9.5	0	0
2	79.8	36.9	9.5	0	0
4	99.8	37.8	9.7	0.4	0
8	99.8	37	10.7	0	0
16	99.7	38.8	10.9	14	0
32	99.9	38.9	15.1	14.6	6.1

Table 1: Performance on synthetic data sets: clustering errors (multiplied by 100) for method without learning (but with tuning) and for our learning method with and without tuning, with $N = 1$ or 10 training data sets; D is the number of irrelevant features.

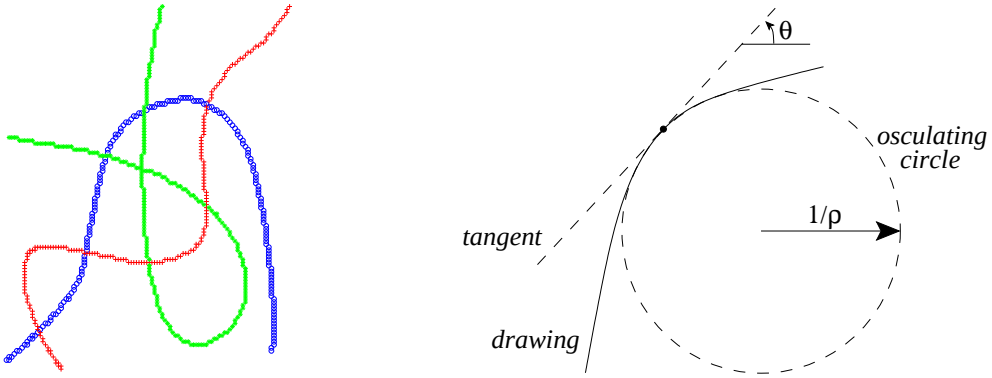


Figure 4: (Left) example of segmented drawing, (Right) tangent and osculating circle.

labelings reflecting the ambiguous segmentation. The goal is then to see if we can disambiguate the test images.

4.6.1 FEATURES FOR HAND DRAWINGS

We represent hand drawings as a two-dimensional image which is obtained by the binning of a continuous drawing. Each drawing is thus represented as an $N_x \times N_y$ binary image. See Figure 4 for an example. In order to segment the drawings, we estimate, at each inked point, the direction and relative curvature. This is done by convolving the drawing with patches of quarters of circles of varying angles and curvature. In simulations, we use 50 different angles and 50 different curvatures. We thus obtain, for each inked point, a score for each angle and each curvature. We take as a feature the angle θ and curvature ρ that attains the maximum score. We also keep the log of the ratio of the maximum score to the median score; this value, denoted c, is an estimate of the confidence of the estimate of θ and ρ.

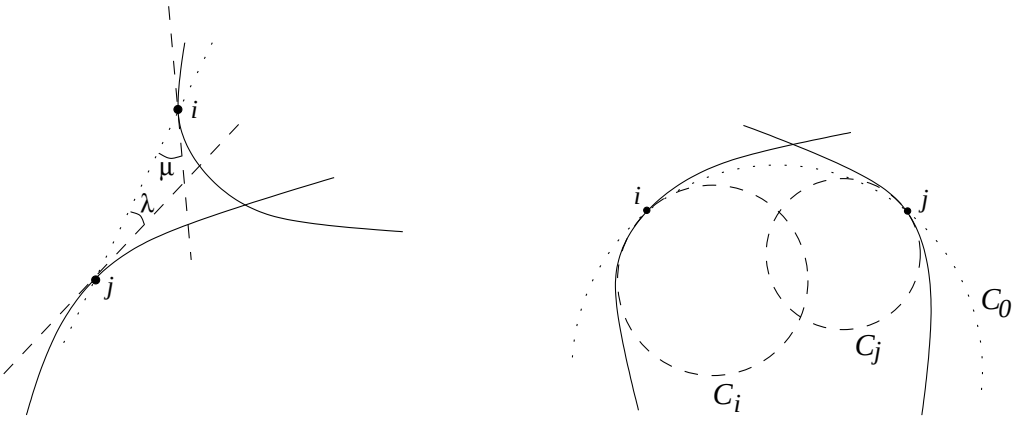


Figure 5: (Left) First pairwise feature, equal to the product of the magnitude of the cosines of angles μ and λ, (Right) Second pairwise features, built from the two osculating circles C_i and C_j, and all circles that go through point i and j.

4.6.2 PARAMETERIZED SIMILARITY MATRICES FOR HAND DRAWINGS

For a given image, we need to build a matrix that contains the pairwise similarities of all the inked points. Let P be the number of inked points in a given drawing. For $i = 1,\ldots,P$, we have the following features for each inked point: the coordinates (x_i, y_i), the angle of the direction of the tangent θ_i (note that directions are defined modulo π), and the curvature ρ_i, as well as the estimated confidence c_i.

We also build "pairwise features"; in particular we build two specialized similarity matrices that are based on the geometry of the problem. Given two points of the same image indexed by i and j, with features $(x_i, y_i, \theta_i, \rho_i)$ and $(x_j, y_j, \theta_j, \rho_j)$, the first feature is defined as

$$a_{ij} = \frac{|(x_j - x_i)\cos\theta_i + (y_j - y_i)\sin\theta_i| \times |(x_j - x_i)\cos\theta_j + (y_j - y_i)\sin\theta_j|}{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

The feature a_{ij} is symmetric and is always between zero and one, and is equal to the product of the cosines between the two tangents and the chord that links the two points. (See the left panel of Figure 5). The feature a_{ij} is equal to one if the tangents are both parallel to the chord, and the two points are then likely to belong to the same cluster.

The second pairwise feature characterizes how well the two osculating circles at point i and j match. We consider all circles that go through points i and points j, and we compute the products of metrics between C_0 and C_j, and C_0 and C_i. The maximum (over all circles C_0) possible metric is chosen as the feature. The metric between two circles that intersect in two points is defined as the product of the cosines of the angles between the tangents at those two points times a Gaussian function of the difference in curvature. The measure b_{ij} is also symmetric and always between zero and one; it characterizes how well a circle can be fit to the two local circles defined by the two sets of features.

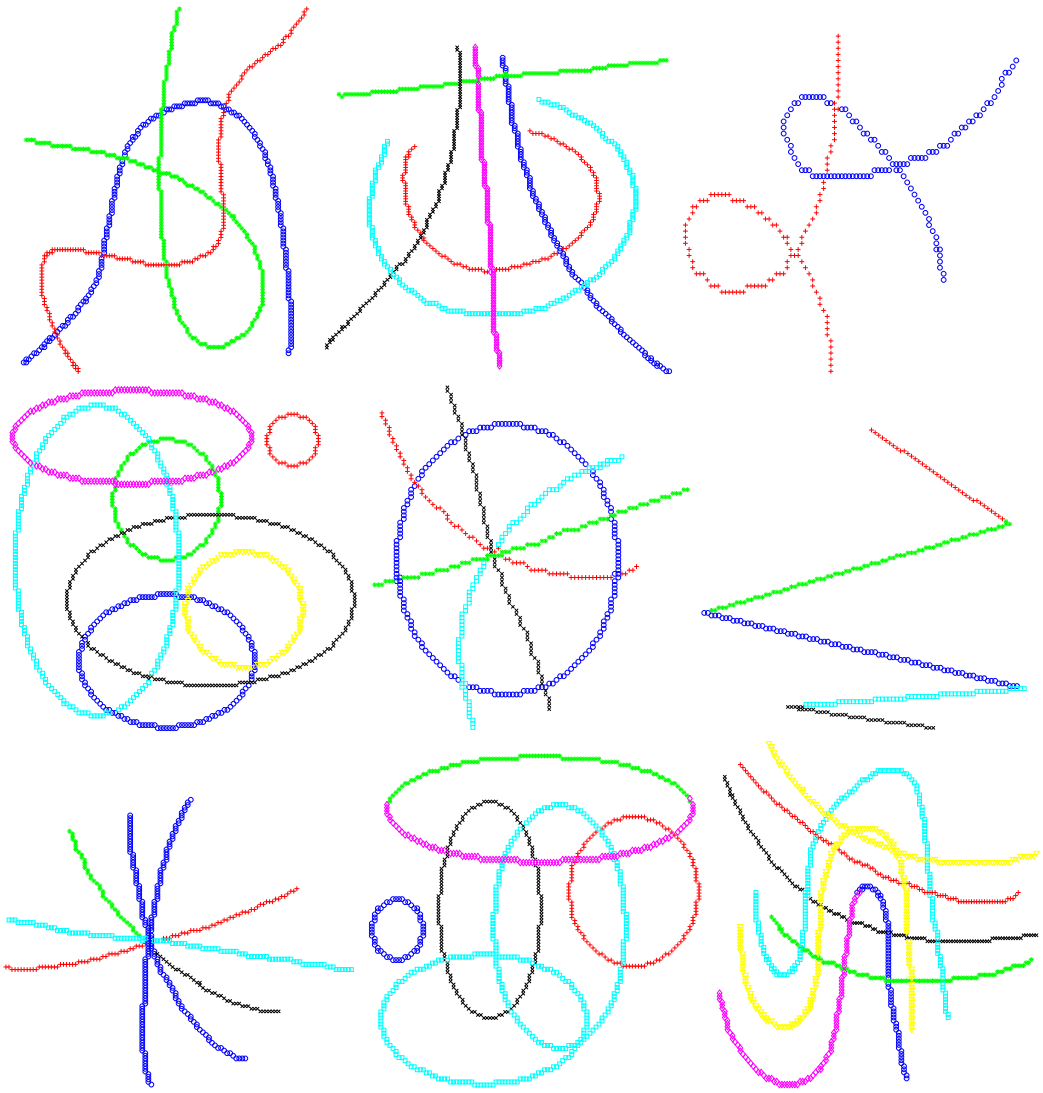


Figure 6: Segmentation results after learning the parameterized similarity matrices. The first two rows are correctly segmented examples while the third row shows examples with some mistakes.

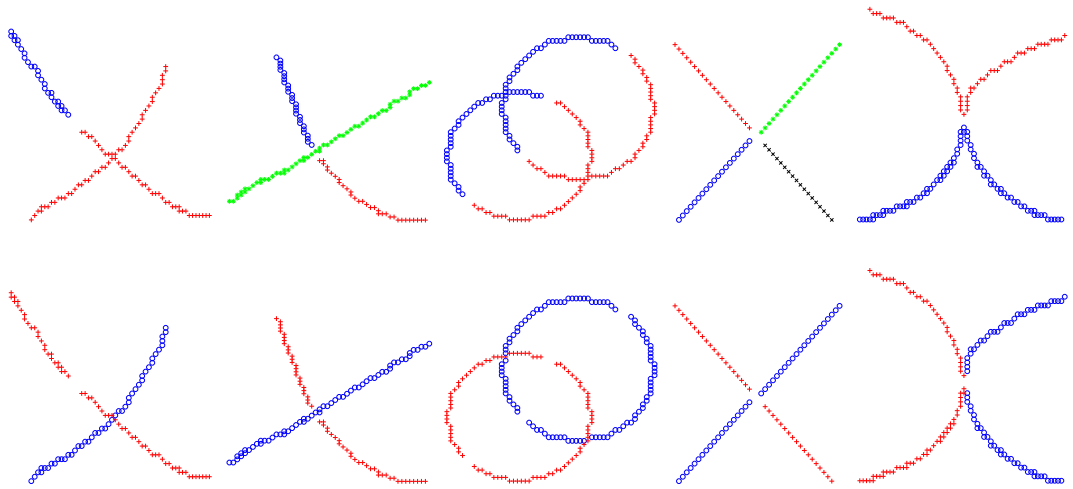


Figure 7: Subsets of training data sets for ambiguous line drawings: (top) first data set, favoring connectedness, (bottom) second data set, favoring direction continuity.

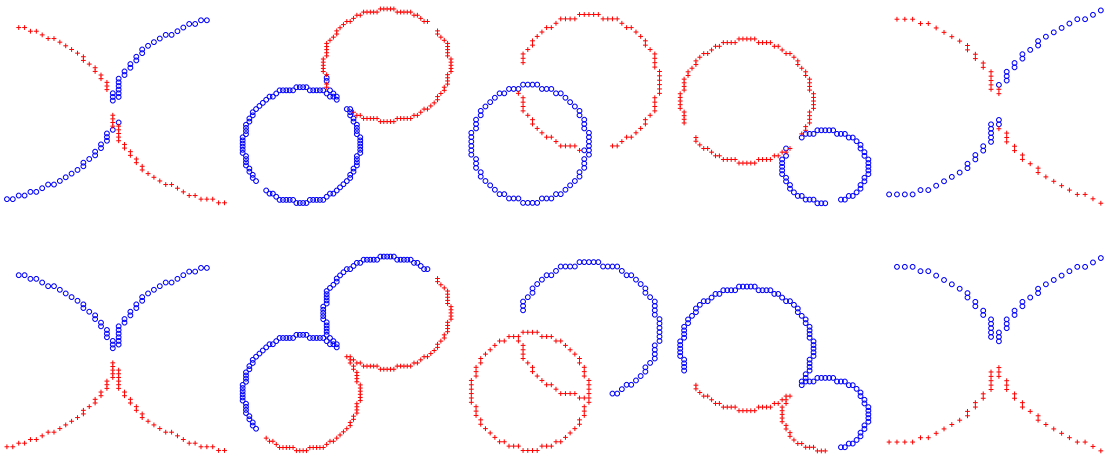


Figure 8: Testing examples: (top) obtained from parameterized similarity matrices learned using the top row of Figure 7 for training, (bottom) obtained from parameterized similarity matrices learned using the bottom row of Figure 7 for training.

The pairwise similarity that we use is thus:

$$\begin{aligned} -\log W_{ij} &= \alpha_1(x_i - x_i)^2 + \alpha_1(y_i - y_i)^2 + \alpha_3(\cos 2\theta_i - \cos 2\theta_i)^2 + \alpha_4(\sin 2\theta_i - \sin 2\theta_i)^2 \\ &\quad + \alpha_5(c_i - c_j)^2 + \alpha_6(|\rho_i| - |\rho_j|)^2 - \alpha_7 \log a_{ij} - \alpha_8 \log b_{ij}. \end{aligned}$$

4.6.3 SIMULATIONS

In a first experiment, we learned the parameters of the similarity matrices on 50 small to medium images with two clusters and tested on various images with varying size and varying number of clusters. In these simulations, the desired number of clusters was always given. See Figure 6 for examples in which segmentation was successful and examples in which the similarity matrices failed to lead to the proper segmentation.

In a second experiment, we used two different training data sets with the same drawings, but with different training partitions. We show some examples in Figure 7. In the first data set, connectedness of a single cluster was considered most important by the human labeller, while in the second data set, continuity of the direction was the main factor. We then tested the two estimated parameterized similarity matrices on ambiguous line drawings and we show some results in Figure 8.

5. Speech Separation as Spectrogram Segmentation

The problem of recovering signals from linear mixtures, with only partial knowledge of the mixing process and the signals—a problem often referred to as *blind source separation*—is a central problem in signal processing. It has applications in many fields, including speech processing, network tomography and biomedical imaging (Hyvärinen et al., 2001). When the problem is over-determined, that is, when there are no more signals to estimate (the sources) than signals that are observed (the sensors), generic assumptions such as statistical independence of the sources can be used in order to demix successfully (Hyvärinen et al., 2001). Many interesting applications, however, involve under-determined problems (more sources than sensors), where more specific assumptions must be made in order to demix. In problems involving at least two sensors, progress has been made by appealing to sparsity assumptions (Zibulevsky et al., 2002; Jourjine et al., 2000).

However, the most extreme case, in which there is only one sensor and two or more sources, is a much harder and still-open problem for complex signals such as speech. In this setting, simple generic statistical assumptions do not suffice. One approach to the problem involves a return to the spirit of classical engineering methods such as matched filters, and estimating specific models for specific sources—for example, specific speakers in the case of speech (Roweis, 2001; Jang and Lee, 2003). While such an approach is reasonable, it departs significantly from the desideratum of "blindness." In this section we present an algorithm that is a blind separation algorithm—our algorithm separates speech mixtures from a single microphone without requiring models of specific speakers.

Our approach involves a "discriminative" approach to the problem of speech separation that is based on the spectral learning methodology presented in Section 4. That is, rather than building a complex model of speech, we instead focus directly on the task of separation and optimize parameters that determine separation performance. We work within a time-frequency representation (a spectrogram), and exploit the sparsity of speech signals in this representation. That is, although two speakers might speak simultaneously, there is relatively little overlap in the time-frequency plane if the speakers are different (Roweis, 2001; Jourjine et al., 2000). We thus formulate speech separation as a problem in segmentation in the time-frequency plane. In principle, we could appeal to classical segmentation methods from vision (see, for example, Shi and Malik, 2000) to solve this two-dimensional segmentation problem. Speech segments are, however, very different from visual segments, reflecting very different underlying physics. Thus we must design features for segmenting speech from first principles.

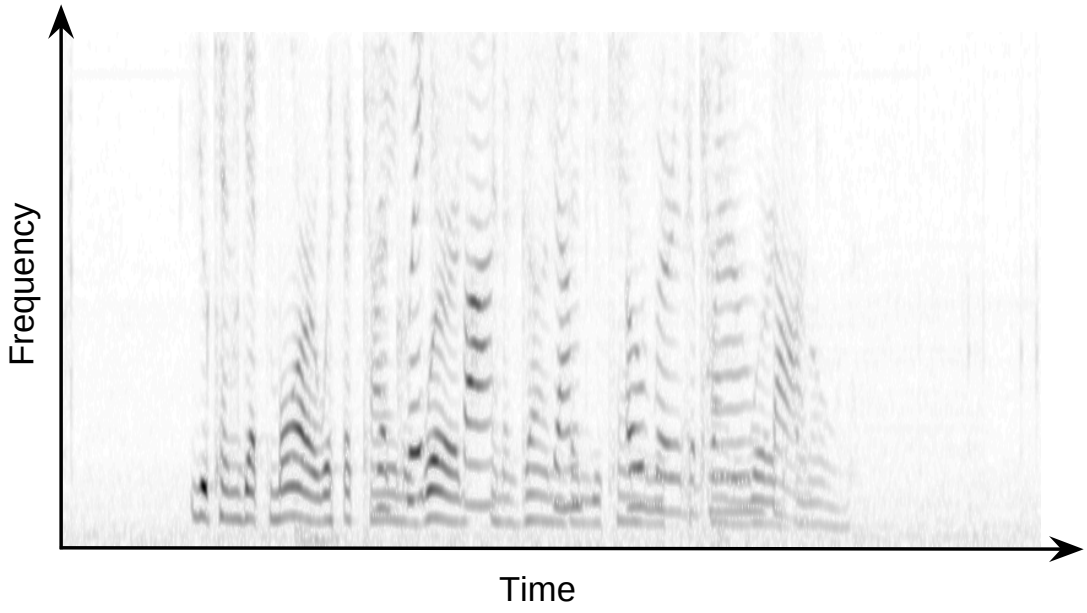


Figure 9: Spectrogram of speech (two simultaneous English speakers). The gray intensity is proportional to the amplitude of the spectrogram.

5.1 Spectrogram

The spectrogram is a two-dimensional (time and frequency) redundant representation of a one-dimensional signal (Mallat, 1998). Let $f[t], t = 0, \ldots, T-1$ be a signal in $\mathbb{R}^T$. The spectrogram is defined via windowed Fourier transforms and is commonly referred to as a short-time Fourier transform or as Gabor analysis (Mallat, 1998). The value $(Uf)_{mn}$ of the spectrogram at time window n and frequency m is defined as $(Uf)_{mn} = \frac{1}{\sqrt{M}} \sum_{t=0}^{T-1} f[t] w[t-na] e^{i2\pi mt/M}$, where w is a window of length T with small support of length c, and $M \geqslant c$. We assume that the number of samples T is an integer multiple of a and c. There are then $N = T/a$ different windows of length c. The spectrogram is thus an $N \times M$ image which provides a redundant time-frequency representation of time signals[5] (see Figure 9).

Inversion Our speech separation framework is based on the segmentation of the spectrogram of a signal $f[t]$ in $R \geqslant 2$ disjoint subsets A_i, $i = 1, \ldots, R$ of $[0, N-1] \times [0, M-1]$. This leads to R spectrograms U_i such that $(U_i)_{mn} = U_{mn}$ if $(m,n) \in A_i$ and zero otherwise. We now need to find R speech signals $f_i[t]$ such that each U_i is the spectrogram of f_i. In general there are no exact solutions (because the representation is redundant), and a classical technique is to find the minimum ℓ_2 norm approximation, that is, find f_i such that $\|U_i - Uf_i\|^2$ is minimal (Mallat, 1998). The solution of this minimization problem involves the pseudo-inverse of the linear operator U (Mallat, 1998) and

5. In our simulations, the sampling frequency is $f_0 = 5.5$ kHz and we use a Hanning window of length $c = 216$ (i.e., 43.2 ms). The spacing between window is equal to $a = 54$ (i.e., 10.8 ms). We use a 512-point FFT ($M = 512$). For a speech sample of length 4 seconds, we have $T = 22,000$ samples and then $N = 407$, which yields $\approx 2 \times 10^5$ spectrogram samples.

is equal to $f_i = (U^*U)^{-1}U^*U_i$, where U^* is the (complex) adjoint of the linear operator U. By our choice of window (Hanning), U^*U is proportional to the identity matrix, so that the solution to this problem can simply be obtained by applying the adjoint operator U^*. Other techniques for spectrogram inversion could be used (Griffin and Lim, 1984; Mallat, 1998; Achan et al., 2003)

5.2 Normalization and Subsampling

There are several ways of normalizing a speech signal. In this paper, we chose to rescale all speech signals as follows: for each time window n, we compute the total energy $e_n = \sum_m |U f_{mn}|^2$, and its 20-point moving average. The signals are normalized so that the 90th percentile of those values is equal to one.

In order to reduce the number of spectrogram samples to consider, for a given pre-normalized speech signal, we threshold coefficients whose magnitudes are less than a value that was chosen so that the resulting distortion is inaudible.

5.3 Generating Training Samples

Our approach is based on the learning algorithm presented in Section 4. The training examples that we provide to this algorithm are obtained by mixing separately-normalized speech signals. That is, given two volume-normalized speech signals, f_1 and f_2, of the same duration, with spectrograms U_1 and U_2, we build a training sample as $U^{train} = U_1 + U_2$, with a segmentation given by $z = \arg\min\{U_1, U_2\}$. In order to obtain better training partitions (and in particular to be more robust to the choice of normalization), we also search over all $\alpha \in [0,1]$ such that the ℓ_2 reconstruction error obtained from segmenting/reconstructing using $z = \arg\min\{\alpha U_1, (1-\alpha)U_2\}$ is minimized. An example of such a partition is shown in Figure 10 (top).

5.4 Features and Grouping Cues for Speech Separation

In this section we describe our approach to the design of features for the spectral segmentation. We base our design on classical cues suggested from studies of perceptual grouping (Cooke and Ellis, 2001). Our basic representation is a "feature map," a two-dimensional representation that has the same layout as the spectrogram. Each of these cues is associated with a specific time scale, which we refer to as "small" (less than 5 frames), "medium" (10 to 20 frames), and "large" (across all frames). (These scales will be of particular relevance to the design of numerical approximation methods in Section 6.3). Any given feature is not sufficient for separating by itself; rather, it is the combination of several features that makes our approach successful.

5.4.1 NON-HARMONIC CUES

The following non-harmonic cues have counterparts in visual scenes and for these cues we are able to borrow from feature design techniques used in image segmentation (Shi and Malik, 2000).

- **Continuity** Two time-frequency points are likely to belong to the same segment if they are close in time or frequency; we thus use time and frequency directly as features. This cue acts at a small time scale.

- **Common fate cues** Elements that exhibit the same time variation are likely to belong to the same source. This takes several particular forms. The first is simply *common offset* and *com-*

mon onset. We thus build an offset map and an onset map, with elements that are zero when no variation occurs, and are large when there is a sharp decrease or increase (with respect to time) for that particular time-frequency point. The onset and offset maps are built using oriented energy filters as used in vision (with one vertical orientation). These are obtained by convolving the spectrogram with derivatives of Gaussian windows (Shi and Malik, 2000).

Another form of the common fate cue is *frequency co-modulation*, the situation in which frequency components of a single source tend to move in sync. To capture this cue we simply use oriented filter outputs for a set of orientation angles (8 in our simulations). Those features act mainly at a medium time scale.

5.4.2 HARMONIC CUES

This is the major cue for voiced speech (Gold and Morgan, 1999; Brown and Cooke, 1994; Bregman, 1990), and it acts at all time scales (small, medium and large): voiced speech is locally periodic and the local period is usually referred to as the pitch.

- **Pitch estimation** In order to use harmonic information, we need to estimate potentially several pitches. We have developed a simple pattern matching framework for doing this that we present in Appendix C. If S pitches are sought, the output that we obtain from the pitch extractor is, for each time frame n, the S pitches $\omega_{n1}, \ldots, \omega_{nS}$, as well as the strength y_{nms} of the s-th pitch for each frequency m.

- **Timbre** The pitch extraction algorithm presented in Appendix C also outputs the spectral envelope of the signal (Gold and Morgan, 1999). This can be used to design an additional feature related to timbre which helps integrate information regarding speaker identification across time. Timbre can be loosely defined as the set of properties of a voiced speech signal once the pitch has been factored out (Bregman, 1990). We add the spectral envelope as a feature (reducing its dimensionality using principal component analysis).

5.4.3 BUILDING FEATURE MAPS FROM PITCH INFORMATION

We build a set of features from the pitch information. Given a time-frequency point (m,n), let $s(m,n) = \arg\max_s \frac{y_{nms}}{(\sum_{m'} y_{nm's})^{1/2}}$ denote the highest energy pitch, and define the features $\omega_{ns(m,n)}$, $y_{nms(m,n)}$, $\sum_{m'} y_{nm's(m,n)}$, $\frac{y_{nms(m,n)}}{\sum_{m'} y_{nm's(m,n)}}$ and $\frac{y_{nms(m,n)}}{(\sum_{m'} y_{nm's(m,n)}))^{1/2}}$. We use a partial normalization with the square root to avoid including very low energy signals, while allowing a significant difference between the local amplitude of the speakers.

Those features all come with some form of energy level and all features involving pitch values ω should take this energy into account when the similarity matrix is built in Section 6. Indeed, this value has no meaning when no energy in that pitch is present.

6. Spectral Clustering for Speech Separation

Given the features described in the previous section, we now show how to build similarity matrices that can be used to define a spectral segmenter. In particular, our approach builds *parameterized* similarity matrices, and uses the learning algorithm presented in Section 4 to adjust these parameters.

6.1 Basis Similarity Matrices

We define a set of "basis similarity" matrices for each set of cues and features defined in Section 5.4. Those basis matrices are then combined as described in Section 6.2 and the weights of this combination are learned as shown in Section 4.

For non-harmonic features, we use a radial basis function to define affinities. Thus, if f_a is the value of the feature for data point a, we use a basis similarity matrix defined as $W_{ab} = \exp(-\|f_a - f_b\|^2)$. For a harmonic feature, on the other hand, we need to take into account the strength of the feature: if f_a is the value of the feature for data point a, with strength y_a, we use $W_{ab} = \exp(-\min\{y_a, y_b\}\|f_a - f_b\|^2)$.

6.2 Combination of Similarity Matrices

Given m basis matrices, we use the following parameterization of W: $W = \sum_{k=1}^{K} \gamma_k W_1^{\alpha_{j1}} \times \cdots \times W_m^{\alpha_{jm}}$, where the products are taken pointwise. Intuitively, if we consider the values of similarity as soft boolean variables, taking the product of two similarity matrices is equivalent to considering the conjunction of two matrices, while taking the sum can be seen as their disjunction. For our application to speech separation, we consider a sum of $K = 2$ matrices. This has the advantage of allowing different approximation schemes for each of the time scales, an issue we address in the following section.

6.3 Approximations of Similarity Matrices

The similarity matrices that we consider are huge, of size at least $50,000 \times 50,000$. Thus a significant part of our effort has involved finding computationally efficient approximations of similarity matrices.

Let us assume that the time-frequency plane is vectorized by stacking one time frame after the other. In this representation, the time scale of a basis similarity matrix W exerts an effect on the degree of "bandedness" of W. Recall that the matrix W is referred to as band-diagonal with bandwidth B, if for all i, j, $|i - j| \geqslant B \Rightarrow W_{ij} = 0$. On a small time scale, W has a small bandwidth; for a medium time scale, the band is larger but still small compared to the total size of the matrix, while for large scale effects, the matrix W has no band structure. Note that the bandwidth B can be controlled by the coefficient of the radial basis function involving the time feature n.

For each of these three cases, we have designed a particular way of approximating the matrix, while ensuring that in each case the time and space requirements are *linear* in the number of time frames, and thus linear in the duration of the signal to demix.

- **Small scale** If the bandwidth B is very small, we use a simple direct sparse approximation. The complexity of such an approximation grows linearly in the number of time frames.

- **Medium and large scale** We use a low-rank approximation of the matrix W, as presented in Section 4.4. For mid-range interactions, we need an approximation whose rank grows with time, but whose complexity does not grow quadratically with time (see Section 4.4), while for large scale interactions, the rank is held fixed.

	Bound	Clust	Pitch	Freq
English (SNR)	2.3%	6.9%	31.1%	33.4%
English (SNR_{dB})	16.4	11.6	5.1	4.8
French (SNR)	3.3%	15.8%	35.4%	40.7%
French (SNR_{dB})	14.8	8.0	4.5	3.9

Table 2: Comparison of signal-to-noise ratios.

6.4 Experiments

We have trained our segmenter using data from four different male and female speakers, with speech signals of duration 3 seconds. There were 15 parameters to estimate using our spectral learning algorithm. For testing, we use mixes from speakers which were different from those in the training set.

In Figure 10, for two English speakers from the testing set, we show an example of the segmentation that is obtained when the two speech signals are known in advance (top panel), a segmentation that would be used for training our spectral clustering algorithm, and in the bottom panel, the segmentation that is output by our algorithm.

Although some components of the "black" speaker are missing, the segmentation performance is good enough to obtain audible signals of reasonable quality. The speech samples for these examples can be downloaded from `http://cmm.ensmp.fr/~bach/speech/`. On this web site, there are several additional examples of speech separation, with various speakers, in French and in English. Similarly, we present in Figure 11, segmentation results for French speakers. Note that the same parameters were used for both languages and that the two languages were present in the training set. An important point is that our method does not require knowing the speakers in advance in order to demix successfully; rather, it is only necessary that the two speakers have distinct pitches most of the time (another but less crucial condition is that one pitch is not too close to twice the other one).

A complete evaluation of the robustness of our approach is outside the scope of this paper; however, for the two examples shown in Figure 10 and Figure 11, we can compare signal-to-noise ratios for various competing approaches. Given the true signal s (known in our simulation experiments) and an estimated signal $\hat{s}$, the signal-to-noise ratio (SNR) is defined as $SNR = \frac{\|s-\hat{s}\|^2}{\|s\|^2}$, and is often reported in decibels, as $SNR_{dB} = -10\log_{10}\frac{\|s-\hat{s}\|^2}{\|s\|^2}$. In order to characterize demixing performance, we use the maximum of the signal-to-noise ratios between the two true signals and the estimated signals (potentially after having permuted the estimated signals). In Table 2, we compare our approach ("Clust"), with the demixing solution obtained from the segmentation that would serve for training purposes ("Bound") (this can be seen as an upper bound on the performance of our approach). We also performed two baseline experiments: (1) In order to show that the combination of features is indeed crucial for performance, we performed K-means clustering on the estimated pitch to separate the two signals ("Pitch"). (2) In order to show that a full time-frequency approach is needed, and not simply frequency-based filtering, we used Wiener filters computed from the true signals ("Freq"). Note that to compute the four SNRs, the "Pitch" and "Freq" methods need the true signals, while the two other methods ("Clust" and "Bound") are pure separating approaches.

From the results in Table 2, we see that pitch alone is not sufficient for successful demixing (see the third column in the table). This is presumably due in part to the fact that pitch is not

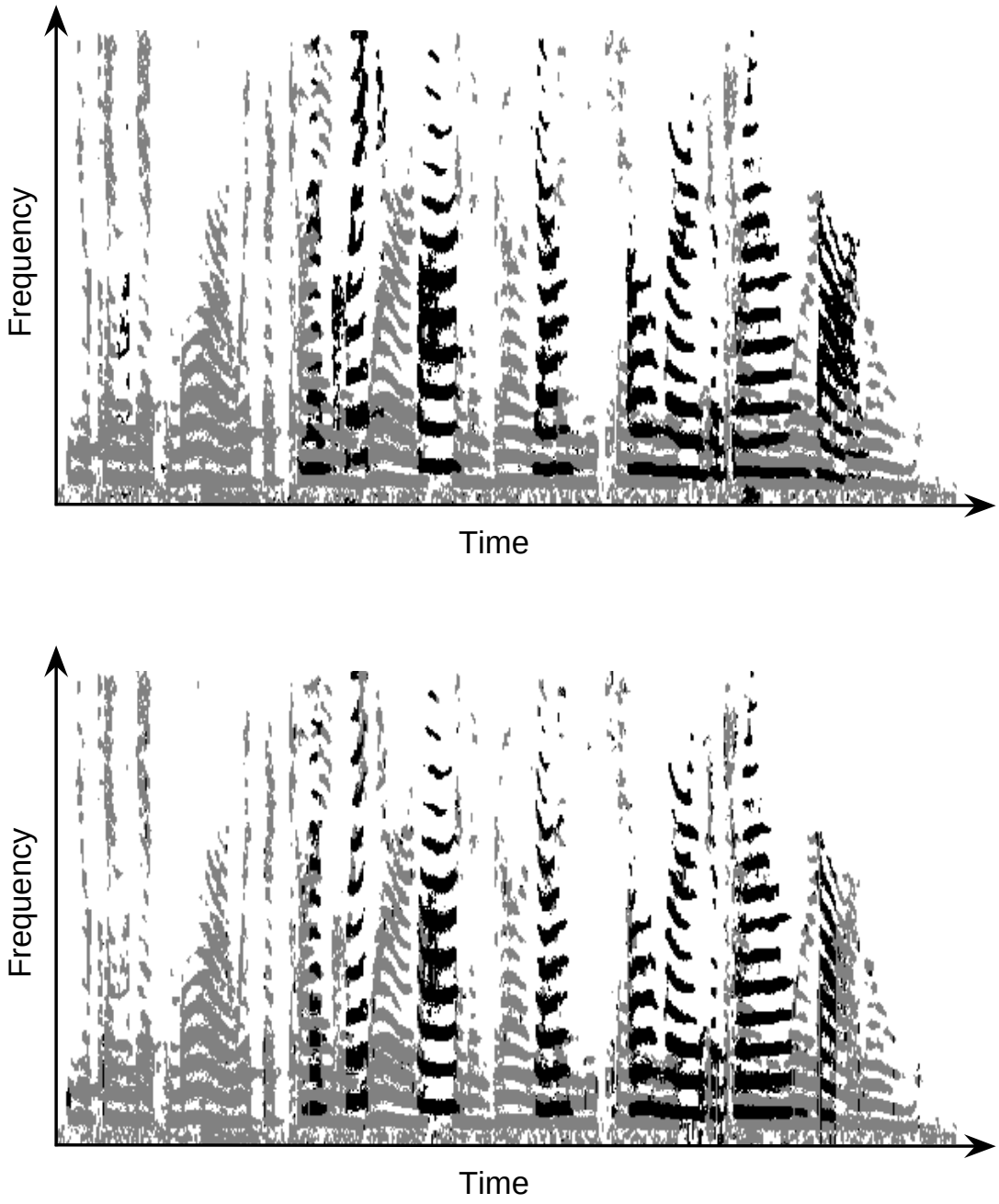


Figure 10: (Top) Optimal segmentation for the spectrogram of English speakers in Figure 9 (right), where the two speakers are “black” and “grey”; this segmentation is obtained from the known separated signals. (Bottom) The blind segmentation obtained with our algorithm.

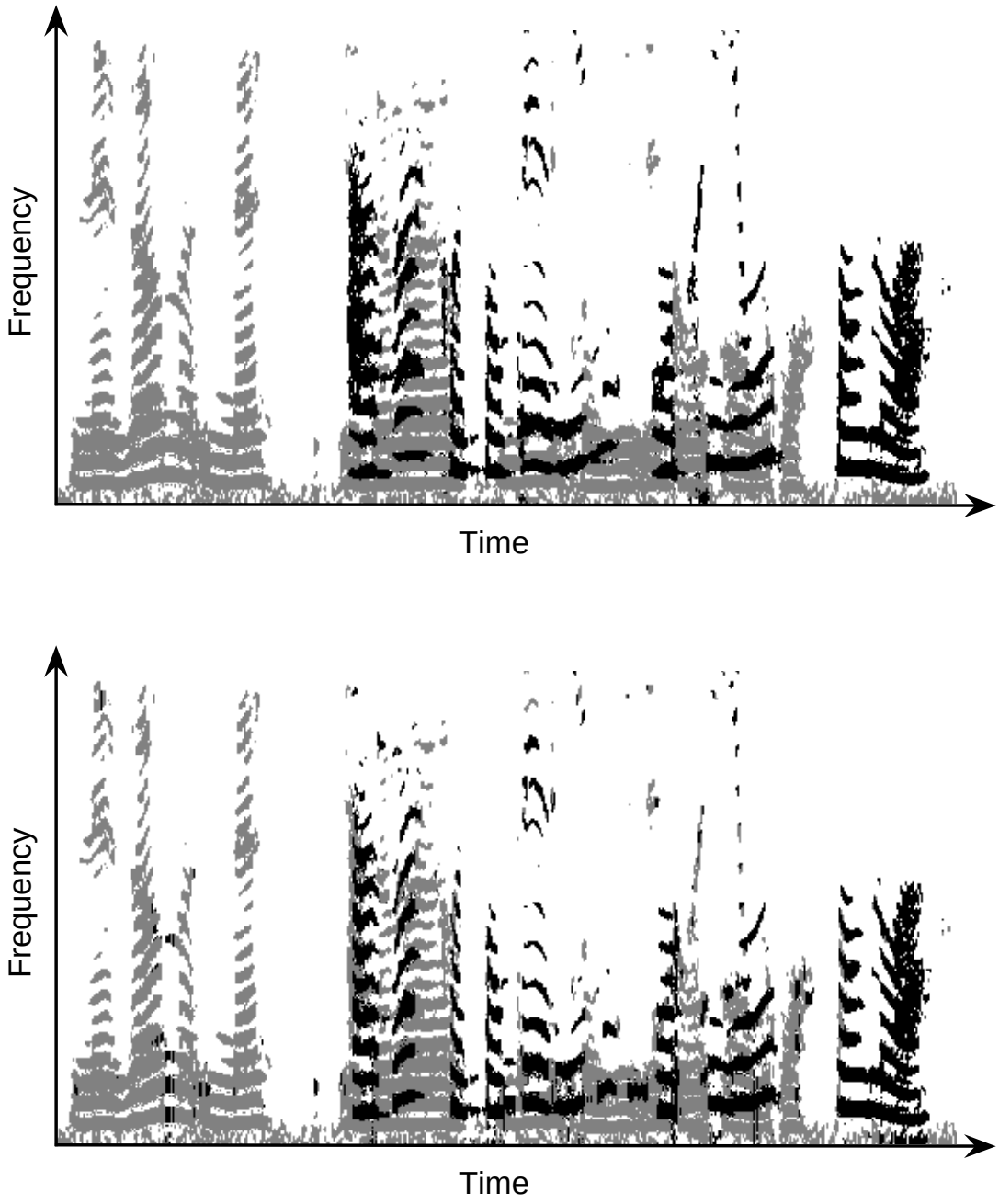


Figure 11: (Top) Optimal segmentation for the spectrogram of French speakers in Figure 9 (right), where the two speakers are “black” and “grey”; this segmentation is obtained from the known separated signals. (Bottom) The blind segmentation obtained with our algorithm.

the only information available for grouping in the frequency domain, and due in part to the fact that multi-pitch estimation is a hard problem and multi-pitch estimation procedures tend to lead to noisy estimates of pitch. We also see (the forth column in the table) that a simple frequency-based approach is not competitive. This is not surprising because natural speech tends to occupy the whole spectrum (because of non-voiced portions and variations in pitch).

Finally, as mentioned earlier, there was a major computational challenge in applying spectral methods to single microphone speech separation. Using the techniques described in Section 6.3, the separation algorithm has linear running time complexity and memory requirement and, coded in Matlab and C, it takes 3 minutes to separate 4 seconds of speech on a 2 GHz processor with 1GB of RAM.

7. Conclusions

In this paper, we have presented two sets of algorithms—one for spectral clustering and one for learning the similarity matrix. These algorithms can be derived as the minimization of a single cost function with respect to its two arguments. This cost function depends directly on the eigenstructure of the similarity matrix. We have shown that it can be approximated efficiently using the power method, yielding a method for learning similarity matrices that can cluster effectively in cases in which non-adaptive approaches fail. Note in particular that our new approach yields a spectral clustering method that is significantly more robust to irrelevant features than current methods.

We applied our learning framework to the problem of one-microphone blind source separation of speech. To do so, we have combined knowledge of physical and psychophysical properties of speech with learning algorithms. The former provide parameterized similarity matrices for spectral clustering, and the latter make use of our ability to generate segmented training data. The result is an optimized segmenter for spectrograms of speech mixtures. We have successfully demixed speech signals from two speakers using this approach.

Our work thus far has been limited to the setting of ideal acoustics and equal-strength mixing of two speakers. There are several obvious extensions that warrant investigation. First, the mixing conditions should be weakened and should allow some form of delay or echo. Second, there are multiple applications where speech has to be separated from non-stationary noise; we believe that our method can be extended to this situation. Third, our framework is based on segmentation of the spectrogram and, as such, distortions are inevitable since this is a "lossy" formulation (Jang and Lee, 2003; Jourjine et al., 2000). We are currently working on post-processing methods that remove some of those distortions. Finally, while the running time and memory requirements of our algorithm are linear in the duration of the signal to be separated, the resource requirements remain a concern. We are currently working on further numerical techniques that we believe will bring our method significantly closer to real-time.

Acknowledgments

We would like to thank anonymous reviewers for helpful comments, and acknowledge support for this project from the National Science Foundation (NSF grant 0412995), the Defense Advanced Research Projects Agency (contract NBCHD030010) and a graduate fellowship to Francis Bach from Microsoft Research.

Appendix A. Proof of Lemma 9

We prove the following lemma that relates distance between orthonormal bases of subspaces:

Lemma 9 *Let S and T be two matrices in $\mathbb{R}^{P\times R}$, with rank R. Let D be a symmetric positive definite matrix in $\mathbb{R}^{P\times P}$. Let $g(S,T)$ denote $\frac{1}{2}\|S(S^\top S)^{-1}S^\top - T(T^\top T)^{-1}T^\top\|_F^2$ and $\eta = \frac{\lambda_1(D)}{\lambda_P(D)} \geqslant 1$ denote the ratio of the largest and smallest eigenvalue of D. We then have:*

$$\frac{1}{\eta} g(D^{1/2}S, D^{1/2}T) \leqslant g(S,T) \leqslant \eta\, g(D^{1/2}S, D^{1/2}T). \tag{9}$$

Proof We can expand the Frobenius norm and rewrite $g(S,T)$ as

$$\begin{aligned} g(S,T) &= \frac{1}{2}\operatorname{tr}\Big\{S(S^\top S)^{-1}S^\top S(S^\top S)^{-1}S^\top \\ &\qquad -2S(S^\top S)^{-1}S^\top T(T^\top T)^{-1}T^\top + T(T^\top T)^{-1}T^\top T(T^\top T)^{-1}T^\top\Big\} \\ &= \operatorname{tr}\Big\{I - S(S^\top S)^{-1}S^\top T(T^\top T)^{-1}T^\top\Big\} \\ &= \operatorname{tr}\Big\{I - (S^\top S)^{-1/2}S^\top T(T^\top T)^{-1}T^\top S(S^\top S)^{-1/2}\Big\} \\ &= \operatorname{tr}\Big\{(S^\top S)^{-1/2}(S^\top S - S^\top T(T^\top T)^{-1}T^\top S)(S^\top S)^{-1/2}\Big\}. \end{aligned}$$

The matrix $S^\top S - S^\top T(T^\top T)^{-1}T^\top S$ is the Schur complement of the top left block in the matrix $M = (S\ T)^\top(S\ T) = \begin{pmatrix} S^\top S & S^\top T \\ T^\top S & T^\top T \end{pmatrix}$. Let $M_D = (S\ T)^\top D(S\ T)$. We have the following inequalities between matrices: $M_D \preceq \lambda_1(D)M$ and $M_D \succeq \lambda_P(D)M$, which implies the same inequality for the top left blocks ($S^\top S$ and $S^\top DS$) and their Schur complements ($N = S^\top S - S^\top T(T^\top T)^{-1}T^\top S$ and $N_D = S^\top DS - S^\top DT(T^\top DT)^{-1}T^\top DS$). We then have:

$$(S^\top S)^{-1/2}N(S^\top S)^{-1/2} \preceq \eta\,(S^\top DS)^{-1/2}N_D(S^\top DS)^{-1/2}$$

$$(S^\top S)^{-1/2}N(S^\top S)^{-1/2} \succeq \frac{1}{\eta}(S^\top DS)^{-1/2}N_D(S^\top DS)^{-1/2},$$

which implies Eq. (9) by taking the trace. ■

Appendix B. Proof of Proposition 7

Proposition 10 *The function $\Pi_R(M)$, the orthogonal projection on the R-th principal eigensubspace of R, is an infinitely differentiable function on $\mathcal{M}_{P,R}$. For any differentiable path $M(t)$ of symmetric matrices with values in $\mathcal{M}_{P,R}$ such that $M(0) = M$, the derivative $\frac{d\Pi_R(M(t))}{dt}$ at $t = 0$ is equal to $UV^\top + VU^\top$, where N is the unique solution of linear system:*

$$MN - NU^\top MU = -(I - UU^\top)M'(0)U \text{ and } U^\top N = 0 \tag{10}$$

where U is any orthonormal basis of the R-th principal subspace of M. The value of the derivative is independent of the chosen orthonormal basis.

Proof For simplicity, we assume that the first $R+1$ eigenvalues of $M_0 \in \mathcal{M}_{P,R}$ are distinct, noting that the result can be easily extended to cases where some of the first R eigenvalues coalesce. We let U_0 denote an orthogonal matrix composed of the first R eigenvectors of M_0 (well defined up to sign because all eigenvalues are simple), and $S_0 = U_0^\top S_0 U_0$ the diagonal matrix of the first R eigenvalues.

It is well known that the unit norm eigenvector associated with a simple eigenvalue is uniquely defined up to multiplication by ± 1, and that once a sign convention is adopted locally, the eigenvector and the eigenvalue are infinitely differentiable at M_0. Since the projection Π_R on the principal subspace is invariant with respect to the sign conventions, this implies that $\Pi_R(M)$ is infinitely differentiable at M_0. We now compute the derivative by showing that it can be obtained from the unique solution of a linear system.

We let U denote the first eigenvectors of M (with appropriate sign conventions) and S the diagonal matrix of eigenvalues of M. Differentiating the system,

$$U^\top U = I \text{ and } MU = US$$

we obtain:

$$U^\top dU + dU^\top U = 0 \text{ and } MdU + dM\, U - dU\, S - UdS = 0.$$

By pre-multiplying the second equation by $U^\top$, we obtain $dS = SU^\top dU + U^\top dM\, U - U^\top dU\, S$, and by substituting dS into the first equation, we obtain:

$$(M - USU^\top)dU - (I - UU^\top)dU\, S = -(I - UU^\top)dM\, U.$$

Moreover, we have $d\Pi = UdU^\top + dU\, U^\top$ and $dU^\top U + U^\top dU = 0$, which implies

$$d\Pi = (I - UU^\top)dU\, U^\top + UdU^\top (I - UU^\top).$$

If we let $N = (I - UU^\top)dU$, we have

$$d\Pi = NU^\top + UN^\top \tag{11}$$

$$N^\top U = 0 \tag{12}$$

$$MN - NS = -(I - UU^\top)dM\, U. \tag{13}$$

We have proved that the differential of Π must satisfy Eq. (11), Eq. (12) and Eq. (13). To complete the proof we have to prove that the system of equations Eq. (12) and Eq. (13) has a unique solution. We let T denote an orthonormal basis of the orthogonal complement of U. We can reparameterize N as $N = UA + TB$. The system then becomes:

$$A = 0 \text{ and } T^\top MTB - BS = -T^\top dMU.$$

The linear operator L from $\mathbb{R}^{(P-R)\times R}$ to $\mathbb{R}^{(P-R)\times R}$ defined by $LB = T^\top MTB - BS$ is self-adjoint; a short calculation shows that its largest eigenvalue is $\lambda_{R+1}(M) - \lambda_R(M) < 0$. The operator is thus negative definite and hence invertible. The system of equations Eq. (12) and Eq. (13) thus has a unique solution.

■

Moreover, when the matrix M_0 is positive semidefinite, The system of equations Eq. (12) and Eq. (13) can be solved by solving a positive definite linear system whose condition condition number is upper bounded by $1/(1 - \lambda_{R+1}(M)/\lambda_R(M))$. The conjugate gradient algorithm can be used to solve the system and can be designed so that each iteration is requiring R matrix vector multiplications by M.

Appendix C. Pitch Extraction

As seen in Section 5, harmonic features such as pitch are essential for successful speech separation. In this appendix, we derive a simple pattern matching procedure for pitch estimation.

C.1 Pitch Estimation for One Pitch

We assume that we are given one time slice s of the spectrogram magnitude, $s \in \mathbb{R}^M$. The goal is to have a specific pattern match s. Since the speech signals are real, the spectrogram is symmetric and we consider only $M/2$ samples.

If the signal is exactly periodic, then the spectrogram magnitude for that time frame is exactly a superposition of bumps at multiples of the fundamental frequency. The patterns we are considering thus have the following parameters: a "bump" function $u \mapsto b(u)$, a pitch $\omega \in [0, M/2]$ and a sequence of harmonics $x_1, \ldots, x_H$ at frequencies $\omega_1 = \omega, \ldots, \omega_H = H\omega$, where H is the largest acceptable harmonic multiple, that is, $H = \lfloor M/2\omega \rfloor$. The pattern $\tilde{s} = \tilde{s}(x, b, \omega)$ is then built as a weighted sum of bumps.

By pattern matching, we mean finding the pattern $\tilde{s}$ that is as close as possible to s in the L^2-norm sense. We impose a constraint on the harmonic strengths (x_h), namely, that they are samples at intervals $h\omega$ of a function g with small second derivative norm $\int_0^{M/2} |g^{(2)}(\omega)|^2 d\omega$. The function g can be seen as the envelope of the signal and is related to the "timbre" of the speaker (Bregman, 1990). The explicit consideration of the envelope and its smoothness is necessary for two reasons: (a) it provides a timbre feature helpful for separation, (b) it helps avoid pitch-halving, a traditional problem of pitch extractors (Gold and Morgan, 1999).

Given b and ω, we minimize with respect to x, $||s - \tilde{s}(x)||^2 + \lambda \int_0^{M/2} |g^{(2)}(\omega)|^2 d\omega$, where $x_h = g(h\omega)$. Since $\tilde{s}(x)$ is linear function of x, this is a spline smoothing problem, and the solution can be obtained in closed form with complexity $O(H^3)$ (Wahba, 1990).

We now have to search over b and ω, knowing that the harmonic strengths x can be found in closed form. We use exhaustive search on a grid for ω, while we take only a few bump shapes. The main reason for using several bump shapes is to account for the fact that voiced speech is only approximately periodic. For further details and extensions, see Bach and Jordan (2005).

C.2 Pitch Estimation for Several Pitches

If we are to estimate S pitches, we estimate them recursively, by removing the estimated harmonic signals. In this paper, we assume that the number of speakers and hence the maximum number of pitches is known. Note, however, that since all our pitch features are always used with their strengths, our separation method is relatively robust to situations in which we try to find too many pitches.

References

K. Achan, S. Roweis, and B. Frey. Probabilistic inference of speech signals from phaseless spectrograms. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

F. R. Bach and M. I. Jordan. Discriminative training of hidden Markov models for multiple pitch tracking. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *International Conference on Machine Learning (ICML)*, 2003.

K.-J. Bathe and E. L. Wilson. *Numerical Methods in Finite Element Analysis*. Prentice Hall, 1976.

D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

M. Blatt, M. Wiesman, and E. Domany. Data clustering using a model granular magnet. *Neural Computation*, 9:1805–1842, 1997.

A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press, 1990.

G. J. Brown and M. P. Cooke. Computational auditory scene analysis. *Computer Speech and Language*, 8:297–333, 1994.

P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral K-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 13(9):1088–1096, 1994.

F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

P. Comon and G. H. Golub. Tracking a few extreme singular values and vectors in signal processing. *Proceedings of the IEEE*, 78(8):1327–1343, 1990.

M. Cooke and D. P. W. Ellis. The auditory organization of speech and other sources in listeners and computational models. *Speech Communication*, 35(3-4):141–177, 2001.

T. Cour, N. Gogin, and J. Shi. Learning spectral graph segmentation. In *Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2005.

N. Cristianini, J. Shawe-Taylor, and J. Kandola. Spectral kernel methods for clustering. In *Advances in Neural Information Processing Systems, 14*. MIT Press, 2002.

I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and and graph cuts. Technical Report #TR-04-25, University of Texas, Computer Science, 2004.

C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2005.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1999.

C. Fowlkes, S. Belongie, and J. Malik. Efficient spatiotemporal grouping using the Nyström method. In *IEEE conference on Computer Vision and Pattern Recognition (ECCV)*, 2001.

B. Gold and N. Morgan. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley Press, 1999.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

A. G. Gray and A. W. Moore. N-Body problems in statistical learning. In *Advances in Neural Information Processing Systems, 13*. MIT Press, 2001.

D. W. Griffin and J. S. Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 32(2):236–243, 1984.

M. Gu, H. Zha, C. Ding, X. He, and H. Simon. Spectral relaxation models and structure analysis for K-way graph clustering and bi-clustering. Technical report, Penn. State Univ, Computer Science and Engineering, 2001.

D. Higham and M. Kibble. A unified view of spectral clustering. Technical Report 02, University of Strathclyde, Department of Mathematics, 2004.

L. J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

G.-J. Jang and T.-W. Lee. A maximum likelihood approach to single-channel source separation. *Journal of Machine Learning Research*, 4:1365–1392, 2003.

A. Jourjine, S. Rickard, and O. Yilmaz. Blind separation of disjoint orthogonal signals: Demixing N sources from 2 mixtures. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000.

S. D. Kamvar, D. Klein, and C. D. Manning. Spectral learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems, 12*. MIT Press, 2000.

J. R. Magnus and H. Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley, 1999.

S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.

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 *International Conference on Computer Vision (ICCV)*, 2001.

M. Meila and D. Heckerman. An experimental comparison of several clustering and initialization methods. *Machine Learning*, 42(1):9–29, 2001.

M. Meila and J. Shi. Learning segmentation by random walks. In *Advances in Neural Information Processing Systems, 14*. MIT Press, 2002.

M. Meila and L. Xu. Multiway cuts and spectral clustering. Technical report, University of Washington, Department of Statistics, 2003.

A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems, 14*. MIT Press, 2002.

M. L. Overton and R. S. Womersley. Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrics. *Mathematical Programming*, 62:321–357, 1993.

S. T. Roweis. One microphone source separation. In *Advances in Neural Information Processing Systems, 13*. MIT Press, 2001.

G. L. Scott and H. C. Longuet-Higgins. Feature grouping by relocalisation of eigenvectors of the proximity matrix. In *British Machine Vision Conference*, 1990.

N. Shental, A. Zomet, T. Hertz, and Y. Weiss. Learning and inferring image segmentations using the gbp typical cut algorithm. In *International Conference on Computer Vision (ICCV)*, 2003.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems, 14*. MIT Press, 2002.

N. Tishby and N. Slonim. Data clustering by Markovian relaxation and the information bottleneck method. In *Advances in Neural Information Processing Systems, 13*. MIT Press, 2001.

U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. In *Advances in Neural Information Processing Systems, 17*. MIT Press, 2005.

K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained K-means clustering with background knowledge. In *International Conference on Machine Learning (ICML)*, 2001.

G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.

Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1999.

E. P. Xing and M. I. Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical Report UCB/CSD-03-1265, EECS Department, University of California, Berkeley, 2003.

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*. MIT Press, 2003.

S. X. Yu and J. Shi. Grouping with bias. In *Advances in Neural Information Processing Systems, 14*. MIT Press, 2002.

S. X. Yu and J. Shi. Multiclass spectral clustering. In *International Conference on Computer Vision (ICCV)*, 2003.

H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for K-means clustering. In *Advances in Neural Information Processing Systems, 14*. MIT Press, 2002.

M. Zibulevsky, P. Kisilev, Y. Y. Zeevi, and B. A. Pearlmutter. Blind source separation via multinode sparse representation. In *Advances in Neural Information Processing Systems, 14*. MIT Press, 2002.

Journal of Machine Learning Research 7 (2006) 2003-2030 Submitted 3/06; Revised 7/06; Published 10/06

A Linear Non-Gaussian Acyclic Model for Causal Discovery

Shohei Shimizu* SHOHEIS@ISM.AC.JP
Patrik O. Hoyer PATRIK.HOYER@HELSINKI.FI
Aapo Hyvärinen AAPO.HYVARINEN@HELSINKI.FI
Antti Kerminen ANTTI.KERMINEN@HELSINKI.FI
Helsinki Institute for Information Technology, Basic Research Unit
Department of Computer Science
University of Helsinki
FIN-00014, Finland

Editor: Michael Jordan

Abstract

In recent years, several methods have been proposed for the discovery of causal structure from non-experimental data. Such methods make various assumptions on the data generating process to facilitate its identification from purely observational data. Continuing this line of research, we show how to discover the complete causal structure of continuous-valued data, under the assumptions that (a) the data generating process is linear, (b) there are no unobserved confounders, and (c) disturbance variables have non-Gaussian distributions of non-zero variances. The solution relies on the use of the statistical method known as independent component analysis, and does not require any pre-specified time-ordering of the variables. We provide a complete Matlab package for performing this LiNGAM analysis (short for Linear Non-Gaussian Acyclic Model), and demonstrate the effectiveness of the method using artificially generated data and real-world data.

Keywords: independent component analysis, non-Gaussianity, causal discovery, directed acyclic graph, non-experimental data

1. Introduction

Several authors (Spirtes et al., 2000; Pearl, 2000) have recently formalized concepts related to causality using probability distributions defined on directed acyclic graphs. This line of research emphasizes the importance of understanding the process which generated the data, rather than only characterizing the joint distribution of the observed variables. The reasoning is that a causal understanding of the data is essential to be able to predict the consequences of interventions, such as setting a given variable to some specified value.

One of the main questions one can answer using this kind of theoretical framework is: 'Under what circumstances and in what way can one determine causal structure on the basis of observational data alone?'. In many cases it is impossible or too expensive to perform controlled experiments, and hence methods for discovering likely causal relations from uncontrolled data would be very valuable.

Existing discovery algorithms (Spirtes et al., 2000; Pearl, 2000) generally work in one of two settings. In the case of discrete data, no functional form for the dependencies is usually assumed.

*. Current address: The Institute of Statistical Mathematics, 4-6-7 Minami-Azabu, Minato-ku, Tokyo 106-8569, Japan

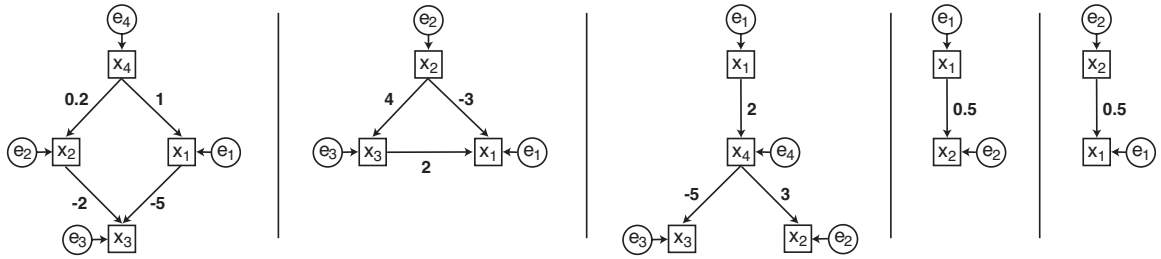


Figure 1: A few examples of data generating models satisfying our assumptions. For example, in the left-most model, the data is generated by first drawing the e_i independently from their respective non-Gaussian distributions, and subsequently setting (in this order) $x_4 = e_4$, $x_2 = 0.2x_4 + e_2$, $x_1 = x_4 + e_1$, and $x_3 = -2x_2 - 5x_1 + e_3$. (Here, we have assumed for simplicity that all the c_i are zero, but this may not be the case in general.) Note that the variables are not causally sorted (reflecting the fact that we usually do not know the causal ordering a priori), but that in each of the graphs they *can* be arranged in a causal order, as all graphs are directed acyclic graphs. In this paper we show that the full causal structure, including all parameters, are identifiable given a sufficient number of observed data vectors $\mathbf{x}$.

On the other hand, when working with continuous variables, a linear-Gaussian approach is almost invariably taken.

In this paper, we show that when working with continuous-valued data, a significant advantage can be achieved by departing from the Gaussianity assumption. While the linear-Gaussian approach usually only leads to a *set* of possible models, equivalent in their conditional correlation structure, a linear-*non-Gaussian* setting allows the full causal model to be estimated, with no undetermined parameters.

The paper is structured as follows.[1] First, in Section 2, we describe our assumptions on the data generating process. These assumptions are essential for the application of our causal discovery method, detailed in Sections 3 through 5. Section 6 discusses how one can test whether the found model seems plausible and proposes a statistical method for pruning edges. In Sections 7 and 8, we conduct a simulation study and provide real data examples to verify that our algorithm works as stated. We conclude the paper in Section 9.

2. Linear Causal Networks

Assume that we observe data generated from a process with the following properties:

1. The observed variables x_i, $i \in \{1,\ldots,m\}$ can be arranged in a *causal order*, such that no later variable causes any earlier variable. We denote such a causal order by $k(i)$. That is, the generating process is *recursive* (Bollen, 1989), meaning it can be represented graphically by a *directed acyclic graph* (DAG) (Pearl, 2000; Spirtes et al., 2000).

1. Preliminary results of the paper were presented at UAI2005 and ICA2006 (Shimizu et al., 2005, 2006b; Hoyer et al., 2006a).

2. The value assigned to each variable x_i is a *linear function* of the values already assigned to the earlier variables, plus a 'disturbance' (noise) term e_i, and plus an optional constant term c_i, that is

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

3. The disturbances e_i are all continuous-valued random variables with *non-Gaussian* distributions of non-zero variances, and the e_i are independent of each other, that is, $p(e_1,\ldots,e_m) = \prod_i p_i(e_i)$.

A model with these three properties we call a *Linear, Non-Gaussian, Acyclic Model*, abbreviated LiNGAM.

We assume that we are able to observe a large number of data vectors $\mathbf{x}$ (which contain the components x_i), and each is generated according to the above-described process, with the same causal order $k(i)$, same coefficients b_{ij}, same constants c_i, and the disturbances e_i sampled independently from the same distributions.

Note that the above assumptions imply that there are *no unobserved confounders* (Pearl, 2000).[2] Spirtes et al. (2000) call this the *causally sufficient* case. Also note that we do not require 'stability' in the sense as described by Pearl (2000), that is, 'faithfulness' (Spirtes et al., 2000) of the generating model. See Figure 1 for a few examples of data models fulfilling the assumptions of our model.

A key difference to most earlier work on the linear, causally sufficient, case is the assumption of non-Gaussianity of the disturbances. In most work, an explicit or implicit assumption of Gaussianity has been made (Bollen, 1989; Geiger and Heckerman, 1994; Spirtes et al., 2000). An assumption of Gaussianity of disturbance variables makes the full joint distribution over the x_i Gaussian, and the covariance matrix of the data embodies all one could possibly learn from observing the variables. Hence, all conditional correlations can be computed from the covariance matrix, and discovery algorithms based on conditional independence can be easily applied.

However, it turns out, as we will show below, that an assumption of *non*-Gaussianity may actually be more useful. In particular, it turns out that when this assumption is valid, the complete causal structure can in fact be estimated, without any prior information on a causal ordering of the variables. This is in stark contrast to what can be done in the Gaussian case: algorithms based only on second-order statistics (i.e., the covariance matrix) are generally not able to discern the full causal structure in most cases. The simplest such case is that of two variables, x_1 and x_2. A method based only on the covariance matrix has no way of preferring $x_1 \to x_2$ over the reverse model $x_1 \leftarrow x_2$; indeed the two are indistinguishable in terms of the covariance matrix (Spirtes et al., 2000). However, assuming non-Gaussianity, one can actually discover the direction of causality, as shown by Dodge and Rousson (2001) and Shimizu and Kano (2006). This result can be extended to several variables (Shimizu et al., 2006a). Here, we further develop the method so as to estimate the full model including all parameters, and we propose a number of tests to prune the graph and to see whether the estimated model fits the data.

2. A simple explanation is as follows: Denote by f hidden common causes and by $\mathbf{G}$ its connection strength matrix. Then a new model with hidden common causes f can be written as $\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{G}f + e'$. Since common causes f introduce some dependency between $e = \mathbf{G}f + e'$, the new model is different from the LiNGAM model with independent (not merely uncorrelated) disturbances e. See Hoyer et al. (2006b) for details.

3. Model Identification Using Independent Component Analysis

The key to the solution to the linear discovery problem is to realize that the observed variables are linear functions of the disturbance variables, and the disturbance variables are mutually independent and non-Gaussian. If we as preprocessing subtract out the mean of each variable x_i, we are left with the following system of equations:

$$\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{e}, \tag{1}$$

where $\mathbf{B}$ is a matrix that could be permuted (by simultaneous equal row and column permutations) to strict lower triangularity if one knew a causal ordering $k(i)$ of the variables (Bollen, 1989). (Strict lower triangularity is here defined as lower triangular with all zeros on the diagonal.) Solving for $\mathbf{x}$ one obtains

$$\mathbf{x} = \mathbf{A}\mathbf{e}, \tag{2}$$

where $\mathbf{A} = (\mathbf{I} - \mathbf{B})^{-1}$. Again, $\mathbf{A}$ could be permuted to lower triangularity (although not *strict* lower triangularity, actually in this case all diagonal elements will be *non-zero*) with an appropriate permutation $k(i)$. Taken together, Equation (2) and the independence and non-Gaussianity of the components of $\mathbf{e}$ define the standard linear *independent component analysis* model.

Independent component analysis (ICA) (Comon, 1994; Hyvärinen et al., 2001) is a fairly recent statistical technique for identifying a linear model such as that given in Equation (2). If the observed data is a linear, invertible mixture of non-Gaussian independent components, it can be shown (Comon, 1994) that the mixing matrix $\mathbf{A}$ is identifiable (up to scaling and permutation of the columns, as discussed below) given enough observed data vectors $\mathbf{x}$. Furthermore, efficient algorithms for estimating the mixing matrix are available (Hyvärinen, 1999).

We again want to emphasize that ICA uses non-Gaussianity (that is, more than covariance information) to estimate the mixing matrix $\mathbf{A}$ (or equivalently its inverse $\mathbf{W} = \mathbf{A}^{-1}$). For Gaussian disturbance variables e_i, ICA cannot in general find the correct mixing matrix because many different mixing matrices yield the same covariance matrix, which in turn implies the exact same Gaussian joint density (Hyvärinen et al., 2001). Our requirement for non-Gaussianity of disturbance variables stems from the same requirement in ICA.

While ICA is essentially able to estimate $\mathbf{A}$ (and $\mathbf{W}$), there are two important indeterminacies that ICA cannot solve: First and foremost, the order of the independent components is in no way defined or fixed (Comon, 1994). Thus, we could reorder the independent components and, correspondingly, the columns of $\mathbf{A}$ (and rows of $\mathbf{W}$) and get an equivalent ICA model (the same probability density for the data). In most applications of ICA, this indeterminacy is of no significance and can be ignored, but in LiNGAM, we can and we have to find the correct permutation as described in Section 4 below.

The second indeterminacy of ICA concerns the scaling of the independent components. In ICA, this is usually handled by assuming all independent components to have unit variance, and scaling $\mathbf{W}$ and $\mathbf{A}$ appropriately. On the other hand, in LiNGAM (as in SEM) we allow the disturbance variables to have arbitrary (non-zero) variances, but fix their weight (connection strength) to their corresponding observed variable to unity. This requires us to re-normalize the rows of $\mathbf{W}$ so that all the diagonal elements equal unity, before computing $\mathbf{B}$, as described in the LiNGAM algorithm below.

Our discovery algorithm, detailed in the next section, can be briefly summarized as follows: First, use a standard ICA algorithm to obtain an estimate of the mixing matrix $\mathbf{A}$ (or equivalently

of $\mathbf{W}$), and subsequently permute it and normalize it appropriately before using it to compute $\mathbf{B}$ containing the sought connection strengths b_{ij}.[3]

4. LiNGAM Discovery Algorithm

Based on the observations given in Sections 2 and 3, we propose the following causal discovery algorithm:

Algorithm A: LiNGAM discovery algorithm

1. Given an $m \times n$ data matrix $\mathbf{X}$ ($m \ll n$), where each column contains one sample vector $\mathbf{x}$, first subtract the mean from each row of $\mathbf{X}$, then apply an ICA algorithm to obtain a decomposition $\mathbf{X} = \mathbf{AS}$ where $\mathbf{S}$ has the same size as $\mathbf{X}$ and contains in its rows the independent components. From here on, we will exclusively work with $\mathbf{W} = \mathbf{A}^{-1}$.

2. Find the one and only permutation of rows of $\mathbf{W}$ which yields a matrix $\widetilde{\mathbf{W}}$ without any zeros on the main diagonal. In practice, small estimation errors will cause all elements of $\mathbf{W}$ to be non-zero, and hence the permutation is sought which minimizes $\sum_i 1/|\widetilde{\mathbf{W}}_{ii}|$.

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 $\widehat{\mathbf{B}} = \mathbf{I} - \widetilde{\mathbf{W}}'$.

5. Finally, to find a causal order, find the permutation matrix $\mathbf{P}$ (applied equally to both rows and columns) of $\widehat{\mathbf{B}}$ which yields a matrix $\widetilde{\mathbf{B}} = \mathbf{P}\widehat{\mathbf{B}}\mathbf{P}^T$ which is as close as possible to strictly lower triangular. This can be measured for instance using $\sum_{i \le j} \widetilde{\mathbf{B}}_{ij}^2$.

A complete Matlab code package implementing this algorithm is available online at our LiNGAM homepage: `http://www.cs.helsinki.fi/group/neuroinf/lingam/`

We now describe each of these steps in more detail.

In the first step of the algorithm, the ICA decomposition of the data is computed. Here, any standard ICA algorithm can be used. Although our implementation uses the FastICA algorithm (Hyvärinen, 1999), one could equally well use one of the many other algorithms available (see e.g., Hyvärinen et al., 2001). However, it is important to select an algorithm which can estimate independent components of many different distributions, as in general the distributions of the disturbance variables will not be known in advance. For example, FastICA can estimate both super-Gaussian and sub-Gaussian independent components, and we don't need to know the actual functional form of the non-Gaussian distributions (Hyvärinen, 1999).

Because of the permutation indeterminacy of ICA, the rows of $\mathbf{W}$ will be in random order. This means that we do not yet have the correct correspondence between the disturbance variables e_i and the observed variables x_i. The former correspond to the rows of $\mathbf{W}$ while the latter correspond to the columns of $\mathbf{W}$. Thus, our first task is to permute the rows to obtain a correspondence between the rows and columns. If $\mathbf{W}$ were estimated exactly, there would be only a single row permutation

3. It would be extremely difficult to estimate $\mathbf{B}$ directly using a variant of ICA algorithms, because we don't know the correct order of the variables, that is, the matrix $\mathbf{B}$ should be restricted to 'permutable to lower triangularity' not 'lower triangular' directly. This is due to the permutation problem illustrated in Appendix B.

that would give a matrix with no zeros on the diagonal, and this permutation would give the correct correspondence. This is because of the assumption of DAG structure, which is the key to solving the permutation indeterminacy of ICA. (A proof of this is given in Appendix A, and an example of the permutation problem is provided in Appendix B.)

In practice, however, ICA algorithms applied on finite data sets will yield estimates which are only approximately zero for those elements which should be exactly zero, and the model is only approximately correct for real data. Thus, our algorithm searches for the permutation using a cost function which heavily penalizes small absolute values in the diagonal, as specified in step 2. In addition to being intuitively sensible, this cost function can also be derived from a maximum-likelihood framework; for details, see Appendix C.

When the number of observed variables x_i is relatively small (less than eight or so) then finding the best permutation is easy, since a simple exhaustive search can be performed. However, for higher dimensionalities a more sophisticated method is required. We also provide such a permutation method for large dimensions; for details, see Section 5.

Once we have obtained the correct correspondence between rows and columns of the ICA decomposition, calculating our estimates of the b_{ij} is straightforward. First, we normalize the rows of the permuted matrix to yield a diagonal with all ones, and then remove this diagonal and flip the sign of the remaining coefficients, as specified in steps 3 and 4.

Although we now have estimates of all coefficients b_{ij} we do not yet have available a causal ordering $k(i)$ of the variables. Such an ordering (in general there may exist many if the generating network is not fully connected) is important for visualizing the resulting graph. A causal ordering can be found by permuting both rows and columns (using the same permutation) of the matrix $\widehat{\mathbf{B}}$ (containing the estimated connection strengths) to yield a strictly lower triangular matrix. If the estimates were exact, this would be a trivial task. However, since our estimates will not contain exact zeros, we will have to settle for approximate strict lower triangularity, measured for instance as described in step 5.[4]

It has to be noted that the computational stability of our method cannot be guaranteed. This is because ICA estimation is typically based on optimization of non-quadratic, possibly non-convex functions, and the algorithm might get stuck in local minima. Thus, for different random initial points used in the optimization algorithm, we might get different estimates of $\mathbf{W}$. An empirical observation is that typically ICA algorithms are relatively stable when the model holds, and unstable when the model does not hold. For a computational method addressing this issue, based on rerunning the ICA estimation part with different initial points, see Himberg et al. (2004).

5. Permutation Algorithms for Large Dimensions

In this section, we describe efficient algorithms for finding the permutations in steps 2 and 5 of the LiNGAM algorithm.

4. A reviewer pointed out that from a Bayesian viewpoint, the non-zero entries of the matrix $\widehat{\mathbf{B}}$ that would be zero in the infinite data case manifest a more general concept: the data cannot identify 'the' DAG structure, they can only help assign posterior probabilities to different structures.

5.1 Permuting the Rows of W

An exhaustive search over all possible row-permutations is feasible only in relatively small dimensions. For larger problems other optimization methods are needed. Fortunately, it turns out that the optimization problem can be written in the form of the classical *linear assignment problem*. To see this set $C_{ij} = 1/|\widetilde{\mathbf{W}}_{ij}|$, in which case the problem can be written as the minimization of

$$\sum_{i=1}^{m} C_{\phi(i),i},$$

where ϕ denotes the permutation to be optimized over. A great number of algorithms exist for this problem, with the best achieving worst-case complexity of $O(m^3)$ where m is the number of variables (see e.g., Burkard and Cela, 1999).

5.2 Permuting B to Get a Causal Order

It would be trivial to permute both rows and columns (using the same permutation) of $\widehat{\mathbf{B}}$ to yield a strictly lower triangular matrix if the estimates were exact, because one could use the following algorithm:

Algorithm B: Testing for DAGness, and returning a causal order if true

1. Initialize the permutation p to be an empty list
2. Repeat until $\widehat{\mathbf{B}}$ contains no more elements:
   (a) Find a row i of $\widehat{\mathbf{B}}$ containing all zeros, if not possible return **false**
   (b) Append i to the end of the list p
   (c) Remove the i-th row and the i-th column from $\widehat{\mathbf{B}}$
3. Return **true** and the found permutation p

However, since our estimates will not contain exact zeros, we will have to find a permutation such that setting the upper triangular elements to zero changes the matrix as little as possible. For instance, we could define our objective to be to minimize the sum of squares of elements on and above the diagonal, that is $\sum_{i\leq j} \widetilde{\mathbf{B}}_{ij}^2$ where $\widetilde{\mathbf{B}} = \mathbf{P}\widehat{\mathbf{B}}\mathbf{P}^T$ denotes the permuted $\widehat{\mathbf{B}}$, and $\mathbf{P}$ denotes the permutation matrix representing the sought permutation. In low dimensions, the optimal permutation can be found by exhaustive search. However, for larger problems this is obviously infeasible. Since we are not aware of any efficient method for exactly solving this combinatorial problem, we have taken another approach to handling the high-dimensional case.

Our approach is based on setting small (absolute) valued elements to zero, and testing whether the resulting matrix can be permuted to strict lower triangularity. Thus, the algorithm is:

Algorithm C: Finding a permutation of $\widehat{\mathbf{B}}$ by iterative pruning and testing

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

2. Repeat

   (a) Test if $\widehat{\mathbf{B}}$ can be permuted to strict lower triangularity (using Algorithm B above). If the answer is yes, stop and return the permuted $\widehat{\mathbf{B}}$, that is, $\widetilde{\mathbf{B}}$.

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

---

If in the estimated $\widehat{\mathbf{B}}$, all the true zeros resulted in estimates smaller than all of the true non-zeros, this algorithm finds the optimal permutation. In general, however, the result is not optimal in terms of the above proposed objective. However, simulations below show that the approximation works quite well.

6. Statistical Tests for Pruning Edges

The LiNGAM algorithm consistently estimates the connection strengths (and a causal order) if the model assumptions hold and the amount of data is sufficient. But what if our assumptions do not in fact hold? In such a case there is of course no guarantee that the proposed discovery algorithm will find true causal relationships between the variables.

The good news is that, in some cases, it is possible to detect violations of the model assumptions. In the following sections, we provide three statistical tests: i) testing significance of b_{ij} for pruning edges; ii) examining an overall fit of the model assumptions including estimated structure and connection strengths to data; iii) comparing two nested models. Then we propose a method for pruning edges of an estimated network using these statistical tests.

Unfortunately, however, it is never possible to completely confirm the assumptions (and hence the found causal model) purely from observational data. Controlled experiments, where the individual variables are explicitly manipulated (often by random assignment) and their effects monitored, are the only way to verify any causal model. Nevertheless, by testing the fit of the estimated model to the data we can recognize situations in which the assumptions clearly do not hold and reject models (e.g., Bollen, 1989). Only pathological cases constructed by mischievous data designers seem likely to be problematic for our framework. Thus, we think that a LiNGAM analysis will prove a useful first step in many cases for providing educated guesses of causal models, which might subsequently be verified in systematic experiments.

6.1 Wald Test for Examining Significance of Edges

After finding a causal ordering $k(i)$, we set to zero the coefficients of $\widehat{\mathbf{B}}$ which are implied zero by the order (i.e., those corresponding to the upper triangular part of the causally permuted connection matrix $\widetilde{\mathbf{B}}$). However, all remaining connections are in general non-zero. Even estimated connection strengths which are exceedingly weak (and hence probably zero in the generating model) remain and the network is fully connected. Both for achieving an intuitive understanding of the data, and especially for visualization purposes, a pruned network would be desirable. The Wald statistics provided below can be used to test which remaining connections should be pruned.

We would like to test if the coefficients of $\mathbf{B}$ are zero or not, which is equivalent to testing the coefficients of $\widetilde{\mathbf{W}}$ (see steps 3 and 4 in the LiNGAM algorithm above). Such tests are conducted to answer the fundamental question: Does the observed variable x_j have a statistically significant

effect on x_i? Here, the null and alternative hypotheses H_0 and H_1 are as follows:

$$H_0 : \widetilde{w}_{ij} = 0 \qquad \text{versus} \qquad H_1 : \widetilde{w}_{ij} \neq 0,$$

equivalently

$$H_0 : b_{ij} = 0 \qquad \text{versus} \qquad H_1 : b_{ij} \neq 0.$$

One can use the following Wald statistics

$$\frac{\widetilde{w}_{ij}^2}{\text{avar}(\widetilde{w}_{ij})},$$

to test significance of $\widetilde{w}_{ij}$ (or b_{ij}), where $\text{avar}(\widetilde{w}_{ij})$ denote the asymptotic variances of $\widetilde{w}_{ij}$ (see Appendix D for the complete formulas). The Wald statistics can be used to test the null hypothesis H_0. Under H_0, the Wald statistic asymptotically approximates to a chi-square variate with one degree of freedom (Bollen, 1989). Then we can obtain the probability of having a value of the Wald statistic larger than or equal to the empirical one computed from data. We reject H_0 if the probability is smaller than a significance level, and otherwise we accept H_0. Acceptance of H_0 implies that the assumption $\widetilde{w}_{ij} = 0$ (or b_{ij}) fits data. Rejection of H_0 suggests that the assumption is in error so that H_1 holds (Bollen, 1989). Thus, we can test significance of remaining edges using Wald statistics above.

6.2 A Chi-Square Test for Evaluating the Overall Fit of the Estimated Model

Next we propose a statistical measure using the model-based second-order moment structure to evaluate an overall fit of the model, for example, linearity, lower-triangularity (acyclicity), estimated structure and connection strengths, to data.

6.2.1 MOMENT STRUCTURES OF MODELS

First, we introduce some notations. For simplicity, assume $\mathbf{x}$ to have zero mean. Let us denote by $\sigma_2(\tau)$ the vector that consists of elements of the covariance matrix based on the model where any duplicates due to symmetry have been removed and by τ the vector of statistics of disturbances and coefficients of $\mathbf{B}$ that uniquely determines the second-order moment structures of the model $\sigma_2(\tau)$. Then the $\sigma_2(\tau)$ can be written as

$$\sigma_2(\tau) \quad = \quad \text{vec}^+\{E(\mathbf{x}\mathbf{x}^T)\}, \tag{3}$$

where $\text{vec}^+(\cdot)$ denotes the vectorization operator which transforms a symmetric matrix to a column vector by taking its non-duplicate elements. The parameter vector τ consists of free parameters of $\mathbf{B}$ and $E(e_i^2)$.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a random sample from a LiNGAM model in (1), and define the sample counterparts to the moments in (3) as

$$m_2 \quad = \quad \frac{1}{n}\sum_{j=1}^{n} \text{vec}^+(\mathbf{x}_j \mathbf{x}_j^T).$$

Let us denote by τ_0 the true parameter vector. The $\sigma_2(\tau_0)$ can be estimated by the m_2 when n is enough large: $\sigma_2(\tau_0) \approx m_2$.

We now propose to evaluate the fit of the model by measuring the distance between the moments of the observed data m_2 and those based on the model $\sigma_2(\tau)$ in a weighted least-squares sense (see below for details). In the approach, a large residual can be considered as badness of fit of the model found to data, which would imply violation of the model assumptions. Thus, this approach gives information on validity of the assumptions.

6.2.2 SOME TEST STATISTICS TO EVALUATE A MODEL FIT

We provide some test statistics to examine an overall model fit. Here, the null and alternative hypotheses H_0 and H_1 are as follows:

$$H_0 : E(m_2) = \sigma_2(\tau) \quad \text{versus} \quad H_1 : E(m_2) \neq \sigma_2(\tau),$$

where $E(m_2)$ is the expectation of m_2. Assume that the fourth-order moments of $\mathbf{x}_i$ are finite. Let us denote by $\mathbf{V}$ the covariance matrix of m_2, which consists of fourth-order moments $\mathrm{cov}(x_i x_j, x_k x_l) = E(x_i x_j x_k x_l) - E(x_i x_j)E(x_k x_l)$. One can take a sample covariance matrix of m_2 as a nonparametric estimator $\widehat{\mathbf{V}}$ for $\mathbf{V}$.

Denote $\mathbf{J} = \partial\sigma_2(\tau)/\partial\tau^T$ and assume that $\mathbf{J}$ is of full column rank (see Appendix E for the exact form). Define

$$F(\widehat{\tau}) = \{m_2 - \sigma_2(\widehat{\tau})\}^T \widehat{\mathbf{M}} \{m_2 - \sigma_2(\widehat{\tau})\},$$

where

$$\begin{aligned} \widehat{\mathbf{M}} &= \widehat{\mathbf{V}}^{-1} - \widehat{\mathbf{V}}^{-1}\widehat{\mathbf{J}}(\widehat{\mathbf{J}}^T\widehat{\mathbf{V}}^{-1}\widehat{\mathbf{J}})^{-1}\widehat{\mathbf{J}}^T\widehat{\mathbf{V}}^{-1} \\ \widehat{\mathbf{J}} &= \left.\frac{\partial\sigma_2(\tau)}{\partial\tau^T}\right|_{\tau=\widehat{\tau}}. \end{aligned} \tag{4}$$

Then a test statistic $T_1 = n \times F(\widehat{\tau})$ could be used to test the null hypothesis H_0, that is, to examine a fit of the model considered to data. Under H_0, the statistic T_1 asymptotically approximates to a chi-square variate with degrees $u - v$ of freedom where u is the number of distinct moments employed and v is the number of parameters employed to represent the second-order moment structure $\sigma_2(\tau)$, that is, the number of elements of τ. The required assumption for this is that $\hat{\tau}$ is a $\sqrt{n}$-consistent estimator. No asymptotic normality is required (see Browne, 1984, for details). Acceptance of H_0 implies that the model assumptions fit data. Rejection of H_0 suggests that at least one model assumption is in error so that H_1 holds (Bollen, 1989). Thus, we can assess the overall fit of the estimated model to data.

However, it is often pointed out that this type of test statistics requires large sample sizes for T_1 to behave like a chi-square variate (e.g., Hu et al., 1992). Therefore, we would apply a proposal by Yuan and Bentler (1997) to T_1 to improve its chi-square approximation and employ the following test statistic T_2:

$$T_2 = \frac{T_1}{1 + F(\widehat{\tau})}.$$

6.2.3 A DIFFERENCE CHI-SQUARE TEST FOR MODEL COMPARISON OF NESTED MODELS

Let us consider the comparison of two models that are nested, that is, one is a simplified model of the other. Assume that Models 1 and 2 have q and $q-1$ edges, and Model 2 is a simplified version of

Model 1 by pruning one edge out. Denote by $T_2(q)$ and $T_2(q-1)$ the model fit statistics for Models 1 and 2, respectively. Then, the difference between $T_2(q) - T_2(q-1)$ asymptotically approximates to a chi-square variate with one degree of freedom (e.g., Bollen, 1989), by which we can test if the two models with q and $q-1$ edges have significantly different model fits. In principle, a more complex model fits better. If the two model fits are significantly different, the edge should not be pruned since the model fit becomes significantly worse. This means that we examine significance of the edge in terms of overall model fit.

6.3 A Method for Pruning Edges

Using the tests developed above, we now propose a sophisticated method for pruning the edges (connection strengths).

The Wald statistics above tell us how likely each edge is, which can be considered an evaluation of the individual fit of each edge to data. On the other hand, the chi-square test assesses the overall model fit by measuring the residual between the data covariance matrix and model-based covariance matrix. A straightforward approach would be to test the significance of remaining edges using Wald statistics only. That is, we prune all the non-significant edges with the p values higher than a significance level, for example, 0.05 (5%). However, it would be more effective (e.g., the test has more power) to use both the individual and overall fits for assessing significance of edges. Furthermore, it is also important that the pruned estimated model is accepted by the chi-square test of model fit. Thus, we propose a pruning method utilizing all the three tests above, Wald test, the chi-square test and the difference test (see Section 6.2.3 for the difference test). The algorithm is as follows:

Algorithm D: Pruning edges using Wald test, model fit test and difference test.

1. Set a significance level α (e.g., 0.05)
2. Find non-significant edges by applying Wald test to each edge
3. Set the least significant strictly lower triangular element of $\widetilde{\mathbf{B}}$ (in Step 5 of the LiNGAM discovery algorithm) among the non-significant edges accepted by Wald test to zero
4. Repeat until all the non-significant edges by Wald test are examined
   (a) Test if the overall model fits for the last model and current model with one less edge than the last model are significantly different by the difference test. Further test the model fit of the current model by the chi-square test. If both null hypotheses are accepted in the two tests, adopt the current model, that is, prune the edge out. Otherwise, adopt the last model, that is, do not prune the edge.
   (b) Additionally set the next least significant element of $\widetilde{\mathbf{B}}$ to zero
5. Return the pruned $\widetilde{\mathbf{B}}$

The pruned $\widehat{\mathbf{B}}$ can be obtained by the relation $\widehat{\mathbf{B}} = \mathbf{P}^T \widetilde{\mathbf{B}} \mathbf{P}$ (see step 5 in the LiNGAM algorithm). Thus, we would be able to find a pruned network that fits data. We conduct a simulation to study the empirical performance of this pruning method (Section 7.2).

A potential alternative to Wald statistics would be to use resampling techniques (e.g., Efron and Tibshirani, 1993). We provide a basic method using resamplings as an option in our Matlab code. In our implementation we take the causal ordering obtained from the LiNGAM algorithm, and then simply estimate the connection strengths using covariance information alone for different resamplings of the original data. In this way, it is possible to obtain measures of the variances of the estimates of the b_{ij}, and use these variances to prune those edges whose estimated means are low compared with their standard deviations. Future versions of our software packages should incorporate the more advanced methods including bootstrapping.

The issue of multiple comparisons also arises in this context. Usually, $\widetilde{\mathbf{W}}$ and $\mathbf{B}$ have more than one element. In many cases, we need to perform more than one test simultaneously to find out if all or a set of the coefficients are significantly large in an absolute value sense. Although a given significance level may be appropriate for each individual test, it is not for the set of all the tests. We could have a lot of spurious significance if we just repeat tests without any corrections. In such a case, it would be effective to employ multiple comparison procedures (see Hochberg and Tamhane, 1987, for details). A simple and basic method is the Bonferroni correction, where we simply divide a significance level by the number of tests to obtain the significance level for individual test. However, it is often pointed out that the Bonferroni method is too conservative when the number of tests is large. Some authors have improved the Bonferroni procedure or devised new techniques so that they have more power of test (e.g., Benjamini and Hochberg, 1995; Hochberg, 1988; Holm, 1979; Simes, 1986). We would like to study such multiple comparison techniques in future work and implement them in our software package.[5]

7. Simulations

To verify the validity of our method (and of our Matlab code), we performed extensive experiments with simulated data. All experimental code (including the precise code to produce Figures 2, 3, 4 and Table 7.2) is included in the LiNGAM code package.

7.1 Estimation of B

We repeatedly performed the following experiment:

1. First, we randomly constructed a strictly lower-triangular matrix $\mathbf{B}$. Various dimensionalities (3, 5, 10, 20 and 100) were used. Both fully connected (no zeros in the strictly lower triangular part) and sparse networks (many zeros) were tested. We also randomly selected variances of the disturbance variables and values for the constants c_i.

2. Next, we generated data by independently drawing the disturbance variables e_i from Gaussian distributions and subsequently passing them through a power non-linearity (raising the absolute value to an exponent in the interval [0.5, 0.8] or [1.2, 2.0], but keeping the original sign) to make them non-Gaussian. Various data set sizes (200, 1000 and 5000) were tested. The e_i were then scaled to yield the desired variances, and the observed data $\mathbf{X}$ was generated according to the assumed recursive process.

5. It would also be possible to devise a Bayesian technique for scoring models as proposed by Geiger and Heckerman (1994) if we knew the distributions of non-Gaussian disturbances. However, in practice, it is quite difficult to model the exact functional form of the non-Gaussian distributions, and therefore it would be difficult to score the models requiring parametric models.

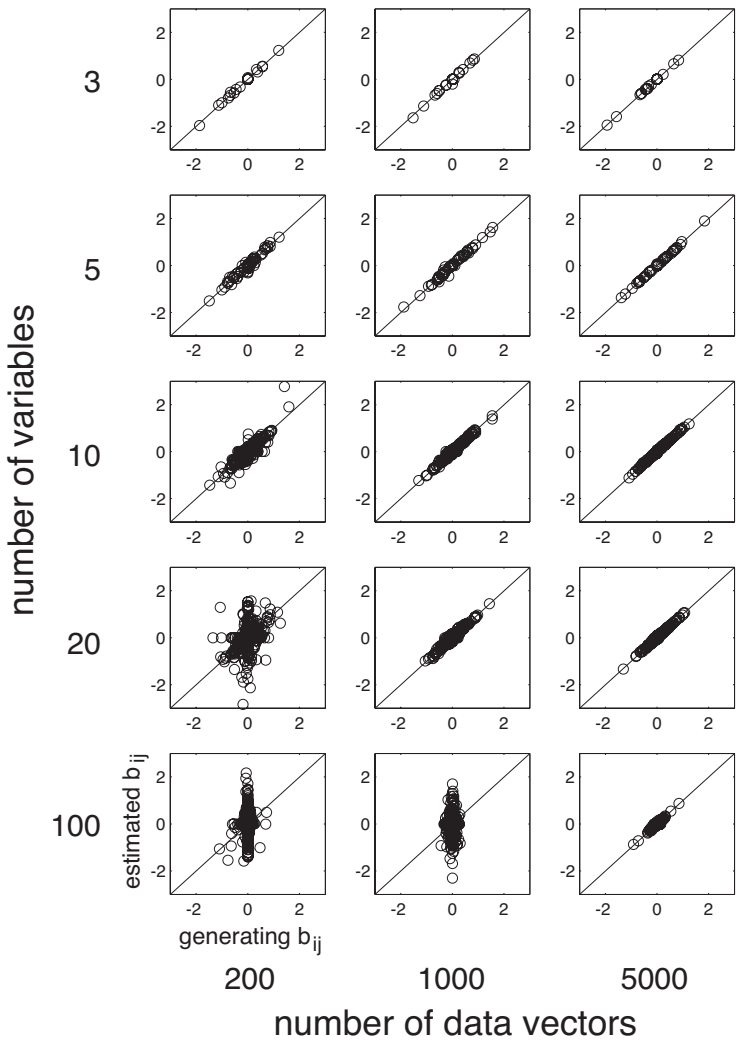


Figure 2: Scatterplots of the estimated b_{ij} versus the original (generating) values. The different plots correspond to different numbers of variables and different numbers of data vectors. Five data sets were generated for each scatterplot. For the last two rows, 1,000 plot points were randomly selected and plotted to improve the clarity of the figures.

3. Before feeding the data to the LiNGAM algorithm, we randomly permuted the rows of the data matrix $\mathbf{X}$ to hide the causal order with which the data was generated. At this point, we also permuted $\mathbf{B}$, the c_i, as well as the variances of the disturbance variables to match the new order in the data.

4. Finally, we fed the data to our discovery algorithm, and compared the estimated parameters to the generating parameters. In particular, we made a scatterplot of the entries in the estimated matrix $\widehat{\mathbf{B}}$ against the corresponding ones in $\mathbf{B}$.

Since the number of different possible parameter configurations is limitless, we feel that the reader is best convinced by personally running the simulations using various settings. Nevertheless, we here show some representative results.

Figure 2 gives combined scatterplots of the elements of **B** versus the generating coefficients. The different plots correspond to different dimensionalities (numbers of variables) and different data sizes (numbers of data vectors), where each plot combines the data for a number of different network sparseness levels and non-linearities. Although for very small data sizes the estimation often fails, when the data size grows the estimation works practically flawlessly, as evidenced by the grouping of the data points onto the main diagonal.

In summary, the experiments verify the correctness of the method and demonstrate that reliable estimation is possible even with fairly limited amounts of data. We note that for larger dimensions we clearly need more data, but the amounts of data required are still reasonable.

7.2 Pruning Edges

We examined the performance of the pruning method developed in Section 6.3 using artificial data. The simulation consisted of 1000 trials. In each trial, we generated five- and ten-dimensional data of sample size $n = 1000, 5000, 10000$ in the same manner as in Section 7.1 above.

The LiNGAM discovery algorithm was then applied to the data. We subsequently applied the pruning method to the estimated networks. The significance level was set at 5%. Then we computed the numbers of correctly identified edges (true positives) and the numbers of correctly identified absence of edges (true negatives) only in the strictly lower triangular part of the matrix **B** to see the performance of our pruning method. We also counted how many edges were falsely added (false positives) and how many were falsely missing (false negatives).

	True pos.	False neg.	True neg.	False pos.	Sums of false pos. and neg.
Dim.=5					
n =1000	8101 (90.5%)	849 (9.5%)	921 (87.7%)	129 (12.3%)	978 (9.8%)
5000	8556 (95.6%)	394 (4.4%)	943 (89.8%)	107 (10.2%)	501 (5.0%)
10000	8691 (97.1%)	259 (2.9%)	972 (92.6%)	78 (7.4%)	337 (3.4%)
Dim.=10					
n =1000	27825 (80.6%)	6698 (19.4%)	8171 (78.0%)	2306 (22.0%)	9004 (20.0%)
5000	31623 (91.6%)	2900 (8.4%)	9350 (89.2%)	1127 (10.8%)	4027 (8.9%)
10000	32477 (94.1%)	2046 (5.9%)	9466 (90.4%)	1011 (9.6%)	3057 (6.8%)

Table 1: Numbers of true positives, false negatives, true negatives, and false positives (1000 trials). n is sample size.

The results are shown in Table 7.2. Some representative pruned estimated networks are shown in Figures 3 and 4.[6] First, we examine the numbers of false positives for the edges that had non-zero values. The false positive rates were approximately 10% except the case with sample size 1000 for both 5 and 10 variables. Second, we see the statistical power of the test (numbers of true positives)

6. Graphs were plotted using the latest version of the LiNGAM package which connects seamlessly to the free Graphviz software, a sophisticated tool for plotting graphs.

for the other edges that had non-zero values. The power of 0.90 (8055 true positives for 5 variables and 31071 true positives for 10 variables) was achieved for all the conditions other than when the number of variables was 10 and the sample size was 1000. Finally, we would mention that the sums of the two errors (false negatives and false positives) were small enough (less than 10%) except for the case with the number of variables 10 and the sample size 1000. Thus, Table 7.2 implied that our pruning method worked well for reasonable sample sizes.

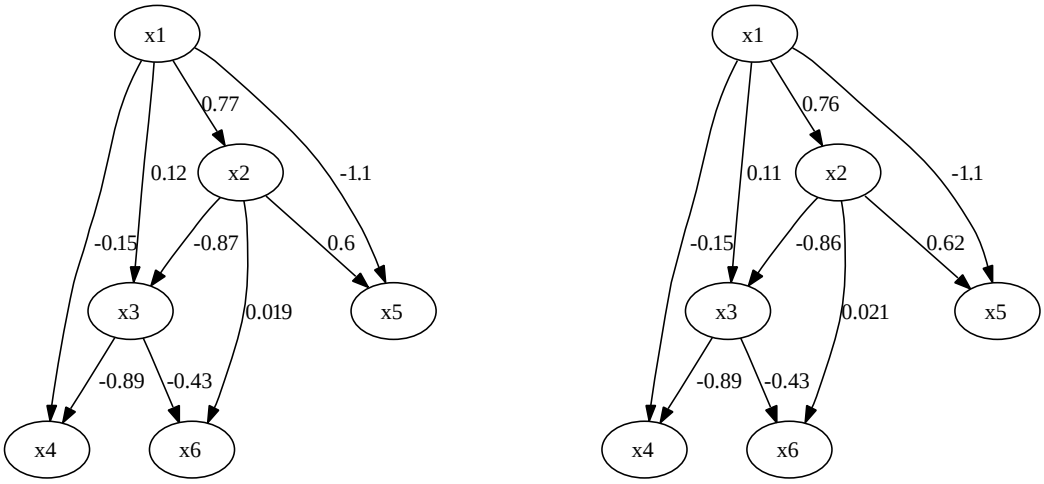


Figure 3: Left: example original network. Right: estimated network. The sample size was 10000. The structure of the original network was correctly estimated, and all the edge strengths were approximately correct.

8. Examples With Real-World Data

As a real-world example, we have applied the LiNGAM analysis to a set of time series. As a cause must precede its effect in time, we can expect the LiNGAM analysis to find the correct time ordering of the variables in any data generated from a LiNGAM model.

A time series can be approximated by a LiNGAM model if it is a stationary AR(p) process. An AR(p) process, or an *autoregressive process* of order p, is defined by the equation (Box and Jenkins, 1976)

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + a_t.$$

That is, the value of X_t is a weighted sum of p previous variables and white noise (a_t). The weights $\phi_1, \phi_2, \ldots, \phi_p$ are called the parameters of the process. A process is *stationary*, if the variance is finite, the mean remains the same over time, and the autocovariance function depends only on the time lag of two variables (Brockwell and Davis, 1987). The last condition also implies that the variance remains the same over time. If we want to approximate a stationary AR(p) process by a LiNGAM model, the white noise process must be non-Gaussian.

A time series must be presented to the LiNGAM analysis as a multivariate data set. To do this, the time series is divided into time windows with the same size as the number of variables in the

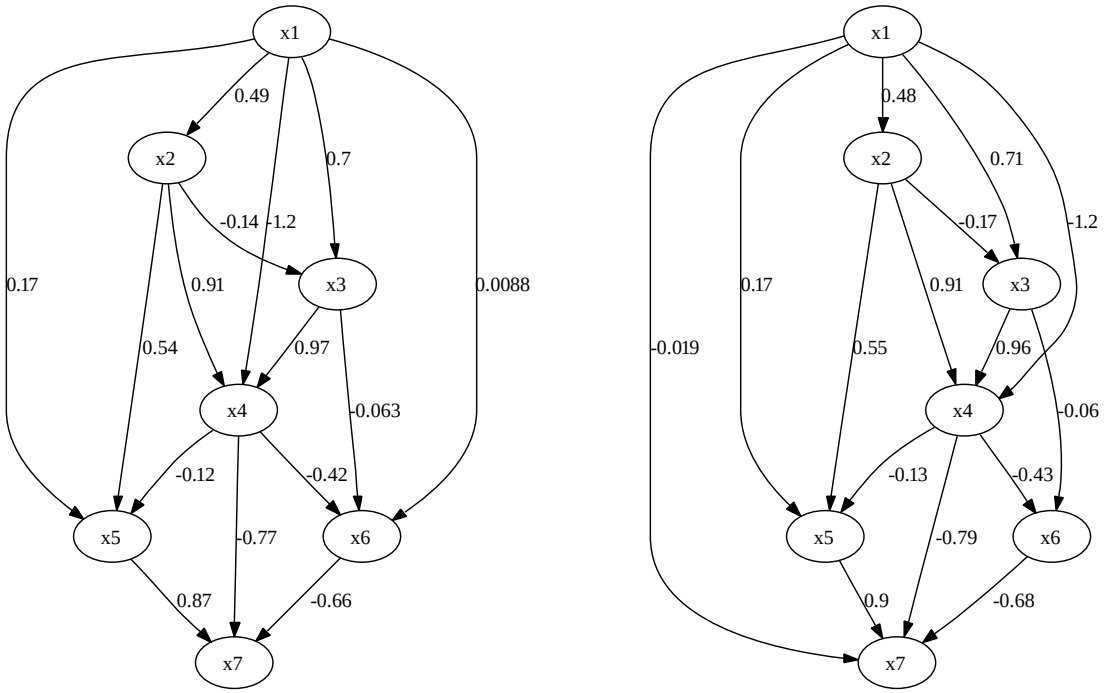


Figure 4: Left: example original network. Right: estimated network. The sample size was 10000. This shows what kind of mistakes the LiNGAM algorithm might make. The estimated network had one added edge ($x_1 \to x_7$) and one missing edge ($x_1 \to x_6$). However, both the added and missing edges had quite low strengths ($x_1 \to x_7$, -0.019 and $x_1 \to x_6$, 0.0088). Note that the other edges were correctly identified, and the connection strengths were approximately correct as well.

LiNGAM model. These time windows are then treated as samples of multivariate data. This introduces confounding variables to the model, against the assumptions of the LiNGAM analysis. To see this, consider an example of an AR(2) process, and a time window of three variables (Figure 5). Here, variables X_t and X_{t+1} are confounded by a variable outside the time window. In a general case, an AR(p) process introduces confounding variables to p first variables in a time window. The LiNGAM model holds strictly only for first order processes.

For the tests, a total of 22 data sets were selected from time series data repositories on the Internet (Hyndman, 2005; Statistical Software Information; National Statistics). We did not seek data sets that would fit the LiNGAM model, but a diverse set of data to see how well the LiNGAM analysis will perform with real-world data, when the assumptions of the model are violated at least to some extent. The data sets can be roughly categorized as economic time series and environmental time series. Economic time series included data sets like currency exchange rates and stock rates. Environmental time series included a more diverse set of data, ranging from monthly river-flows to daily temperatures. Before the tests, the sample autocorrelation and partial autocorrelation functions for the series were analyzed to gain insight into how well the series actually fit the AR(p) model.

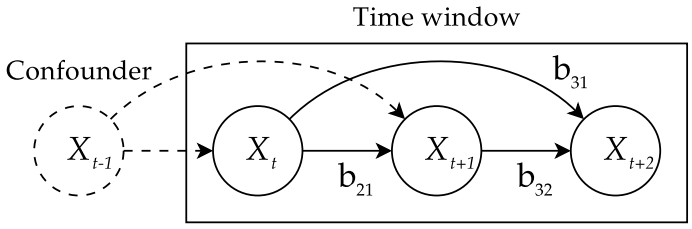


Figure 5: An example of confounding. Variables X_t and X_{t+1} are confounded by a variable outside the time window.

The LiNGAM analysis was run for each data set with varying parameters. The number of variables in the LiNGAM model was 3, 5, or 7, corresponding to AR(p) processes of orders 2, 4, and 6. Since the partial autocorrelations of economic time series indicated that the processes are at most second order processes, only models with 3 or 5 variables were tested for them. All possible time windows were used as multivariate data samples, including overlapping windows. For each number of variables, the LiNGAM analysis was run many times (100) to see if the analysis produced consistent results, where initial points of FastICA algorithm were randomized to assess the computational stability, that is, the effect of local minima.

Keeping in mind that there are possibly deviations from the LiNGAM model in the data, there are different possible results for applying the LiNGAM analysis. If the data generating process is indeed a stationary AR(p) process with non-Gaussian noise, we can expect to find the correct time ordering of the variables. An important nonstationary process, commonly encountered in economic time series, is the random walk $X_t = X_{t-1} + a_t$. If the generating process is a random walk, we cannot determine the direction of time. Hence, both the correct time ordering and the reverse time ordering are possible results. It is also possible that the variables are not causally related, and the weight matrix **B** is a matrix, none of which elements are significantly different from zero. In this case, any ordering of the variables is plausible. Also, any of the assumptions may fail to hold in real-world data: the process might be non-linear, the error terms might have Gaussian distribution, or there might be confounding variables. In this case, the LiNGAM analysis will probably fail, producing unpredictable results.

Reflecting the possibilities explained above, four distinct categories of results were distinguished from the tests.

1. The correct causal order was found (5 cases).
2. The reverse causal order was found (9 cases).
3. The **B** matrix was estimated as a zero matrix (1 case).
4. No consistent estimate for causal order was found (7 cases).

For the first three categories, the estimates were consistent over all experiments. The last category included data sets for which no consistent estimate was found. It also included some data sets for which a consistent estimate was found for some test setting, but not for all settings. An example from each category is provided for further analysis. The examples are listed Table 2, together with

their sample sizes and descriptions. Figure 6 plots the example series. The sample autocorrelation function (acf) and the partial autocorrelation function (pacf) are plotted for each example in Figure 7. The horizontal lines in pacf plots are 99% (solid line) and 95% (dashed line) confidence intervals for the null hypothesis that the partial autocorrelation is zero.

Name	Size	Description
PEAS	768	Monthly precipitation in Eastport, USA [mm].
DAILYIBM	3333	Daily closing price of IBM stock.
PRECIP	792	Daily precipitation in Hveravellir [mm].
MLCO2	372	Monthly carbon dioxide above Mauna Loa, Hawaii [parts per million].

Table 2: Sample sizes and descriptions of the example data sets.

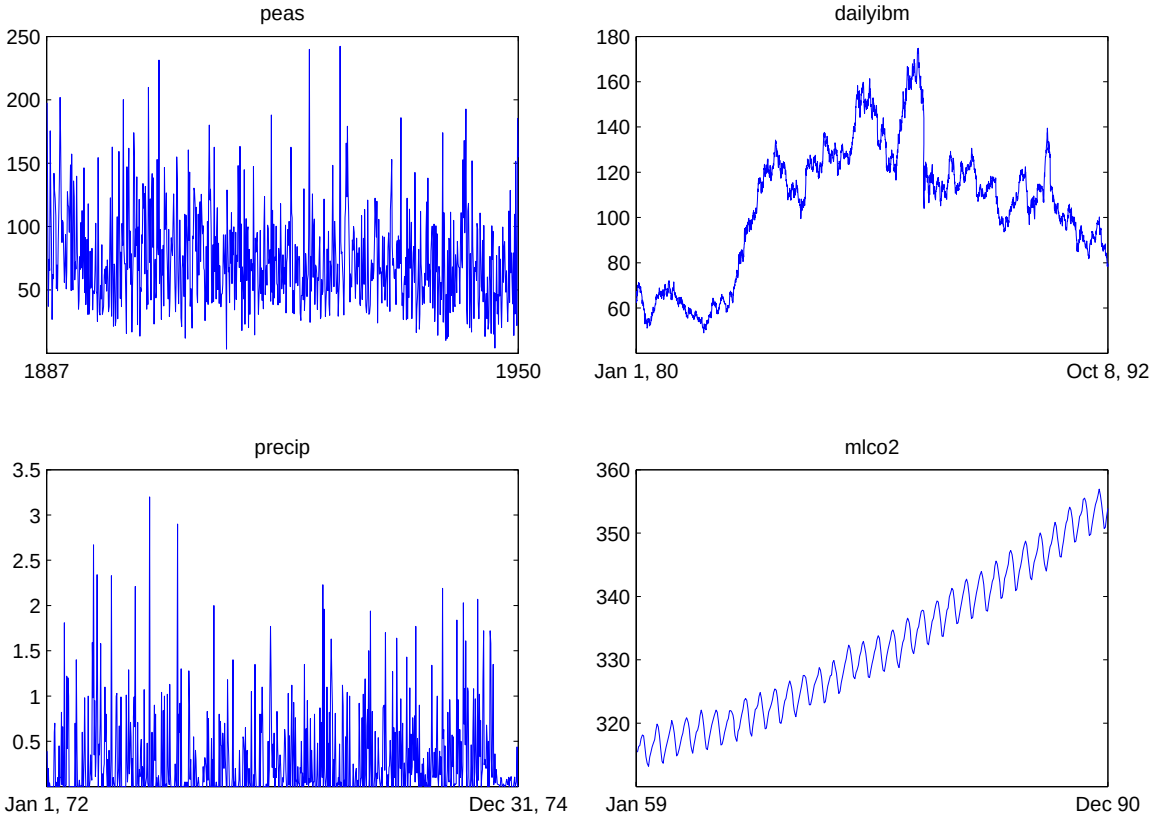


Figure 6: Plots for the example data sets.

The correct causal order was found from the PEAS data set. The statistical properties of the data indicate that the process could be modeled fairly well as an AR(2) process. There are no signs of a trend in the sample autocorrelation function, and the estimated partial correlations are significant for time lags 1 and 2. Still, there seems to be a small seasonal component, reflecting yearly seasonality

of precipitation. The most frequent model estimated by LiNGAM was an AR(2) process with small parameters, in accordance with the estimated partial correlations.

The reverse causal order was found from the DAILYIBM data set. The acf and the pacf of the data resemble those of a random walk, thus the result is expected. Also the estimated causal model is close to a random walk, values of nearly one are estimated consistently for the first parameter of the AR(p) process, other weight estimates being zero.

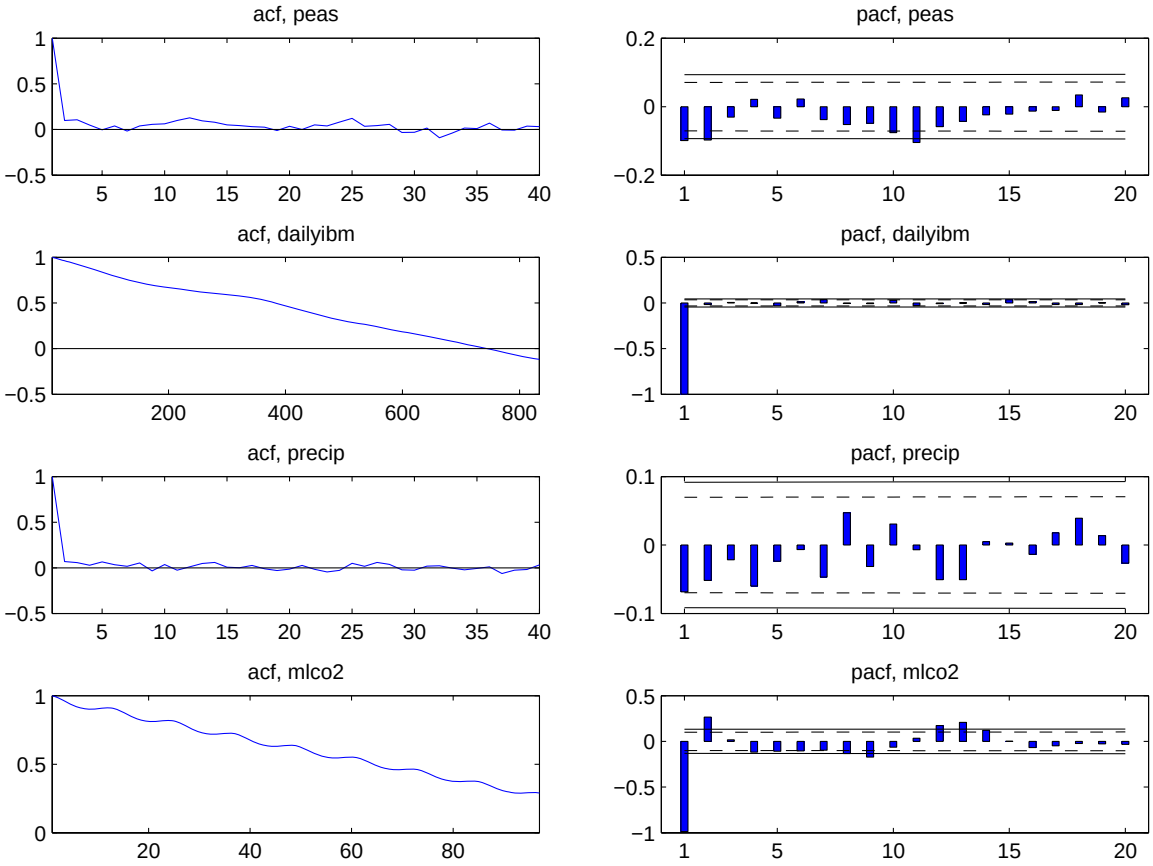


Figure 7: The sample autocorrelation and partial autocorrelation functions for the example data sets.

For the PRECIP data set, the estimates for causal order were consistent, but not consistently either correct or reverse. Most of the time, the estimated **B** matrix was a zero matrix. As a zero **B** matrix is consistent with any time ordering of the variables, this largely explains the results. For the cases when the **B** matrix was not a zero matrix, the estimated weights were small (less than 0.1), and did not correspond to an AR(p) model of any order, but rather random estimation errors. The pacf of this series supports the results of the LiNGAM analysis: none of the partial autocorrelations is statistically significant.

For the MLCO2 data set, the LiNGAM analysis produced inconsistent estimates of the causal order. There are several possible reasons for this. First of all, the series has a trend and a seasonal

component. More probably, the basic assumptions of the LiNGAM analysis are violated. The process might be non-linear, or the level of carbon dioxide might be caused by other environmental variables, leading to a confounded model. It is also possible, although unlikely, that the data is Gaussian.

9. Conclusions

Developing methods for causal inference from non-experimental data is a fundamental problem with a very large number of potential applications. Although one can never fully prove the validity of a causal model from observational data alone, such methods are nevertheless crucial in cases where it is impossible or very costly to perform experiments.

Previous methods developed for linear causal models (Bollen, 1989; Spirtes et al., 2000; Pearl, 2000) have been based on an explicit or implicit assumption of Gaussianity, and have hence been based solely on the covariance structure of the data. Because of this, additional information (such as the time-order of the variables) is usually required to obtain a full causal model of the variables. Without such information, algorithms based on the Gaussian assumption cannot in most cases distinguish between multiple equally possible causal models.

In this paper, we have shown that an assumption of non-Gaussianity of the disturbance variables, together with the assumption of linearity and causal sufficiency, allows the causal model to be completely identified. Furthermore, we have proposed a practical algorithm which estimates the causal structure under these assumptions and provided a number of tests to prune the graph and to see whether the estimated model fits the data.

The practical value of the LiNGAM analysis needs to be determined by applying it to real-world data sets and comparing it to other methods for causal inference from non-experimental data. The real data examples reported here are rather limited. Also, in many cases involving real-world data, practitioners in the field already have a fairly good understanding of the causal processes underlying the data. An interesting question is how well methods such as ours do on such data sets. These are important topics for future work.

Acknowledgments

This work was partially carried out at Division of Mathematical Science, Graduate School of Engineering Science, Osaka University and Transdisciplinary Research Integration Center, Research Organization of Information and Systems. The authors would like to thank Aristides Gionis, Heikki Mannila, and Alex Pothen for discussions relating to algorithms for solving the permutation problems, Niclas Börlin for contributing the Matlab code for solving the assignment problem, Yutaka Kano and Michiwo Kanekiyo for comments on the manuscript and Michael Jordan and three anonymous reviewers for useful comments to improve this paper. S.S. was supported by Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science. P.O.H. was supported by the Academy of Finland project #204826. A.H. was supported by the Academy of Finland through an Academy Research Fellow Position and project #203344.

Appendix A. Proof of Uniqueness of Row Permutation

Here, we show that, were the estimates of ICA exact, there is only a single permutation of the rows of $\mathbf{W}$ which results in a diagonal with no zero entries.

It is well-known (Bollen, 1989) that the DAG structure of the network guarantees that for some permutation of the variables, the matrix $\mathbf{B}$ is strictly lower-triangular. This implies that the correct $\widetilde{\mathbf{W}}$ (where the disturbance variables are aligned with the observed variables) can be permuted to lower-triangular form (with no zero entries on the diagonal) by equal row and column permutations, that is,

$$\widetilde{\mathbf{W}} = \mathbf{P}_d \mathbf{M} \mathbf{P}_d^T,$$

where $\mathbf{M}$ is lower-triangular and has no zero entries on the diagonal, and $\mathbf{P}_d$ is a permutation matrix representing a causal ordering of the variables. Now, ICA returns a matrix with randomly permuted rows,

$$\mathbf{W} = \mathbf{P}_{\text{ica}} \widetilde{\mathbf{W}} = \mathbf{P}_{\text{ica}} \mathbf{P}_d \mathbf{M} \mathbf{P}_d^T = \mathbf{P}_1 \mathbf{M} \mathbf{P}_2^T,$$

where $\mathbf{P}_{\text{ica}}$ is the random ICA row permutation, and on the right we have denoted by $\mathbf{P}_1 = \mathbf{P}_{\text{ica}} \mathbf{P}_d$ and $\mathbf{P}_2 = \mathbf{P}_d$, respectively, the row and column permutations from the lower triangular matrix $\mathbf{M}$.

We now prove that $\mathbf{W}$ has no zero entries on the diagonal if and only if the row and column permutations are equal, that is, $\mathbf{P}_1 = \mathbf{P}_2$. Hence, there is only one row permutation of $\mathbf{W}$ which yields no zero entries on the diagonal, and it is the one which finds the correspondence between the disturbance variables and the observed variables.

Lemma 1 *Assume* $\mathbf{M}$ *is lower triangular and all diagonal elements are non-zero. A permutation of rows and columns of* $\mathbf{M}$ *has only non-zero entries in the diagonal if and only if the row and column permutations are equal.*

Proof First, we prove that if the row and columns permutations are not equal, there will be zero elements in the diagonal.

Denote by $\mathbf{K}$ a lower triangular matrix of all ones in the lower triangular part. Denote by $\mathbf{P}_1$ and $\mathbf{P}_2$ two permutation matrices. The number of non-zero diagonal entries in a permuted version of $\mathbf{K}$ is $\text{tr}(\mathbf{P}_1 \mathbf{K} \mathbf{P}_2^T)$. This is the maximum number of non-zero diagonal entries when an arbitrary lower triangular matrix is permuted.

We have $\text{tr}(\mathbf{P}_1 \mathbf{K} \mathbf{P}_2^T) = \text{tr}(\mathbf{K} \mathbf{P}_2^T \mathbf{P}_1)$. Thus, we can consider permutations of columns only, given by $\mathbf{P}_2^T \mathbf{P}_1$. Assume the columns of $\mathbf{K}$ are permuted so that the permutation is not equal to identity. Then, there exists an index i so that the column of index i has been moved to column index j where $j < i$ (If there were no such columns, all the columns would be moved to the right, which is impossible.) Obviously, the diagonal entry in the j-th column in the permuted matrix is zero. Thus, any column permutation not equal to the identity creates at least one zero entry in the diagonal.

Thus, to have non-zero diagonal, we must have $\mathbf{P}_2^T \mathbf{P}_1 = \mathbf{I}$. This means that the column and row permutations must be equal.

Next, assume that the row and column permutations are equal. Consider $\mathbf{M} = \mathbf{I}$ as a worst-case scenario. Then the permuted matrix equals $\mathbf{P}_1 \mathbf{I} \mathbf{P}_2^T$ which equals identity, and all the diagonal elements are non-zero. Adding more non-zero elements in the matrix only increases the number of non-zero elements in the permuted version.

Thus, the lemma is proven.

Appendix B. An Example of the Permutation Problem

Here we show with an example that if the permutation is not correctly determined, the parameters b_{ij} can have very different values, yet give the same data distribution. This example considers the general case where the system is not DAG. For simplicity, let us consider the two variables case. Assume we parameterize the mixing model in (2) as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \frac{1}{1-b_{12}b_{21}} \begin{bmatrix} 1 & b_{12} \\ b_{21} & 1 \end{bmatrix} \mathrm{diag}(\sigma_1,\sigma_2) \begin{bmatrix} s_1 \\ s_2 \end{bmatrix},$$

where σ_1 and σ_2 are standard deviations of e_1 and e_2, and s_1 and s_2 are normalized versions of e_1 and e_2, that is, e_1/σ_1 and e_2/σ_2.

Then, take the following new set of parameters:

$$\begin{aligned} b'_{12} &= 1/b_{21} \\ b'_{21} &= 1/b_{12} \\ \sigma'_1 &= \sigma_2/b_{21} \\ \sigma'_2 &= \sigma_1/b_{12}, \end{aligned}$$

and do the permutation and sign change:

$$\begin{aligned} s'_1 &= -s_2 \\ s'_2 &= -s_1. \end{aligned}$$

Then, the two parameterizations give the same data, that is, the same model fit. This is because

$$\begin{aligned} &\frac{1}{1-b'_{12}b'_{21}} \begin{bmatrix} 1 & b'_{12} \\ b'_{21} & 1 \end{bmatrix} \mathrm{diag}(\sigma'_1,\sigma'_2) \begin{bmatrix} s'_1 \\ s'_2 \end{bmatrix} \\ &= \frac{1}{1-1/(b_{12}b_{21})} \begin{bmatrix} 1 & 1/b_{21} \\ 1/b_{12} & 1 \end{bmatrix} \mathrm{diag}(\sigma_2/b_{21},\sigma_1/b_{12}) \begin{bmatrix} -s_2 \\ -s_1 \end{bmatrix} \\ &= \frac{-b_{12}b_{21}}{1-b_{12}b_{21}} \begin{bmatrix} 1/b_{21} & 1/(b_{21}b_{12}) \\ 1/(b_{12}b_{21}) & 1/b_{12} \end{bmatrix} \mathrm{diag}(\sigma_2,\sigma_1) \begin{bmatrix} -s_2 \\ -s_1 \end{bmatrix} \\ &= \frac{b_{12}b_{21}}{1-b_{12}b_{21}} \begin{bmatrix} 1/b_{21} & 1/(b_{21}b_{12}) \\ 1/(b_{12}b_{21}) & 1/b_{12} \end{bmatrix} \mathrm{diag}(\sigma_2,\sigma_1) \begin{bmatrix} s_2 \\ s_1 \end{bmatrix} \\ &= \frac{1}{1-b_{12}b_{21}} \begin{bmatrix} b_{12} & 1 \\ 1 & b_{21} \end{bmatrix} \mathrm{diag}(\sigma_2,\sigma_1) \begin{bmatrix} s_2 \\ s_1 \end{bmatrix} \\ &= \frac{1}{1-b_{12}b_{21}} \begin{bmatrix} 1 & b_{12} \\ b_{21} & 1 \end{bmatrix} \mathrm{diag}(\sigma_1,\sigma_2) \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}. \end{aligned}$$

Therefore, the parameter sets with or without "prime" are equivalent. The model fit is the same with two different sets of parameters. Estimation of the model can equally well give any of these two sets. However, the numerical values of b and b' are quite different. In this example, the system was not constrained to be a DAG. In fact, if the original system is a DAG, the system with primes has infinite coefficients. Thus, we see how the constraint of DAG is helpful.

Appendix C. ML Derivation of Objective Function for Finding the Correct Row Permutation

Since the ICA estimates are never exact, all elements of $\mathbf{W}$ will be non-zero, and one cannot base the permutation on exact zeros. Here we show that the objective function for step 2 of the LiNGAM algorithm can be derived from a maximum likelihood framework.

Let us denote by e_{it} the value of disturbance variable i for the t-th data vector of the data set. Assume that we model the disturbance variables e_{it} by a generalized Gaussian density:

$$\log p(e_{it}) = -|e_{it}|^{\alpha}/\beta + Z,$$

where the α, β are parameters and Z is a normalization constant. Then, the log-likelihood of the model equals

$$\sum_t \sum_i -\frac{1}{\beta}\left|\frac{e_{it}}{w_{ii}}\right|^{\alpha} = -\sum_i \frac{1}{\beta |w_{ii}|^{\alpha}} \sum_t |e_{it}|^{\alpha},$$

because each row of $\mathbf{W}$ is subsequently divided by its diagonal element. To maximize the likelihood, we find the permutation of rows for which the diagonal elements maximize this term. For simplicity, assuming that the pdf's of all independent components are the same, this means we solve

$$\min_{\text{all row perms}} \sum_i \frac{1}{|w_{ii}|^{\alpha}}.$$

In principle, we could estimate α from the data using ML estimation as well, but for simplicity we fix it to unity because it does not really change the qualitative behavior of the objective function. Regardless of its value, this objective function heavily penalizes small values on the diagonal, as we intuitively (based on the argumentation in Section 4) require.

Appendix D. Asymptotic Variance of ICA

Several authors studied asymptotic variance of ICA (Pham and Garrat, 1997; Hyvärinen, 1997; Cardoso and Laheld, 1996; Tichavský et al., 2006), where the theory of estimating functions (Godambe, 1991) was often used. Let us consider a semiparametric model $p(\mathbf{x}|\theta)$, where θ is a r-dimensional parameter vector of interest. Note that the density function $p(\mathbf{x}|\theta)$ is unknown. Let us denote by θ_0 the true parameter vector of interest. A r-dimensional vector-valued function $f(\mathbf{x}, \theta)$ is called an estimating function when it satisfies the following conditions for any $p(\mathbf{x}|\theta_0)$:

$$\begin{aligned} E[f(\mathbf{x}, \theta_0)] &= \mathbf{0} \\ |\det \mathbf{J}| \neq 0, &\quad \text{where } \mathbf{J} = E\left[\frac{\partial}{\partial \theta^T} f(\mathbf{x}, \theta)\Big|_{\theta=\theta_0}\right] \\ E[\|f(\mathbf{x}, \theta_0)\|^2] &< \infty, \end{aligned}$$

where the expectation E is taken over $\mathbf{x}$ with respect to $p(\mathbf{x}|\theta_0)$.

Let $\mathbf{x}(1), \cdots, \mathbf{x}(n)$ be a random sample from $p(\mathbf{x}|\theta_0)$. Then an estimator $\widehat{\theta}$ is obtained by solving the estimating equation:

$$\sum_{i=1}^{n} f(\mathbf{x}(i), \theta) = \mathbf{0}.$$

Under some regularity conditions including identification conditions for θ, the estimator $\widehat{\theta}$ is consistent when n goes to infinity and asymptotically distributes according to the Gaussian distribution $N(\theta_0, \mathbf{G})$, and

$$\mathbf{G} = \frac{1}{n}\mathbf{J}^{-1}E[f(\mathbf{x},\theta_0)f^T(\mathbf{x},\theta_0)]\mathbf{J}^{-T}. \tag{5}$$

Pham and Garrat (1997) derived an estimating function for (quasi-) maximum likelihood estimation. Kawanabe and Müller (2005) provided estimating functions for JADE (Cardoso and Souloumiac, 1993) and for ICA based on non-Gaussianity maximization with orthogonality (uncorrelatedness) constraints including FastICA (Hyvärinen, 1999).

In this paper, we restrict ourselves to testing mixing and demixing coefficients estimated by FastICA. In the FastICA, we first center the data to make its mean zero and whiten the data by computing a matrix $\mathbf{V}$ such that the covariance matrix of $z = \mathbf{V}\mathbf{x}$ is the identity matrix. After that, we find an orthogonal matrix $\mathbf{Q}$ so that components of $\mathbf{Q}^T z = \mathbf{Q}^T\mathbf{V}\mathbf{x}$ have maximum non-Gaussianity. Then we obtain estimates of $\mathbf{A}$ and $\mathbf{W}$ by $\mathbf{A} = \mathbf{V}^{-1}\mathbf{Q}$ and $\mathbf{W} = \mathbf{Q}^T\mathbf{V}$.

Let us consider the following function:

$$\mathbf{F}(\mathbf{x},\mathbf{Q}) = yy^T - \mathbf{I} + yg^T(y) - g(y)y^T,$$

where $y = \mathbf{W}\mathbf{x} = \mathbf{Q}^T\mathbf{V}\mathbf{x} = \mathbf{Q}^T z$ and $g(u)$ is the non-linearity. The estimating function for FastICA is obtained as $f = \mathrm{vec}(\mathbf{F})$ taking $\theta = \mathrm{vec}(\mathbf{Q})$ (Kawanabe and Müller, 2005). Here, $\mathrm{vec}(\cdot)$ denotes the vectorization operator which creates a column vector from a matrix by stacking its columns.

According to the estimating function theory, we obtain the asymptotic covariance matrix of $\mathrm{vec}(\mathbf{Q})$ by (5). Here we assume that the variance in the estimate of $\mathbf{V}$ is negligible with respect to the variance in $\mathbf{Q}$. Then we obtain the asymptotic covariance matrix of $\mathrm{vec}(\mathbf{A})$ and $\mathrm{vec}(\mathbf{W})$ as follows:

$$\begin{aligned}
\mathrm{acov}\{\mathrm{vec}(\mathbf{A})\} &= \mathrm{acov}\{\mathrm{vec}(\mathbf{V}^{-1}\mathbf{Q})\} \\
&= (\mathbf{I}\otimes\mathbf{V}^{-1})\mathrm{acov}\{\mathrm{vec}(\mathbf{Q})\}(\mathbf{I}\otimes\mathbf{V}^{-1})^T \\
\mathrm{acov}\{\mathrm{vec}(\mathbf{W})\} &= \mathrm{acov}\{\mathrm{vec}(\mathbf{Q}^T\mathbf{V})\} \\
&= (\mathbf{V}^T\otimes\mathbf{I})\mathrm{acov}\{\mathrm{vec}(\mathbf{Q}^T)\}(\mathbf{V}^T\otimes\mathbf{I})^T,
\end{aligned}$$

where $\otimes$ denotes the Kronecker product.[7]

The formula of $\mathrm{acov}\{\mathrm{vec}(\mathbf{Q})\}$ for FastICA is written as

$$\mathrm{acov}\{\mathrm{vec}(\mathbf{Q})\} = \frac{1}{n}\mathbf{J}^{-1}E[\mathrm{vec}\{\mathbf{F}(\mathbf{x},\mathbf{Q})\}\mathrm{vec}\{\mathbf{F}(\mathbf{x},\mathbf{Q})\}^T]\mathbf{J}^{-T}.$$

Let us denote by F_{pq} and $\mathbf{F}_q$ the (p,q)-element and the q-th column of $\mathbf{F}$, respectively. We shall provide $E(F_{pq}F_{rs})$ to compute $E\{\mathrm{vec}(\mathbf{F})\mathrm{vec}(\mathbf{F})^T\}$. Denote by i,j,k,l four different subscripts. Then

7. The Kronecker product $\mathbf{Y}\otimes\mathbf{Z}$ of matrices $\mathbf{Y}$ and $\mathbf{Z}$ is defined as a partitioned matrix with (i,j)-th block equal to $y_{ij}\mathbf{Z}$.

we have

$$\begin{aligned}
E(F_{ii}F_{ii}) &= E(s_i^4)+1,\ E(F_{ii}F_{jj})=2,\ E(F_{ki}F_{ij})=-E\{g(s_k)\}E\{g(s_j)\}\\
E(F_{ki}F_{li}) &= E\{g(s_k)\}E\{g(s_l)\},\ E(F_{ki}F_{kj})=E\{g(s_i)\}E\{g(s_j)\}\\
E(F_{ii}F_{li}) &= -E(s_i^3)E\{g(s_l)\},\ E(F_{ki}F_{ii})=-E(s_i^3)E\{g(s_k)\}\\
E(F_{ii}F_{ij}) &= E(s_i^3)E\{g(s_j)\},\ E(F_{ji}F_{jj})=E(s_j^3)E\{g(s_i)\}\\
E(F_{ii}F_{lj}) &= 0,\ E(F_{ki}F_{jj})=0,\ E(F_{ki}F_{lj})=0\\
E(F_{ji}F_{ij}) &= 1+2E\{s_ig(s_i)\}E\{s_jg(s_j)\}-E\{g(s_i)^2\}-E\{g(s_j)^2\}\\
E(F_{ji}F_{lj}) &= 1+E\{s_ig(s_i)\}+E\{s_jg(s_j)\}+E\{s_ig(s_i)\}E\{s_jg(s_j)\}-E\{g(s_i)\}E\{g(s_l)\}\\
E(F_{ki}F_{ki}) &= 1+2E\{s_ig(s_i)\}-2E\{s_kg(s_k)\}+E\{g(s_i)^2\}+E\{g(s_k)^2\}\\
&\quad -2E\{s_ig(s_i)\}E\{s_kg(s_k)\}.
\end{aligned}$$

Further we shall give $E\left\{(\partial\mathbf{F}_i)/(\partial q_j^T)\right\}$ to compute $\mathbf{J}=E\left[\{\partial\mathrm{vec}(\mathbf{F})\}/\{\partial\mathrm{vec}(\mathbf{Q})^T\}\right]$:

$$\begin{aligned}
E\left[\frac{\partial\mathbf{F}_i}{\partial q_i^T}\right] &= \begin{cases} 2E(q_i^Tzz^T) \\ E\{q_k^Tzz^T-z^Tg(q_k^Tz)+q_k^Tzg'(q_i^Tz)z^T\}\end{cases}\\
&= \begin{cases} 2q_i^T & (i\text{−th row})\\ [1-E\{s_kg(s_k)\}+E\{g'(s_i)\}]\,q_k^T & (k\text{−th row},\ k\neq i)\end{cases}\\
E\left[\frac{\partial\mathbf{F}_i}{\partial q_j^T}\right] &= \begin{cases} E[\{1-g'(q_j^Tz)\}q_i^Tzz^T+z^Tg(q_i^Tz)] \\ \mathbf{0}^T\end{cases}\\
&= \begin{cases} [1-E\{g'(s_j)\}+E\{s_ig(s_i)]q_i^T & (j\text{−th row},\ j\neq i)\\ \mathbf{0}^T & (k\text{−th row},\ k\neq j)\end{cases}.
\end{aligned}$$

Appendix E. Exact Form of $\mathbf{J}=\partial\sigma_2(\tau)/\partial\tau^T$

We here derive the exact form of $\mathbf{J}=\partial\sigma_2(\tau)/\partial\tau^T$ in (4). Let us denote by Σ_2 the covariance matrix based on the model or $E(\mathbf{x}\mathbf{x}^T)$. The $\sigma_2(\tau)$ in (3) is obtained by $\mathrm{vec}^+(\Sigma_2)$. Let us rewrite the LiNGAM model as:

$$\begin{aligned}
\mathbf{x} &= (\mathbf{I}-\mathbf{B})^{-1}\mathbf{e}\\
&= (\mathbf{I}-\mathbf{B})^{-1}\mathbf{D}^{\frac{1}{2}}\widetilde{\mathbf{e}},
\end{aligned}$$

where $\mathbf{D}=\mathrm{cov}(\mathbf{e})$ and $\widetilde{\mathbf{e}}=\mathbf{D}^{-\frac{1}{2}}\mathbf{e}$. Note that $E(\mathbf{e})=0$ is assumed. Then the model-based covariance matrix Σ is:

$$\begin{aligned}
\Sigma &= (\mathbf{I}-\mathbf{B})^{-1}\mathbf{D}^{\frac{1}{2}}\mathrm{cov}(\widetilde{\mathbf{e}})\mathbf{D}^{\frac{1}{2}}(\mathbf{I}-\mathbf{B})^{-T}\\
&= \left\{\mathbf{D}^{-\frac{1}{2}}(\mathbf{I}-\mathbf{B})\right\}^{-1}\left\{\mathbf{D}^{-\frac{1}{2}}(\mathbf{I}-\mathbf{B})\right\}^{-T}\\
&= \mathbf{Y}\mathbf{Y}^T,
\end{aligned}$$

where

$$\mathbf{Y} = \left\{\mathbf{D}^{-\frac{1}{2}}(\mathbf{I}-\mathbf{B})\right\}^{-1}.$$

Now we need to compute the following derivatives:

$$\begin{aligned}\frac{\partial \Sigma_{ij}}{\partial b_{kl}} &= \frac{\partial (\mathbf{Y}\mathbf{Y}^T)_{ij}}{\partial b_{kl}} = \sum_p \sum_q \frac{\partial (\mathbf{Y}\mathbf{Y}^T)_{ij}}{\partial \mathbf{Y}_{pq}} \frac{\partial \mathbf{Y}_{pq}}{\partial b_{kl}} \\ \frac{\partial \Sigma_{ij}}{\partial d_{kk}} &= \frac{\partial (\mathbf{Y}\mathbf{Y}^T)_{ij}}{\partial d_{kk}} = \sum_p \sum_q \frac{\partial (\mathbf{Y}\mathbf{Y}^T)_{ij}}{\partial \mathbf{Y}_{pq}} \frac{\partial \mathbf{Y}_{pq}}{\partial d_{kk}}.\end{aligned}$$

We provide $\partial(\mathbf{Y}\mathbf{Y}^T)_{ij}/\partial \mathbf{Y}_{pq}$, $\partial \mathbf{Y}_{pq}/\partial b_{kl}$, and $\partial \mathbf{Y}_{pq}/\partial d_{kk}$ to compute the derivatives above:

$$\begin{aligned}\frac{\partial (\mathbf{Y}\mathbf{Y}^T)_{ij}}{\partial \mathbf{Y}_{pq}} &= \begin{cases} 2\mathbf{Y}_{pq} & (i=p,\ j=p) \\ \mathbf{Y}_{jq} & (i=p,\ j\neq p) \\ \mathbf{Y}_{iq} & (i\neq p,\ j=p) \\ 0 & (i\neq p,\ j\neq p) \end{cases} \\ \frac{\partial \mathbf{Y}}{\partial b_{kl}} &= -\mathbf{Y}\frac{\partial \mathbf{Y}^{-1}}{\partial b_{kl}}\mathbf{Y} \\ &= \mathbf{Y}\mathbf{D}^{-\frac{1}{2}}\mathbf{J}^{kl}\mathbf{Y} \\ \frac{\partial \mathbf{Y}}{\partial d_{kk}} &= -\mathbf{Y}\frac{\partial \mathbf{Y}^{-1}}{\partial d_{kk}}\mathbf{Y} \\ &= \mathbf{Y}\frac{d_{kk}^{-3/2}}{2}\mathbf{J}^{kk}(\mathbf{I}-\mathbf{B})\mathbf{Y},\end{aligned}$$

where $\mathbf{J}^{kl}$ is the single-entry matrix with 1 at (k,l) and zero elsewhere. Thus, we can compute $\mathbf{J} = \partial \sigma_2/\partial \tau^T = \partial \mathrm{vec}^+(\Sigma_2)/\partial \tau^T$.

References

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57:289–300, 1995.

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

G. E. P. Box and G. M. Jenkins. *Time Series Analysis: forecasting and control*. Holden-Day, Oakland, California, USA, revised edition, 1976.

P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer-Verlag, New York, USA, 1987.

M. W. Browne. Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, 9:665–672, 1984.

R. E. Burkard and E. Cela. Linear assignment problems and extensions. In P. M. Pardalos and D. Z. Du, editors, *Handbook of Combinatorial Optimization - Supplement Volume A*, pages 75–149. Kluwer, 1999.

J.-F. Cardoso and B. H. Laheld. Equivariant adaptive source separation. *IEEE Trans. on Signal Processing*, 44:3017–3030, 1996.

J.-F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings-F*, 140(6):362–370, 1993.

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

Y. Dodge and V. Rousson. On asymptotic properties of the correlation coefficient in the regression setting. *The American Statistician*, 55(1):51–54, 2001.

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

D. Geiger and D. Heckerman. Learning gaussian networks. In *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 235–243, 1994.

V. P. Godambe. *Estimating functions*. Oxford University Press, New York, 1991.

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

Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 4: 800–802, 1988.

Y. Hochberg and A. C. Tamhane. *Multiple comparison procedures*. John Wiley & Sons, New York, 1987.

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

P. O. Hoyer, S. Shimizu, A. Hyvärinen, Y. Kano, and A. J. Kerminen. New permutation algorithms for causal discovery using ICA. In *Proceedings of International Conference on Independent Component Analysis and Blind Signal Separation, Charleston, SC, USA*, pages 115–122, 2006a.

P. O. Hoyer, S. Shimizu, and A. J. Kerminen. Estimation of linear, non-gaussian causal models in the presence of confounding latent variables. In *Proc. the third European Workshop on Probabilistic Graphical Models (PGM2006)*, 2006b. In press.

L. Hu, P. M. Bentler, and Y. Kano. Can test statistics in covariance structure analysis be trusted? *Psychological Bulletin*, 112:351–362, 1992.

R. J. Hyndman. Time series data library, 2005. URL `http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/`. [June 2005].

A. Hyvärinen. One-unit contrast functions for independent component analysis: A statistical analysis. In *Neural Networks for Signal Processing VII (Proceedings of IEEE Workshop on Neural Networks for Signal Processing)*, pages 388–397, 1997.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. on Neural Networks*, 10(3):626–634, 1999.

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

M. Kawanabe and K. R. Müller. Estimating functions for blind separation when sources have variance dependencies. *Journal of Machine Learning Research*, 6:453–482, 2005.

National Statistics, 2005. URL `http://www.statistics.gov.uk/`. [June 2005].

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

D. T. Pham and P. Garrat. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *Signal Processing*, 45:1457–1482, 1997.

S. Shimizu, A. Hyvärinen, P. O. Hoyer, and Y. Kano. Finding a causal ordering via independent component analysis. *Computational Statistics & Data Analysis*, 50(11):3278–3293, 2006a.

S. Shimizu, A. Hyvärinen, Y. Kano, and P. O. Hoyer. Discovery of non-gaussian linear causal models using ICA. In *Proc. the 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)*, pages 526–533, 2005.

S. Shimizu, A. Hyvärinen, Y. Kano, P. O. Hoyer, and A. J. Kerminen. Testing significance of mixing and demixing coefficients in ICA. In *Proceedings of International Conference on Independent Component Analysis and Blind Signal Separation, Charleston, SC, USA*, pages 901–908, 2006b.

S. Shimizu and Y. Kano. Use of non-normality in structural equation modeling: Application to direction of causation. *Journal of Statistical Planning and Inference*, 2006. In press.

R. J. Simes. An improved Bonferroni procedure for multiple tests of significance. *Biometrika*, 73: 751–754, 1986.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search, 2nd ed.* MIT Press, 2000.

Statistical Software Information, 2005. URL `http://www-unix.oit.umass.edu/~statdata/`. [June 2005].

P. Tichavský, Z. Koldovský, and E. Oja. Performance analysis of the FastICA algorithm and Cramèr-Rao bounds for linear independent component analysis. *IEEE Trans. on Signal Processing*, 54(4):1189–1203, 2006.

K-H. Yuan and P. M. Bentler. Mean and covariance structure analysis: Theoretical and practical improvements. *Journal of the American Statistical Association*, 92(438):767–774, 1997.

Journal of Machine Learning Research 7 (2006) 2031-2064 Submitted 4/06; Revised 8/06; Published 10/06

Walk-Sums and Belief Propagation in Gaussian Graphical Models*

Dmitry M. Malioutov DMM@MIT.EDU
Jason K. Johnson JASONJ@MIT.EDU
Alan S. Willsky WILLSKY@MIT.EDU
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Editor: Michael I. Jordan

Abstract

We present a new framework based on walks in a graph for analysis and inference in Gaussian graphical models. The key idea is to decompose the correlation between each pair of variables as a sum over all walks between those variables in the graph. The weight of each walk is given by a product of edgewise partial correlation coefficients. This representation holds for a large class of Gaussian graphical models which we call walk-summable. We give a precise characterization of this class of models, and relate it to other classes including diagonally dominant, attractive, non-frustrated, and pairwise-normalizable. We provide a walk-sum interpretation of Gaussian belief propagation in trees and of the approximate method of loopy belief propagation in graphs with cycles. The walk-sum perspective leads to a better understanding of Gaussian belief propagation and to stronger results for its convergence in loopy graphs.

Keywords: Gaussian graphical models, walk-sum analysis, convergence of loopy belief propagation

1. Introduction

We consider multivariate Gaussian distributions defined on undirected graphs, which are often referred to as Gauss-Markov random fields (GMRFs). The nodes of the graph denote random variables and the edges capture the statistical dependency structure of the model. The family of all Gauss-Markov models defined on a graph is naturally represented in the *information form* of the Gaussian density. The key parameter of the information form is the *information matrix*, which is the inverse of the covariance matrix. The information matrix is sparse, reflecting the structure of the defining graph such that only the diagonal elements and those off-diagonal elements corresponding to edges of the graph are non-zero.

Given such a model, we consider the problem of computing the mean and variance of each variable, thereby determining the marginal densities as well as the mode. In principle, these can be obtained by inverting the information matrix, but the complexity of this computation is cubic in the number of variables. More efficient recursive calculations are possible in graphs with very sparse

*. This paper elaborates upon our earlier brief publication (Johnson, Malioutov, and Willsky, 2006) and presents subsequent developments. This research was supported by the Air Force Office of Scientific Research under Grants FA9550-04-1-0351, FA9550-06-1-0324, and the Army Research Office under Grant W911NF-05-1-0207. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the Air Force or Army.

structure—for example, in chains, trees and in graphs with "thin" junction trees. For these models, belief propagation (BP) or its junction tree variants efficiently compute the marginals (Pearl, 1988; Cowell et al., 1999). In large-scale models with more complex graphs, for example, for models arising in oceanography, 3D-tomography, and seismology, even the junction tree approach becomes computationally prohibitive. Iterative methods from numerical linear algebra (Varga, 2000) can be used to compute the marginal means. However, in order to efficiently compute both means and variances, approximate methods such as loopy belief propagation (LBP) are needed (Pearl, 1988; Yedidia, Freeman, and Weiss, 2003; Weiss and Freeman, 2001; Rusmevichientong and Van Roy, 2001). Another important motivation for using LBP, emphasized for example by Moallemi and Van Roy (2006a), is its distributed nature which is important for applications such as sensor networks. While LBP has been shown to often provide good approximate solutions for many problems, it is not guaranteed to do so in general, and may even fail to converge.

In prior work, Rusmevichientong and Van Roy (2001) analyzed Gaussian LBP on the turbo-decoding graph. For this special case they established that variances converge, means follow a linear system upon convergence of the variances, and that if means converge then they are correct. Weiss and Freeman (2001) analyzed LBP from the computation tree perspective to give a sufficient condition (equivalent to diagonal dominance of the information matrix) for convergence, and also showed correctness of the means upon convergence. Wainwright et al. (2003) introduced the tree reparameterization view of belief propagation and, in the Gaussian case, also showed correctness of the means upon convergence. Convergence of other forms of LBP are analyzed by Ihler et al. (2005), and Mooij and Kappen (2005), but unfortunately their sufficient conditions are not directly applicable to the Gaussian case.

We develop a "walk-sum" formulation for computation of means, variances and correlations as sums over certain sets of weighted walks in a graph.[1,2] This walk-sum formulation applies to a wide class of Gauss-Markov models which we call *walk-summable*. We characterize the class of walk-summable models and show that it contains (and extends well beyond) some "easy" classes of models, including models on trees, attractive, non-frustrated, and diagonally dominant models. We also show the equivalence of walk-summability to the fundamental notion of pairwise-normalizability, and that inference in walk-summable models can be reduced to inference in an attractive model based on a certain extended graph.

We use the walk-sum formulation to develop a new interpretation of BP in trees and of LBP in general. Based on this interpretation we are able to extend the previously known sufficient conditions for convergence of LBP to the class of walk-summable models. Our sufficient condition is stronger than that given by Weiss and Freeman (2001) as the class of diagonally dominant models is a strict subset of the class of pairwise-normalizable models. Our results also explain why they did not find any examples where LBP does not converge. The reason is that they presumed pairwise-normalizability. We also give a new explanation, in terms of walk-sums, of why LBP converges to the correct means but not to the correct variances. The reason is that LBP captures all of the walks needed to compute the means but only computes a subset of the walks needed for the variances.

1. After submitting the paper we became aware of a related decomposition for non-Gaussian classical spin systems in statistical physics developed by Brydges et al. (1983). Similarly to our work, the decomposition is connected to the Neumann series expansion of the matrix inverse, but in addition to products of edge weights, their weight of a walk includes a complicated multi-dimensional integral.
2. Another interesting decomposition of the covariance in Gaussian models in terms of *path sums* has been proposed in Jones and West (2005). It is markedly different from our approach (e.g., unlike paths, walks can cross an edge multiple times, and the weight of a path is rather hard to calculate, as opposed to our walk-weights).

In general, walk-summability is not necessary for LBP convergence. Hence, we also provide a tighter (essentially necessary) condition for convergence of LBP *variances* based on a weaker form of walk-summability defined on the LBP computation tree. This provides deeper insight into why LBP can fail to converge—because the LBP computation tree is not always well-posed—which suggests connections to Tatikonda and Jordan (2002).

In related work, concurrent with Johnson et al. (2006), Moallemi and Van Roy (2006a) have shown convergence of their consensus propagation algorithm, which uses a pairwise-normalized model. In this paper, we demonstrate the equivalence of pairwise-normalizability and walk-summability, which suggests a connection between their results and ours. In their more recent work (Moallemi and Van Roy, 2006b), concurrent with this paper, they make use of our walk-sum analysis of LBP, assuming pairwise-normalizability, to consider other initializations of the algorithm.[3] However, the critical condition is still walk-summability, which is presented in this paper.

In Section 2 we introduce Gaussian graphical models and describe exact BP for tree-structured graphs as well as approximate BP for loopy graphs, and their connection to Gaussian elimination. Next, in Section 3 we describe our walk-based framework for inference, define walk-summable models, and explore the connections between walk-summable models and other subclasses of Gaussian models. We present the walk-sum interpretation of LBP and our conditions for its convergence in Section 4. We discuss non-walksummable models, and tighter conditions for LBP convergence in Section 5. Finally, conclusions and directions for further work are discussed in Section 6. Detailed proofs omitted from the main body of the paper appear in the appendices.

2. Preliminaries

In this section we give a brief background of Gaussian graphical models (Section 2.1) and of Gaussian elimination and its relation to belief propagation (Section 2.2).

2.1 Gaussian Graphical Models

A Gaussian graphical model is defined by an undirected graph $G = (V,E)$, where V is the set of nodes (or vertices) and E is the set of edges (a set of unordered pairs $\{i, j\} \subset V$), and a collection of jointly Gaussian random variables $x = (x_i, i \in V)$. The probability density is given by

$$p(x) \propto \exp\{-\tfrac{1}{2}x^T Jx + h^T x\} \tag{1}$$

where J is a symmetric, positive definite matrix ($J \succ 0$) that is sparse so as to respect the graph G: if $\{i, j\} \notin E$ then $J_{ij} = 0$. The condition $J \succ 0$ is necessary so that (1) defines a *valid* (i.e., normalizable) probability density. This is the *information form* of the Gaussian density. We call J the *information matrix* and h the *potential vector*. They are related to the standard Gaussian parameterization in terms of the mean $\mu \triangleq \mathbb{E}\{x\}$ and covariance $P \triangleq \mathbb{E}\{(x-\mu)(x-\mu)^T\}$ as follows:

$$\mu = J^{-1}h \quad \text{and} \quad P = J^{-1}.$$

This class of densities is precisely the family of non-degenerate Gaussian distributions which are Markov with respect to the graph G (Speed and Kiiveri, 1986): if a subset of nodes $B \subset V$ separates

3. Here, we choose one particular initialization of LBP. However, fixing this initialization does not restrict the class of models or applications for which our results apply. For instance, the application considered by Moallemi and Van Roy (2006a) can also be handled in our framework by a simple reparameterization.

two other subsets $A \subset V$ and $C \subset V$ in G, then the corresponding subsets of random variables x_A and x_C are conditionally independent given x_B. In particular, define the neighborhood of a node i to be the set of its neighbors: $\mathcal{N}(i) = \{j \mid \{i,j\} \in E\}$. Then, conditioned on $x_{\mathcal{N}(i)}$, the variable x_i is independent of the rest of the variables in the graph.

The *partial correlation coefficient* between variables x_i and x_j measures their conditional correlation given the values of the other variables $x_{V\setminus ij} \triangleq (x_k, k \in V \setminus \{i,j\})$. These are computed by normalizing the off-diagonal entries of the information matrix (Lauritzen, 1996):

$$r_{ij} \triangleq \frac{\mathrm{cov}(x_i; x_j | x_{V\setminus ij})}{\sqrt{\mathrm{var}(x_i|x_{V\setminus ij})\mathrm{var}(x_j|x_{V\setminus ij})}} = -\frac{J_{ij}}{\sqrt{J_{ii}J_{jj}}}. \tag{2}$$

Hence, we observe the relation between the sparsity of J and conditional independence between variables. In agreement with the Hammersley-Clifford theorem (Hammersley and Clifford, 1971), for Gaussian models we may factor the probability distribution

$$p(x) \propto \prod_{i\in V} \psi_i(x_i) \prod_{\{i,j\}\in E} \psi_{ij}(x_i, x_j)$$

in terms of node and edge *potential functions*:[4]

$$\psi_i(x_i) = \exp\{-\tfrac{1}{2}A_i x_i^2 + h_i x_i\} \quad \text{and} \quad \psi_{ij}(x_i,x_j) = \exp\{-\tfrac{1}{2}\left[\begin{smallmatrix} x_i & x_j \end{smallmatrix}\right] B_{ij} \left[\begin{smallmatrix} x_i \\ x_j \end{smallmatrix}\right]\}. \tag{3}$$

Here, A_i and B_{ij} must add up to J such that

$$x^T J x = \sum_i A_i x_i^2 + \sum_{\{i,j\}\in E} \left(\begin{smallmatrix} x_i & x_j \end{smallmatrix}\right) B_{ij} \left(\begin{smallmatrix} x_i \\ x_j \end{smallmatrix}\right).$$

The choice of a decomposition of J into such A_i and B_{ij} is not unique: the diagonal elements J_{ii} can be split in various ways between A_i and B_{ij}, but the off-diagonal elements of J are copied directly into the corresponding B_{ij}. It is *not* always possible to find a decomposition of J such that both $A_i > 0$ and $B_{ij} \succ 0$.[5] We call models where such a decomposition exists *pairwise-normalizable*.

Our analysis is not limited to pairwise-normalizable models. Instead we use the decomposition $A_i = J_{ii}$ and $B_{ij} = \left[\begin{smallmatrix} 0 & J_{ij} \\ J_{ij} & 0 \end{smallmatrix}\right]$, which always exists, and leads to the following node and edge potentials:

$$\psi_i(x_i) = \exp\{-\tfrac{1}{2}J_{ii}x_i^2 + h_i x_i\} \quad \text{and} \quad \psi_{ij}(x_i,x_j) = \exp\{-x_i J_{ij} x_j\}. \tag{4}$$

Note that *any* decomposition in (3) can easily be converted to our decomposition (4) using local operations (the required elements of J can be read off by adding overlapping matrices).

We illustrate this framework with a prototypical estimation problem. Suppose that we wish to estimate an unknown signal x (e.g., an image) based on noisy observations y. A commonly used prior model in image processing is the *thin membrane model* $p(x) \propto \exp(\{-\frac{1}{2}((\alpha\sum_i x_i^2 +$

4. To be precise, it is actually the negative logarithms of ψ_i and ψ_{ij} that are usually referred to as potentials in the statistical mechanics literature. We abuse the terminology slightly for convenience.

5. For example the model with $J = \left[\begin{smallmatrix} 1 & 0.6 & 0.6 \\ 0.6 & 1 & 0.6 \\ 0.6 & 0.6 & 1 \end{smallmatrix}\right]$ is a valid model with $J \succ 0$, but no decomposition into single and pairwise positive definite factors exists. This can be verified by posing an appropriate semidefinite feasibility problem, or as we discuss later through walk-summability.

$\beta \sum_{\{i,j\}\in E}(x_i - x_j)^2))))\}$ where $\alpha, \beta > 0$ and E specifies nearest neighbors in the image. This model is described by a sparse information matrix with $J_{ii} = \alpha + \beta|\mathcal{N}(i)|$ and $J_{ij} = -\beta$ for $\{i, j\} \in E$.

Now, consider local observations y, such that $p(y|x) = \prod_i p(y_i|x_i)$. The distribution of interest is then $p(x|y) \propto p(y|x)p(x)$, which is Markov with respect to the same graph as $p(x)$, but with modified information parameters. For instance, let $y = x + v$ where v is Gaussian distributed measurement noise with zero mean and covariance $\sigma^2 I$. Then $p(x|y) \propto \exp\{-\frac{1}{2}x^T \hat{J} x + h^T x\}$, where $\hat{J} = J + \frac{1}{\sigma^2} I$ and $h = \frac{1}{\sigma^2} y$. Hence, introducing local observations only changes the potential vector h and the diagonal of the information matrix J. Without loss of generality, in subsequent discussion we assume that any observations have already been absorbed into J and h.

2.2 Belief Propagation and Gaussian Elimination

An important inference problem for a graphical model is computing the marginals $p_i(x_i)$, obtained by integrating $p(x)$ over all variables except x_i, for each node i.[6] This problem can be solved very efficiently in graphs that are trees by a form of variable elimination, known as belief propagation, which also provides an approximate method for general graphs.

Belief Propagation in Trees In principle, the marginal of a given node can be computed by recursively eliminating variables one by one until just the desired node remains. Belief propagation in trees can be interpreted as an efficient form of variable elimination. Rather than computing the marginal for each variable independently, we instead compute these together by sharing the results of intermediate computations. Ultimately each node j must receive information from each of its neighbors, where the *message*, $m_{i\to j}(x_j)$, from neighbor i to j represents the result of eliminating all of the variables in the subtree rooted at node i and including all of its neighbors other than j (see Figure 1). Since each of these messages is itself made up of variable elimination steps corresponding to the subtrees rooted at the other neighbors of node i, there is a set of fixed-point equations that relate messages throughout the tree:

$$m_{i\to j}(x_j) = \int \psi_{ij}(x_i, x_j)\psi_i(x_i) \prod_{k\in\mathcal{N}(i)\setminus j} m_{k\to i}(x_i)\ dx_i. \tag{5}$$

Given these fixed-point messages, the marginals are obtained by combining messages at each node,

$$p_i(x_i) \propto \psi_i(x_i) \prod_{k\in\mathcal{N}(i)} m_{k\to i}(x_i),$$

and normalizing the result.

The equations (5) can be solved in a finite number of steps using a variety of *message schedules*, including one schedule that corresponds roughly to sequential variable elimination and back-substitution (a first pass from leaf nodes toward a common, overall "root" node followed by a reverse pass back to the leaf nodes) and a fully parallel schedule in which each node begins by sending non-informative messages (all $m_{i\to j}$ initially set to 1), followed by iterative computation of (5) throughout the tree. For trees, either message schedule will terminate with the correct values after a finite number of steps (equal to the diameter of the tree in the case of the fully parallel iteration).

6. Another important problem is computation of max-marginals $\hat{p}_i(x_i)$, obtaining by *maximizing* with respect to the other variables, which is useful to determine the mode $\hat{x} = \arg\max p(x)$. In Gaussian models, these are equivalent inference problems because marginals are proportional to max-marginals and the mean is equal to the mode.

As we have discussed, there are a variety of ways in which the information matrix in GMRFs can be decomposed into edge and node potential functions, and each such decomposition leads to BP iterations that are different in detail.[7] In our development we will use the simple decomposition in (4), directly in terms of the elements of J.

For Gaussian models expressed in information form, variable elimination/marginalization corresponds to *Gaussian elimination*.[8] For example, if we wish to eliminate a single variable i to obtain the marginal over $U = V \backslash i$, the formulas yielding the information parameterization for the marginal on U are:

$$\hat{J}_U = J_{U,U} - J_{U,i} J_{ii}^{-1} J_{i,U} \quad \text{and} \quad \hat{h}_U = h_U - J_{U,i} J_{ii}^{-1} h_i.$$

Here $\hat{J}_U$ and $\hat{h}_U$ specify the marginal density on x_U, whereas $J_{U,U}$ and h_U are a submatrix and a subvector of the information parameters on the full graph. The messages in Gaussian models can be parameterized in information form

$$m_{i\to j}(x_j) \triangleq \exp\{-\tfrac{1}{2}\Delta J_{i\to j} x_j^2 + \Delta h_{i\to j} x_j\}, \tag{6}$$

so that the fixed-point equations (5) can be stated in terms of these information parameters. We do this in two steps. The first step corresponds to preparing the message to be sent from node i to node j by collecting information from all of the other neighbors of i:

$$\hat{J}_{i\backslash j} = J_{ii} + \sum_{k\in\mathcal{N}(i)\backslash j} \Delta J_{k\to i} \quad \text{and} \quad \hat{h}_{i\backslash j} = h_i + \sum_{k\in\mathcal{N}(i)\backslash j} \Delta h_{k\to i}. \tag{7}$$

The second step produces the information quantities to be propagated to node j:

$$\Delta J_{i\to j} = -J_{ji} \hat{J}_{i\backslash j}^{-1} J_{ji} \quad \text{and} \quad \Delta h_{i\to j} = -J_{ji} \hat{J}_{i\backslash j}^{-1} \hat{h}_{i\backslash j}. \tag{8}$$

As before, these equations can be solved by various message schedules, ranging from leaf-root-leaf Gaussian elimination and back-substitution to fully parallel iteration starting from the non-informative messages in which all $\Delta J_{i\to j}$ and $\Delta h_{i\to j}$ are set to zero. When the fixed point solution is obtained, the computation of the marginal at each node is obtained by combining messages and local information:

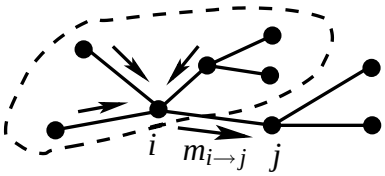
$$\hat{J}_i = J_{ii} + \sum_{k\in\mathcal{N}(i)} \Delta J_{k\to i} \quad \text{and} \quad \hat{h}_i = h_i + \sum_{k\in\mathcal{N}(i)} \Delta h_{k\to i}, \tag{9}$$

which can be easily inverted to recover the marginal mean and variance:

$$\mu_i = \hat{J}_i^{-1} \hat{h}_i \quad \text{and} \quad P_{ii} = \hat{J}_i^{-1}.$$

In general, performing Gaussian elimination corresponds, upto a permutation, to computing an LDL^T factorization of the information matrix—that is, $PJP^T = LDL^T$ where L is lower-triangular, D is diagonal and P is a permutation matrix corresponding to a particular choice of elimination order. This factorization exists if J is non-singular. In trees, the elimination order can be chosen such that at each step of the procedure, the next node eliminated is a leaf node of the remaining subtree. Each node elimination step then corresponds to a message in the "upward" pass of the leaf-root-leaf form

7. One common decomposition for pairwise-normalizable models selects $A_i > 0$ and $B_{ij} \succ 0$ in (3) (Plarre and Kumar, 2004; Weiss and Freeman, 2001; Moallemi and Van Roy, 2006a).
8. The connection between Gaussian elimination and belief propagation has been noted before by Plarre and Kumar (2004), although they do not use the information form.



A message $m_{i\to j}$ passed from node i to node $j \in \mathcal{N}(i)$ captures the effect of eliminating the subtree rooted at i.

Figure 1: An illustration of BP message-passing on trees.

of Gaussian BP. In particular, $D_{ii} = \hat{J}_{i\backslash j}$ at all nodes i except the last (here, j is the parent of node i when i is eliminated) and $D_{ii} = \hat{J}_i$ for that last variable corresponding to the root of the tree. It is clear that $D_{ii} > 0$ for all i if and only if J is positive definite. We conclude that for models on trees, J being positive definite is equivalent to all of the quantities $\hat{J}_{i\backslash j}$ and $\hat{J}_i$ in (7),(9) being positive, a condition we indicate by saying that BP on this tree is *well-posed*. Thus, performing Gaussian BP on trees serves as a simple test for validity of the model. The importance of this notion will become apparent shortly.

Loopy Belief Propagation The message passing formulas derived for tree models can also be applied to models defined on graphs with cycles, even though this no longer corresponds precisely to variable elimination in the graph. This approximation method, called *loopy belief propagation* (LBP), was first proposed by Pearl (1988). Of course in this case, since there are cycles in the graph, only iterative message-scheduling forms can be defined. To be precise, a message schedule $\{\mathcal{M}^{(n)}\}$ specifies which messages $m_{i\to j}^{(n)}$, corresponding to directed edges $(i,j) \in \mathcal{M}^{(n)}$,[9] are updated at step n. The messages in $\mathcal{M}^{(n)}$ are updated using

$$m_{i\to j}^{(n)}(x_j) = \int \psi_{ij}(x_i,x_j)\psi_i(x_i) \prod_{k\in\mathcal{N}(i)\backslash j} m_{k\to i}^{(n-1)}(x_i)\ dx_i \tag{10}$$

and $m_{i\to j}^{(n)} = m_{i\to j}^{(n-1)}$ for the other messages. For example, in the fully parallel case *all* messages are updated at each iteration whereas, in serial versions, only one message is updated at each iteration. For GMRFs, application of (10), with messages $m_{i\to j}^{(n)}$ parameterized as in (6), reduces to iterative application of equations (7),(8). We denote the information parameters at step n by $\Delta J_{i\to j}^{(n)}$ and $\Delta h_{i\to j}^{(n)}$. We initialize LBP with non-informative zero values for all of the information parameters in these messages. It is well known that LBP may or may not converge. If it does converge, it will not, in general, yield the correct values for the marginal distributions. In the Gaussian case, however, it is known (Weiss and Freeman, 2001; Rusmevichientong and Van Roy, 2001) that if LBP converges, it yields the correct mean values but, in general, incorrect values for the variances. While there has been considerable work on analyzing the convergence of LBP in general and for GMRFs in particular, the story has been far from complete. One major contribution of this paper is analysis that both provides new insights into LBP for Gaussian models and also brings that story several steps closer to completion.

A key component of our analysis is the insightful interpretation of LBP in terms of the so-called *computation tree* (Yedidia et al., 2003; Weiss and Freeman, 2001; Tatikonda and Jordan, 2002), which captures the structure of LBP computations. The basic idea here is that to each message $m_{i\to j}^{(n)}$

9. For each undirected edge $\{i,j\} \in E$ there are two messages: $m_{i\to j}$ for direction (i,j), and $m_{j\to i}$ for (j,i).

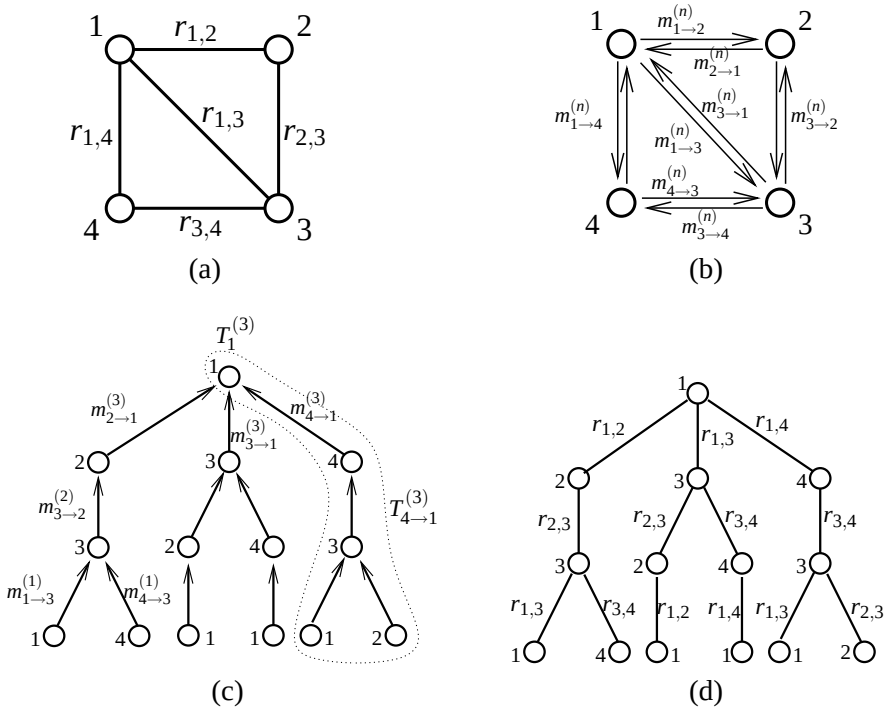


Figure 2: (a) Graph of a Gauss-Markov model with nodes $\{1,2,3,4\}$ and with edge weights (partial correlations) as shown. (b) The parallel LBP message passing scheme. In (c), we show how, after 3 iterations, messages link up to form the computation tree $T_1^{(3)}$ of node 1 (the subtree $T_{4\to 1}^{(3)}$, associated with message $m_{4\to 1}^{(3)}$, is also indicated within the dotted outline). In (d), we illustrate an equivalent Gauss-Markov tree model, with edge weights copied from (a), which has the same marginal at the root node as computed by LBP after 3 iterations.

and marginal estimate $p_i^{(n)}$ there are associated computation trees $T_{i\to j}^{(n)}$ and $T_i^{(n)}$ that summarize their pedigree. Initially, these trees are just single nodes. When message $m_{i\to j}^{(n)}$ is computed, its computation tree $T_{i\to j}^{(n)}$ is constructed by joining the trees $T_{k\to i}^{(n-1)}$, for all neighbors k of i except j, at their common root node i and then adding an additional edge (i,j) to form $T_{i\to j}^{(n)}$ rooted at j. When marginal estimate $p_i^{(n)}$ is computed, its computation tree $T_i^{(n)}$ is formed by joining the trees $T_{k\to i}^{(n-1)}$, for all neighbors k of i, at their common root. Each node and edge of the original graph may be replicated many times in the computation tree, but in a manner which preserves the local neighborhood structure. Potential functions are assigned to the nodes and edges of $T_i^{(n)}$ by copying these from the corresponding nodes and edges of the original loopy graphical model. In this manner, we obtain a Markov tree model in which the marginal at the root node is precisely $p_i^{(n)}$ as computed by LBP. In the case of the fully parallel form of LBP, this leads to a collection of "balanced" computation trees $T_i^{(n)}$ (assuming there are no leaf nodes in G) having uniform depth n, as illustrated in Figure 2. The same construction applies for other message schedules with the only difference being that the resulting computation trees may grow in a non-uniform manner. Our walk-

sum analysis of LBP in Section 6, which relies on computation trees, applies for general message passing schedules.

As we have mentioned, BP on trees, which corresponds to performing Gaussian elimination, is well-posed if and only if J is positive definite. LBP on Gaussian models corresponds to Gaussian elimination in the computation tree, which has its own information matrix corresponding to the unfolding illustrated in Figure 2 and involving replication of information parameters of the original loopy graphical model. Consequently, LBP is well-posed, yielding non-negative variances at each stage of the iteration, if and only if the model on the computation tree is *valid*, that is, if and only if the information matrix for the computation tree is positive definite. Very importantly, this is *not* always the case (even though the matrix J on the original graph is positive definite). The analysis in this paper, among other things, makes this point clear through analysis of the situations in which LBP converges and when it fails to converge.

3. Walk-Summable Gaussian Models

Now we describe our walk-sum framework for Gaussian inference. It is convenient to assume that we have normalized our model (by rescaling variables) so that $J_{ii} = 1$ for all i. Then, $J = I - R$ where R has zero diagonal and the off-diagonal elements are equal to the partial correlation coefficients r_{ij} in (2). We label each edge $\{i, j\}$ of the graph G with partial correlations r_{ij} as edge weights (e.g., see Figures 3 and 5).

3.1 Walk-Summability

A *walk* of length $l \geq 0$ in a graph G is a sequence $w = (w_0, w_1, \ldots, w_l)$ of nodes $w_k \in V$ such that each step of the walk (w_k, w_{k+1}) corresponds to an edge of the graph $\{w_k, w_{k+1}\} \in E$. Walks may visit nodes and cross edges multiple times. We let $l(w)$ denote the length of walk w. We define the *weight* of a walk to be the product of edge weights along the walk:

$$\phi(w) = \prod_{k=1}^{l(w)} r_{w_{k-1},w_k} \ .$$

We also allow zero-length "self" walks $w = (v)$ at each node v for which we define $\phi(w) = 1$. To make a connection between these walks and Gaussian inference, we decompose the covariance matrix using the Neumann power series for the matrix inverse:[10]

$$P = J^{-1} = (I - R)^{-1} = \sum_{k=0}^{\infty} R^k, \quad \text{for} \quad \rho(R) < 1.$$

Here $\rho(R)$ is the spectral radius of R, the maximum absolute value of eigenvalues of R. The power series converges if $\rho(R) < 1$.[11] The (i, j)-th element of R^l can be expressed as a sum of weights of walks w that go from i to j and have length l (denoted $w : i \xrightarrow{l} j$):

$$(R^l)_{ij} = \sum_{w_1,\ldots,w_{l-1}} r_{i,w_1} r_{w_1,w_2} \ldots r_{w_{l-1},j} = \sum_{w : i \xrightarrow{l} j} \phi(w).$$

10. The Neumann series holds for the unnormalized case as well: $J = D - K$, where D is the diagonal part of J. With the weight of a walk defined as $\phi(w) = \prod_{k=1}^{l(w)} K_{w_{k-1},w_k} / \prod_{k=0}^{l(w)} D_{w_k,w_k}$, all our analysis extends to the unnormalized case.
11. Note that $\rho(R)$ can be greater than 1 while $I - R \succ 0$. This occurs if R has an eigenvalue less than -1. Such models are not walk-summable, so the analysis in Section 5 (rather than Section 4.2) applies.

The last equality holds because only the terms that correspond to walks in the graph have non-zero contributions: for all other terms at least one of the partial correlation coefficients $r_{w_k,w_{k+1}}$ is zero. The set of walks from i to j of length l is finite, and the sum of weights of these walks (the walk-sum) is well-defined. We would like to also define walk-sums over arbitrary countable sets of walks. However, care must be taken, as walk-sums over countably many walks may or may not converge, and convergence may depend on the order of summation. This motivates the following definition:

We say that a Gaussian distribution is *walk-summable* (WS) if for all $i, j \in V$ the unordered sum over all walks w from i to j (denoted $w : i \to j$)

$$\sum_{w:i\to j} \phi(w)$$

is well-defined (i.e., converges to the same value for every possible summation order). Appealing to basic results of analysis (Rudin, 1976; Godement, 2004), the unordered sum is well-defined if and only if it *converges absolutely*, that is, if $\sum_{w:i\to j} |\phi(w)|$ converges.

Before we take a closer look at walk-summability, we introduce additional notation. For a matrix A, let $\bar{A}$ be the element-wise absolute value of A, that is, $\bar{A}_{ij} = |A_{ij}|$. We use the notation $A \geq B$ for element-wise comparisons, and $A \succeq B$ for comparisons in positive definite ordering. The following version of the Perron-Frobenius theorem (Horn and Johnson, 1985; Varga, 2000) for non-negative matrices (here $\bar{R} \geq 0$) is used on several occasions in the paper:

Perron-Frobenius theorem There exists a non-negative eigenvector $x \geq 0$ of $\bar{R}$ with eigenvalue $\rho(\bar{R})$. If the graph G is connected (where $r_{ij} \neq 0$ for all edges of G) then $\rho(\bar{R})$ and x are strictly positive and, apart from γx with $\gamma > 0$, there are no other non-negative eigenvectors of $\bar{R}$.

In addition, we often use the following monotonicity properties of the spectral radius:

$$\text{(i)}\ \ \rho(R) \leq \rho(\bar{R}) \quad \text{(ii) If } \bar{R}_1 \leq \bar{R}_2 \ \text{ then } \ \rho(\bar{R}_1) \leq \rho(\bar{R}_2). \tag{11}$$

We now present several equivalent conditions for walk-summability:

Proposition 1 (Walk-Summability) *Each of the following conditions are equivalent to walk-summability:*

(i) $\sum_{w:i\to j} |\phi(w)|$ *converges for all* $i, j \in V$.

(ii) $\sum_l \bar{R}^l$ *converges.*

(iii) $\rho(\bar{R}) < 1$.

(iv) $I - \bar{R} \succ 0$.

The proof appears in Appendix A. It uses absolute convergence to rearrange walks in order of increasing length, and the Perron-Frobenius theorem for part (iv). The condition $\rho(\bar{R}) < 1$ is stronger than $\rho(R) < 1$. The latter is sufficient for the convergence of the walks ordered by increasing length, whereas walk-summability enables convergence to the same answer in arbitrary order of summation. Note that (iv) implies that the model is walk-summable if and only if we can replace all negative partial correlation coefficients by their absolute values and still have a well-defined model (i.e., with information matrix $I - \bar{R} \succ 0$). We also note that condition (iv) relates walk-summability to the

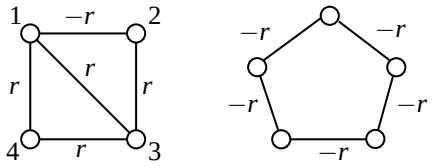


Figure 3: Example graphs: (a) 4-cycle with a chord. (b) 5-cycle.

so-called H-matrices in linear algebra (Horn and Johnson, 1991; Varga, 2000).[12] As an immediate corollary, we identify the following important subclass of walk-summable models:

Corollary 2 (Attractive Models) *Let $J = I - R$ be a valid model ($J \succ 0$) with non-negative partial correlations $R \geq 0$. Then, $J = I - R$ is walk-summable.*

A superclass of attractive models is the set of *non-frustrated models*. A model is non-frustrated if it does not contain any frustrated cycles, that is, cycles with an odd number of negative edge weights. We show in Appendix A (in the proof of Corollary 3) that if the model is non-frustrated, then one can negate some of the variables to make the model attractive[13]. Hence, we have another subclass of walk-summable models (the inclusion is strict as some frustrated models are walk-summable, see Example 1):

Corollary 3 (Non-frustrated models) *Let $J = I - R$ be valid. If R is non-frustrated then J is walk-summable.*

Example 1. In Figure 3 we illustrate two small Gaussian graphical models, which we use throughout the paper. In both models the information matrix J is normalized to have unit diagonal and to have partial correlations as indicated in the figure. Consider the 4-cycle with a chord in Figure 3(a). The model is frustrated (due to the opposing sign of one of the partial correlations), and increasing r worsens the frustration. For $0 \leq r \leq 0.39039$, the model is valid and walk-summable: for example, for $r = 0.39$, $\lambda_{\min}(J) = 0.22 > 0$, and $\rho(\bar{R}) \approx 0.9990 < 1$. In the interval $0.39039 \leq r \leq 0.5$ the model is valid, but not walk-summable: for example, for $r = 0.4$, $\lambda_{\min} = 0.2 > 0$, and $\rho(\bar{R}) \approx 1.0246 > 1$. Also, note that for R (as opposed to $\bar{R}$), $\rho(R) \leq 1$ for $r \leq 0.5$ and $\rho(R) > 1$ for $r > 0.5$. Finally, the model stops being diagonally dominant above $r = \frac{1}{3}$, but walk-summability is a strictly larger set and extends until $r \approx 0.39039$. We summarize various critical points for this model and for the model in Figure 3(b) in the diagram in Figure 4.

Here are additional useful implications of walk-summability, with proof in Appendix A:

Proposition 4 (WS Necessary Conditions) *All of the following are implied by walk-summability:*

12. A (possibly non-symmetric) matrix A is an *H-matrix* if all eigenvalues of the matrix $M(A)$, where $M_{ii} = |A_{ii}|$, and $M_{ij} = -|A_{ij}|$ for $i \neq j$, have positive real parts. For symmetric matrices this is equivalent to M being positive definite. In (iv) J is an H-matrix since $M(J) = I - \bar{R} \succ 0$.
13. This result is referred to in Kirkland et al. (1996). However, in addition to proving that there exists such a sign similarity, our proof also gives an algorithm which checks whether or not the model is frustrated, and determines which subset of variables to negate if the model is non-frustrated.

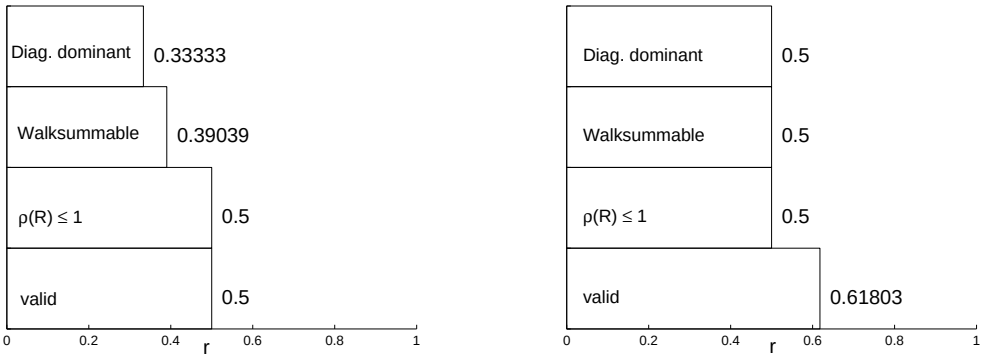


Figure 4: Critical regions for example models from Figure 3. (a) 4-cycle with a chord. (b) 5-cycle.

(i) $\rho(R) < 1$.

(ii) $J = I - R \succ 0$.

(iii) $\sum_k R^k = (I-R)^{-1}$.

Implication (ii) shows that walk-summability is a sufficient condition for validity of the model. Also, (iii) shows the relevance of walk-sums for inference since $P = J^{-1} = (I-R)^{-1} = \sum_k R^k$ and $\mu = J^{-1}h = \sum_k R^k h$.

3.2 Walk-Sums for Inference

Next we show that, in walk-summable models, means and variances correspond to walk-sums over certain sets of walks.

Proposition 5 (WS Inference) *If $J = I - R$ is walk-summable, then the covariance $P = J^{-1}$ is given by the walk-sums:*

$$P_{ij} = \sum_{w: i \to j} \phi(w).$$

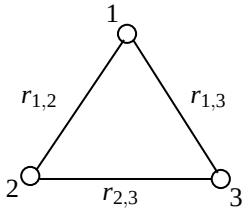
Also, the means are walk-sums reweighted by the value of h at the start of each walk:

$$\mu_i = \sum_{w: * \to i} h_* \phi(w)$$

*where the sum is over **all** walks which end at node i (with arbitrary starting node), and where $*$ denotes the starting node of the walk w.*

Proof. We use the fact that $(R^l)_{ij} = \sum_{w: i \overset{l}{\to} j} \phi(w)$. Then,

$$P_{ij} = \sum_l (R^l)_{ij} = \sum_l \sum_{w: i \overset{l}{\to} j} \phi(w) = \sum_{w: i \to j} \phi(w)$$



Single walk: $w = (1,2,3)$. Weight: $\phi(w) = r_{1,2}r_{2,3}$,
$\phi_h(w) = h_1 r_{1,2}r_{2,3}$.

Self-return walks, $\mathcal{W}(1 \to 1)$: $\{(1),(1,2,1),(1,3,1),$
$(1,2,3,1),(1,3,2,1),(1,2,1,2,1),...\}$
$P_{1,1} = \phi(1 \to 1) = 1 + r_{1,2}r_{2,1} + r_{1,3}r_{3,1} + r_{1,2}r_{2,3}r_{3,1} + ...$

Set of walks $\mathcal{W}(* \to 1)$: $\{(1),(2,1),(3,1),(2,3,1),$
$(3,2,1),(1,2,1)(1,3,1),...\}$
$\mu_1 = \phi_h(* \to 1) = h_1 + h_2 r_{2,1} + h_3 r_{3,1} + h_2 r_{2,3}r_{3,1} + ...$

Figure 5: Illustration of walk-sums for means and variances.

and

$$\mu_i = \sum_j h_j P_{ji} = \sum_j \sum_{w: j \xrightarrow{l} i} h_j \phi(w) = \sum_{w: * \to i} h_* \phi(w) \ \square$$

Walk-Sum Notation We now provide a more compact notation for walk-sets and walk-sums. In general, given a set of walks $\mathcal{W}$ we define the walk-sum:

$$\phi(\mathcal{W}) = \sum_{w \in \mathcal{W}} \phi(w)$$

and the reweighted walk-sum:

$$\phi_h(\mathcal{W}) = \sum_{w \in \mathcal{W}} h_{w_0} \phi(w)$$

where w_0 denotes the initial node in the walk w. Also, we adopt the convention that $\mathcal{W}(\dots)$ denotes the set of all walks having some property $\dots$ and denote the associated walk-sums simply as $\phi(\dots)$ or $\phi_h(\dots)$. For instance, $\mathcal{W}(i \to j)$ denotes the set of all walks from i to j and $\phi(i \to j)$ is the corresponding walk-sum. Also, $\mathcal{W}(* \to i)$ denotes the set all walks that end at node i and $\phi_h(* \to i)$ is the corresponding reweighted walk-sum. In this notation, $P_{ij} = \phi(i \to j)$ and $\mu_i = \phi_h(* \to i)$. An illustration of walk-sums and their connection to inference appears in Figure 5 where we list some walks and walk-sums for a 3-cycle graph.

Walk-Sum Algebra We now show that the walk-sums required for inference in walk-summable models can be significantly simplified by exploiting the recursive structure of walks. To do so, we make use of some simple algebraic properties of walk-sums. The following lemmas all assume that the model is walk-summable.

Lemma 6 *Let $\mathcal{W} = \cup_{k=1}^{\infty} \mathcal{W}_k$ where the subsets $\mathcal{W}_k$ are disjoint. Then, $\phi(\mathcal{W}) = \sum_{k=1}^{\infty} \phi(\mathcal{W}_k)$.*

Proof. By the sum-partition theorem for absolutely convergent series (Godement, 2004): $\sum_{w \in \mathcal{W}} \phi(w) = \sum_k \sum_{w \in \mathcal{W}_k} \phi(w)$. $\square$

Lemma 7 *Let $\mathcal{W} = \cup_{k=1}^{\infty} \mathcal{W}_k$ where $\mathcal{W}_k \subset \mathcal{W}_{k+1}$ for all k. Then, $\phi(\mathcal{W}) = \lim_{k \to \infty} \phi(\mathcal{W}_k)$.*

Proof. Let $\mathcal{W}_0$ be the empty set. Then, $\mathcal{W} = \cup_{k=1}^{\infty}(\mathcal{W}_k \setminus \mathcal{W}_{k-1})$. By Lemma 6,

$$\phi(\mathcal{W}) = \sum_{k=1}^{\infty} \phi(\mathcal{W}_k \setminus \mathcal{W}_{k-1}) = \lim_{N\to\infty} \sum_{k=1}^{N} (\phi(\mathcal{W}_k) - \phi(\mathcal{W}_{k-1})) = \lim_{N\to\infty} (\phi(\mathcal{W}_N) - \phi(\mathcal{W}_0))$$

where we use $\phi(\mathcal{W}_0) = 0$ in the last step to obtain the result. □

Given two walks $u = (u_0, \ldots, u_n)$ and $v = (v_0, \ldots, v_m)$ with $u_n = v_0$ (walk v begins where walk u ends) we define the product of walks $uv = (u_0, \ldots, u_n, v_1, \ldots, v_m)$. Let $\mathcal{U}$ and $\mathcal{V}$ be two countable sets of walks such that every walk in $\mathcal{U}$ ends at a given node i and every walk in $\mathcal{V}$ begin at this node. Then we define the product set $\mathcal{UV} = \{uv \mid u \in \mathcal{U}, v \in \mathcal{V}\}$. We say that $(\mathcal{U}, \mathcal{V})$ is a *valid decomposition* if for every $w \in \mathcal{UV}$ there is a unique pair $(u, v) \in \mathcal{U} \times \mathcal{V}$ such that $uv = w$.

Lemma 8 *Let $(\mathcal{U}, \mathcal{V})$ be a valid decomposition. Then, $\phi(\mathcal{UV}) = \phi(\mathcal{U})\phi(\mathcal{V})$.*

Proof. For individual walks it is evident that $\phi(uv) = \phi(u)\phi(v)$. Note that $\mathcal{UV} = \cup_{u\in\mathcal{U}} u\mathcal{V}$, where the sets $u\mathcal{V} \triangleq \{uv | v \in \mathcal{V}\}$ are mutually disjoint. By Lemma 6,

$$\phi(\mathcal{UV}) = \sum_{u\in\mathcal{U}} \phi(u\mathcal{V}) = \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \phi(uv) = \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{V}} \phi(u)\phi(v) = \left(\sum_{u\in\mathcal{U}} \phi(u)\right)\left(\sum_{v\in\mathcal{V}} \phi(v)\right)$$

where we have used $\phi(u\mathcal{V}) = \sum_{v\in\mathcal{V}} \phi(uv)$ because $u\mathcal{V}$ is one-to-one with $\mathcal{V}$. □

Note that $\mathcal{W}(i \to i)$ is the set of *self-return walks* at node i, that is, walks which begin and end at node i. These self-return walks include walks which return to i many times. Let $\mathcal{W}(i \xrightarrow{\setminus i} i)$ be the set of all walks with non-zero length that begin and end at i but do not visit i at any other point in between. We call these the *single-revisit* self-return walks at node i. The set of self-return walks that return exactly k times is generated by taking the product of k copies of $\mathcal{W}(i \xrightarrow{\setminus i} i)$ denoted by $\mathcal{W}^k(i \xrightarrow{\setminus i} i)$. Thus, we obtain all self-return walks as

$$\mathcal{W}(i \to i) = \cup_{k\geq 0} \mathcal{W}^k(i \xrightarrow{\setminus i} i) \tag{12}$$

where $\mathcal{W}^0(i \xrightarrow{\setminus i} i) \triangleq \{(i)\}$.

Similarly, recall that $\mathcal{W}(* \to i)$ denotes the set of all walks which end at node i. Let $\mathcal{W}(* \xrightarrow{\setminus i} i)$ denote the set of walks with non-zero length which end at node i and do not visit i previously (we call them *single-visit walks*). Thus, all walks which end at i are obtained as:

$$\mathcal{W}(* \to i) = \left(\{(i)\} \cup \mathcal{W}(* \xrightarrow{\setminus i} i)\right) \mathcal{W}(i \to i), \tag{13}$$

which is a valid decomposition.

Now we can decompose means and variances in terms of single-visit and single-revisit walk-sums, which we will use in section 4.1 to analyze BP.

Proposition 9 *Let $\alpha_i = \phi(i \xrightarrow{\setminus i} i)$ and $\beta_i = \phi_h(* \xrightarrow{\setminus i} i)$. Then,*

$$P_{ii} = \frac{1}{1-\alpha_i} \quad \textit{and} \quad \mu_i = \frac{h_i + \beta_i}{1-\alpha_i}.$$

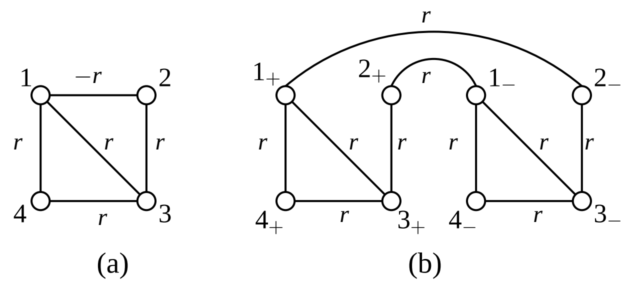


Figure 6: (a) A frustrated model defined on G with one negative edge ($r > 0$). (b) The corresponding attractive model defined on $\hat{G}$.

Proof. First note that the decomposition of $\mathcal{W}^k(i \xrightarrow{\backslash i} i)$ into products of k single-revisit self-return walks is a valid decomposition. By Lemma 8, $\phi(\mathcal{W}^k(i \xrightarrow{\backslash i} i)) = \phi^k(i \xrightarrow{\backslash i} i) = \alpha_i^k$. Then, by (12) and Lemma 6:

$$P_{ii} = \phi(i \to i) = \sum_k \alpha_i^k = \frac{1}{1-\alpha_i}.$$

Walk-summability of the model implies convergence of the geometric series (i.e., $|\alpha_i| < 1$). Lastly, the decomposition in (13) implies

$$\mu_i = \phi_h(* \to i) = (h_i + \phi_h(* \xrightarrow{\backslash i} i))\phi(i \to i) = \frac{h_i + \beta_i}{1-\alpha_i} \quad \square$$

3.3 Correspondence to Attractive Models

We have already shown that attractive models are walk-summable. Interestingly, it turns out that inference in any walk-summable model can be reduced to inference in a corresponding attractive model defined on a graph with twice as many nodes. The basic idea here is to separate out the walks with positive and negative weights.

Specifically, let $\hat{G} = (\hat{V}, \hat{E})$ be defined as follows. For each node $i \in V$ we define two corresponding nodes $i_+ \in V_+$ and $i_- \in V_-$, and set $\hat{V} = V_+ \cup V_-$. For each edge $\{i, j\} \in E$ with $r_{ij} > 0$ we define two edges $\{i_+, j_+\}, \{i_-, j_-\} \in \hat{E}$, and set the partial correlations on these edges to be equal to r_{ij}. For each edge $\{i, j\} \in E$ with $r_{ij} < 0$ we define two edges $\{i_+, j_-\}, \{i_-, j_+\} \in \hat{G}$, and set the partial correlations to be $-r_{ij}$. See Figure 6 for an illustration.

Let $(R_+)_{ij} = \max\{R_{ij}, 0\}$ and $(R_-)_{ij} = \max\{-R_{ij}, 0\}$. Then R can be expressed as the difference of these non-negative matrices: $R = R_+ - R_-$. Based on our construction, we have that $\hat{R} = \begin{pmatrix} R_+ & R_- \\ R_- & R_+ \end{pmatrix}$ and $\hat{J} = I - \hat{R}$. This defines a unit-diagonal information matrix $\hat{J}$ on $\hat{G}$. Note that if $\hat{J} \succ 0$ then this defines a valid attractive model.

Proposition 10 *$\hat{J} = I - \hat{R} \succ 0$ if and only if $J = I - R$ is walk-summable.*

The proof relies on the Perron-Frobenius theorem and is given in Appendix A. Now, let $h = h_+ - h_-$ with $(h_+)_i = \max\{h_i, 0\}$ and $(h_-)_i = \max\{-h_i, 0\}$. Define $\hat{h} = \begin{pmatrix} h_+ \\ h_- \end{pmatrix}$. Now we have the information form model $(\hat{h}, \hat{J})$ which is a valid, attractive model and also has non-negative node

potentials. Performing inference with respect to this augmented model, we obtain the mean vector $\hat{\mu} = \begin{pmatrix} \hat{\mu}_+ \\ \hat{\mu}_- \end{pmatrix} \triangleq \hat{J}^{-1}\hat{h}$ and covariance matrix $\hat{P} = \begin{pmatrix} \hat{P}_{++} & \hat{P}_{+-} \\ \hat{P}_{-+} & \hat{P}_{--} \end{pmatrix} \triangleq \hat{J}^{-1}$. From these calculations, we can obtain the moments (μ, P) of the original walk-summable model (h, J):

Proposition 11 *$P = \hat{P}_{++} - \hat{P}_{+-}$ and $\mu = \hat{\mu}_+ - \hat{\mu}_-$.*

The proof appears in Appendix A. This proposition shows that estimation of walk-summable models may be reduced to inference in an attractive model in which all walk-sums are sums of positive weights. In essence, this is accomplished by summing walks with positive and negative weights separately and then taking the difference, which is only possible for walk-summable models.

3.4 Pairwise-Normalizability

To simplify presentation we assume that the graph does not contain any isolated nodes (a node without any incident edges). Then, we say that the information matrix J is *pairwise-normalizable* (PN) if we can represent J in the form

$$J = \sum_{e \in E} [J_e]$$

where each J_e is a 2×2 symmetric, positive definite matrix.[14] The notation $[J_e]$ means that J_e is zero-padded to a $|V| \times |V|$ matrix with its principal submatrix for $\{i, j\}$ being J_e (with $e = \{i, j\}$). Thus, $x^T[J_e]x = x_e^T J_e x_e$. Pairwise-normalizability implies that $J \succ 0$ because each node is covered by at least one positive definite submatrix J_e. Let $\mathcal{J}_{PN}$ denote the set of $n \times n$ pairwise-normalizable information matrices J (not requiring unit-diagonal normalization). This set has nice convexity properties. Recall that a set $\mathcal{X}$ is *convex* if $x, y \in \mathcal{X}$ implies $\lambda x + (1-\lambda)y \in \mathcal{X}$ for all $0 \leq \lambda \leq 1$ and is a *cone* if $x \in \mathcal{X}$ implies $\alpha x \in \mathcal{X}$ for all $\alpha > 0$. A cone $\mathcal{X}$ is *pointed* if $\mathcal{X} \cap -\mathcal{X} = \{0\}$.

Proposition 12 (Convexity of PN models) *The set $\mathcal{J}_{PN}$ is a convex pointed cone.*

The proof is in Appendix A. We now establish the following fundamental result:

Proposition 13 (WS $\Leftrightarrow$ PN) *$J = I - R$ is walk-summable if and only if it is pairwise-normalizable.*

Our proof appears in in Appendix A. An equivalent result has been derived independently in the linear algebra literature: Boman et al. (2005) establish that symmetric H-matrices with positive diagonals (which is equivalent to WS by part (iv) of Proposition 1) are equivalent to matrices with factor width at most two (PN models). However, the result $PN \Rightarrow WS$ was established earlier by Johnson (2001). Our proof for $WS \Rightarrow PN$ uses the Perron-Frobenius theorem, whereas Boman et al. (2005) use the generalized diagonal dominance property of H-matrices.

Equivalence to pairwise-normalizability gives much insight into the set of walk-summable models. For example, the set of unit-diagonal J matrices that are walk-summable is convex, as it is the intersection of $\mathcal{J}_{PN}$ with an affine space. Also, the set of walk-summable J matrices that are sparse with respect to a particular graph G (with some entries of J are restricted to 0) is convex.

Another important class of models are those that have a *diagonally dominant* information matrix, that is, where for each i it holds that $\sum_{j \neq i} |J_{ij}| < J_{ii}$.

14. An alternative definition of pairwise-normalizability is the existence of a decomposition $J = cI + \sum_{e \in E} [J_e]$, where $c > 0$, and $J_e \succeq 0$. For graphs without isolated nodes, both definitions are equivalent.

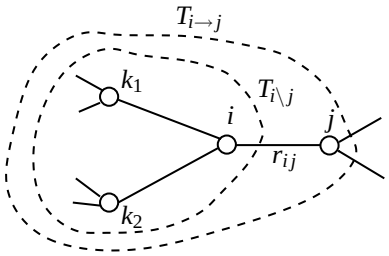


Figure 7: Illustration of the subtree notation, $T_{i\to j}$ and $T_{i\setminus j}$.

Proposition 14 *Diagonally dominant models are pairwise-normalizable (walk-summable).*

A constructive proof is given in Appendix A. The converse does not hold: not all pairwise-normalizable models are diagonally dominant. For instance, in our example of a 4-cycle with a chord shown in Figure 3(a), with $r = .38$ the model is not diagonally dominant but is walk-summable and hence pairwise-normalizable.

4. Walk-sum Interpretation of Belief Propagation

In this section we use the concepts and machinery of walk-sums to analyze belief propagation. We begin with models on trees, for which, as we show, all valid models are walk-summable. Moreover, for these models we show that exact walk-sums over infinite sets of walks for means and variances can be computed efficiently in a recursive fashion. We show that these walk-sum computations map exactly to belief propagation updates. These results (and the computation tree interpretation of LBP recursions) then provide the foundation for our analysis of loopy belief propagation in Section 4.2.

4.1 Walk-Sums and BP on Trees

Our analysis of BP makes use of the following property:

Proposition 15 (Trees are walk-summable) *For tree structured models $J \succ 0 \Leftrightarrow \rho(\bar{R}) \leq 1$ (i.e., all valid trees are walk-summable). Also, for trees $\rho(\bar{R}) = \rho(R) = \lambda_{\max}(R)$.*

Proof. The proof is a special case of the proof of Corollary 3. Trees are non-frustrated (as there are no cycles, let alone frustrated cycles) so they are walk-summable. Negating some variables makes the model attractive and does not change the eigenvalues. □

The proposition shows that walk-sums for means and variances are always defined on tree-structured models, and can be reordered in arbitrary ways without affecting convergence. We rely on this fact heavily in subsequent sections. The next two results identify walk-sum variance and mean computations with the BP update equations. The ingredients for these results are decompositions of the variance and mean walk-sums in terms of sums over walks on subtrees, together with the decomposition in terms of single-revisit and single-visit walks provided in Proposition 9.

Walk-Sum Variance Calculation Let us look first at the computation of the variance at node j, which is equal to the self-return walk-sum $\phi(j \to j)$. This can be computed directly from the single-revisit walk-sum $\alpha_j = \phi(j \stackrel{\setminus j}{\to} j)$ as in Proposition 9. This latter walk-sum can be further decomposed into sums over disjoint subsets of walks each of which corresponds to single-revisit self-return walks that exit node j via a specific one of its neighbors, say i. In particular, as illustrated in Figure 7, the single-revisit self-return walks that do this correspond to walks that live in the subtree $T_{i\to j}$. Using the notation $\mathcal{W}(j \stackrel{\setminus j}{\to} j \mid T_{i\to j})$ for the set of all single-revisit walks which are restricted to stay in subtree $T_{i\to j}$ we see that

$$\alpha_j = \phi(j \stackrel{\setminus j}{\to} j) = \sum_{i\in\mathcal{N}(j)} \phi(j \stackrel{\setminus j}{\to} j \mid T_{i\to j}) \triangleq \sum_{i\in\mathcal{N}(j)} \alpha_{i\to j}\,.$$

Moreover, every single-revisit self-return walk that lives in $T_{i\to j}$ must leave *and* return to node j through the single edge (i,j), and between these first and last steps must execute a (possibly multiple-revisit) self-return walk at node i that is constrained *not* to pass through node j, that is, to live in the subtree $T_{i\setminus j}$ indicated in Figure 7. Thus

$$\alpha_{i\to j} = \phi(j \stackrel{\setminus j}{\to} j \mid T_{i\to j}) = r_{ij}^2 \phi(i \to i \mid T_{i\setminus j}) \triangleq r_{ij}^2 \gamma_{i\setminus j}\,. \tag{14}$$

We next show that the walk-sums α_j and $\alpha_{i\to j}$ (hence variances P_j) can be efficiently calculated by a walk-sum analog of belief propagation. We have the following result:

Proposition 16 *Consider a valid tree model $J = I - R$. Then $\alpha_{i\to j} = -\Delta J_{i\to j}$ and $\gamma_{i\setminus j} = \hat{J}_{i\setminus j}^{-1}$, where $\Delta J_{i\to j}$ and $\hat{J}_{i\setminus j}^{-1}$ are the quantities defined in the Gaussian BP equations (7) and (8).*

See Appendix A for the proof.

Walk-Sum Mean Calculation We extend the above analysis to calculate means in trees. Mean μ_j is the reweighted walk-sum over walks that start anywhere and end at node j, $\mu_j = \phi_h(* \to j)$. Any walk that ends at node j can be expressed as a single-visit walk to node j followed by a multiple-revisit self-return walk from node j: $\phi_h(* \to j) = \left(h_j + \phi_h(* \stackrel{\setminus j}{\to} j)\right)\phi(j \to j)$, where the term h_j corresponds to the length-zero walk that starts and ends at node j.

As we have done for the variances, the single-visit walks to node j can be partitioned into the single-visit walks that reach node j from each of its neighbors, say node i and thus prior to this last step across the edge (i,j), reside in the subtree $T_{i\setminus j}$, so that

$$\beta_{i\to j} \triangleq \phi_h(* \stackrel{\setminus j}{\to} j \mid T_{i\to j}) = r_{ij}\phi_h(* \to i \mid T_{i\setminus j}).$$

Proposition 17 *Consider a valid tree model $J = I - R$. Then $\beta_{i\to j} = \Delta h_{i\to j}$, where $\Delta h_{i\to j}$ is the quantity defined in the Gaussian BP equation (8).*

The proof appears in Appendix A.

4.2 LBP in Walk-Summable Models

In this subsection we use the LBP computation tree to show that LBP includes all the walks for the means, but only a subset of the walks for the variances. This allows us to prove LBP convergence for all walk-summable models. In contrast, for non-walksummable models LBP may or may not converge (and in fact the variances may converge but the means may not). As we will see in Section 5, this can be analyzed by examining walk-summability (and hence validity) of the computation tree, rather than walk-summability of the original model.

As we have discussed, running LBP for some number of iterations yields identical calculations at any particular node i to the exact inference calculations on the corresponding computation tree rooted at node i. We use the notation $T_i^{(n)}$ for the nth computation tree at node i, T_i for the full computation tree (as $n \to \infty$) and we assign the label 0 to the root node. Then, $P_0(T_i^{(n)})$ denotes the variance at the root node of the nth computation tree rooted at node i in G. The LBP variance estimate at node i after n steps is equal to

$$\hat{P}_i^{(n)} = P_0(T_i^{(n)}) = \phi(0 \to 0 \mid T_i^{(n)}).$$

Similarly, the LBP estimate of the mean μ_i after n steps of LBP is

$$\hat{\mu}_i^{(n)} = \mu_0(T_i^{(n)}) = \phi_h(* \to 0 \mid T_i^{(n)}).$$

As we have mentioned, the definition of the computation trees $T_i^{(n)}$ depend upon the message schedule $\{\mathcal{M}^{(n)}\}$ of LBP, which specifies which subset of messages are updated at iteration n. We say that a message schedule is *proper* if every message is updated infinitely often, that is, if for every $m > 0$ and every directed edge (i, j) in the graph there exists $n > m$ such that $(i, j) \in \mathcal{M}^{(n)}$. Clearly, the fully parallel form is proper since every message is updated at every iteration. Serial forms which iteratively cycle through the directed edges of the graph are also proper. All of our convergence analysis in this section presumes a proper message schedule. We remark that as walk-summability ensures convergence of walk-sums independent of the order of summation, it makes the choice of a particular message schedule unimportant in our convergence analysis. The following result is proven in Appendix A.

Lemma 18 (Walks in G and in T_i) *There is a one-to one correspondence between finite-length walks in G that end at i, and walks in T_i that end at the root node. In particular, for each such walk in G there is a corresponding walk in $T_i^{(n)}$ for n large enough.*

Now, recall that to compute the mean μ_i we need to gather walk-sums over all walks that start anywhere and end at i. We have just shown that LBP gathers all of these walks as the computation tree grows to infinity. The story for the variances is different. The true variance P_{ii} is a walk-sum over all self-return walks that start and end at i in G. However, walks in G that start and end at i may map to walks that start at the root node of $T_i^{(n)}$, but end at a *replica* of the root node instead of the root. These walks are not captured by the LBP variance estimate.[15] The walks for the variance estimate $P_0(T_i^{(n)})$ are self-return walks $\mathcal{W}(0 \to 0 \mid T_i^{(n)})$ that start and end at the root node in the

15. Recall that the computation tree is a representation of the computations seen at the root node of the tree, and it is *only* the computation at *this node*—that is, at *this replica* of node i that corresponds to the LBP computation at node i in G.

computation tree. Consider Figure 2. The walk $(1,2,3,1)$ is a self-return walk in the original graph G but is *not* a self-return walk in the computation tree shown in Figure 2(d). LBP variances capture only those self-return walks of the original graph G that are also self-return walks in the computation tree—for example, the walk $(1,3,2,3,4,3,1)$ is a self-return walk in both Figures 2(a) and (d). We call such walks *backtracking*. Hence,

Lemma 19 (Self-return walks in G and in T_i) *The LBP variance estimate at each node is a sum over the backtracking self-return walks in G, a subset of all self-return walks needed to calculate the correct variance.*

Note that back-tracking walks for the variances have positive weights, since each edge in the walk is traversed an even number of times. With each LBP step the computation tree grows and new back-tracking walks are included, hence variance estimates grow monotonically.[16]

We have shown which walks LBP gathers based on the computation tree. The convergence of the corresponding walk-sums remains to be analyzed. In walk-summable models the answer is simple:

Lemma 20 (Computation trees of WS models are WS) *For a walk-summable model all its computation trees $T_i^{(n)}$ (for all n and i) are walk-summable and hence valid.*

Intuitively, walks in the computation tree $T_i^{(n)}$ are subsets of the walks in G, and hence they converge. This implies that the computation trees are walk-summable, and hence valid. This argument can be made precise, but a shorter formal proof using monotonicity of the spectral radius (11) appears in Appendix A. Next, we use these observations to show convergence of LBP for walk-summable models.

Proposition 21 (Convergence of LBP for walk-summable models) *If a model on a graph G is walk-summable, then LBP is well-posed, the means converge to the true means and the LBP variances converge to walk-sums over the backtracking self-return walks at each node.*

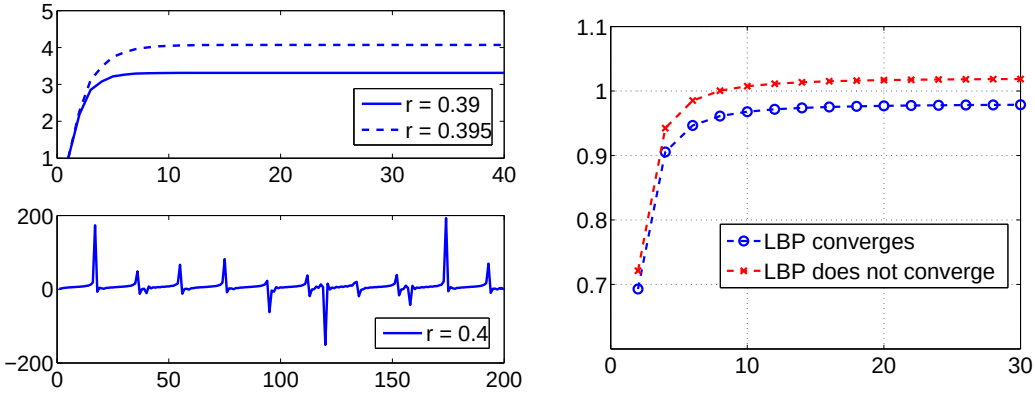
Proof. Let $\mathcal{W}(i \overset{BT}{\to} i)$ denote the back-tracking self-return walks at node i. By Lemmas 18 and 19, we have:

$$\begin{aligned} \mathcal{W}(* \to i) &= \cup_n \mathcal{W}(* \to 0 | T_i^{(n)}) \\ \mathcal{W}(i \overset{BT}{\to} i) &= \cup_n \mathcal{W}(0 \to 0 | T_i^{(n)}). \end{aligned}$$

We note that the computation trees $T_i^{(n)}$ at node i are nested, $T_i^{(n)} \subset T_i^{(n+1)}$ for all n. Hence, $\mathcal{W}(* \to 0|T_i^{(n)}) \subset \mathcal{W}(* \to 0|T_i^{(n+1)})$ and $\mathcal{W}(0 \to 0|T_i^{(n)}) \subset \mathcal{W}(0 \to 0|T_i^{(n+1)})$. Then, by Lemma 7, we obtain the result:

$$\begin{aligned} \mu_i = \phi_h(* \to i) &= \lim_{n\to\infty} \phi_h(* \to 0|T_i^{(n)}) = \lim_{n\to\infty} \hat{\mu}_i^{(n)} \\ P_i^{(BT)} \triangleq \phi(i \overset{BT}{\to} i) &= \lim_{n\to\infty} \phi(0 \to 0|T_i^{(n)}) = \lim_{n\to\infty} \hat{P}_i^{(n)}. \qquad \square \end{aligned}$$

16. Monotonically increasing variance estimates is a characteristic of the particular initialization of LBP that we use, that is, the potential decomposition (4) together with uninformative initial messages. If one instead uses a pairwise-normalized potential decomposition, the variances are then monotonically *decreasing*.



(a) (b)

Figure 8: (a) LBP variances vs. iteration. (b) $\rho(R_n)$ vs. iteration.

Corollary 22 *LBP converges for attractive, non-frustrated, and diagonally dominant models. In attractive and non-frustrated models LBP variance estimates are less than or equal to the true variances (the missing non-backtracking walks all have positive weights).*

In Weiss and Freeman (2001) Gaussian LBP is analyzed for pairwise-normalizable models. They show convergence for the case of diagonally dominant models, and correctness of the means in case of convergence. The class of walk-summable models is strictly larger than the class of diagonally dominant models, so our sufficient condition is stronger. They also show that LBP variances omit some terms needed for the correct variances. These terms correspond to correlations between the root and its replicas in the computation tree. In our framework, each such correlation is a walk-sum over the subset of non-backtracking self-return walks in G that, in the computation tree, begin at a particular replica of the root.

Example 2. Consider the model in Figure 3(a). We summarize various critical points for this model in Figure 9. For $0 \leq r \leq .39039$ the model is walk-summable and LBP converges; then for a small interval $.39039 \leq r \leq .39865$ the model is not walk-summable but LBP still converges, and for larger r LBP does not converge. We apply LBP to this model with $r = 0.39, 0.395$ and 0.4, and plot the LBP variance estimates for node 1 vs. the iteration number in Figure 8(a). LBP converges in the walk-summable case for $r = .39$, with $\rho(\bar{R}) \approx .9990$. It also converges for $r = 0.395$ with $\rho(\bar{R}) \approx 1.0118$, but soon fails to converge as we increase r to 0.4 with $\rho(\bar{R}) \approx 1.0246$.

Also, for $r = .4$, we note that $\rho(R) = .8 < 1$ and the series $\sum_l R^l$ converges (but $\sum_l \bar{R}^l$ does not) and LBP does not converge. Hence, $\rho(R) < 1$ is not sufficient for LBP convergence showing the importance of the stricter walk-summability condition $\rho(\bar{R}) < 1$.

5. LBP in Non-Walksummable Models

While the condition in Proposition 21 is necessary and sufficient for certain special classes of models—for example, for trees and single cycles—it is only sufficient more generally, and, as

in Example 2, LBP may converge for some non-walksummable models. We extend our analysis to develop a tighter condition for convergence of LBP variances based on a weaker form of walk-summability defined with respect to the computation trees (instead of G). We have shown in Proposition 15 that for trees walk-summability and validity are equivalent, and $\rho(\bar{R}) < 1 \Leftrightarrow \rho(R) < 1 \Leftrightarrow J \succ 0$. Hence, our condition essentially corresponds to validity of the computation tree.

First, we note that when a model on G is valid (J is positive definite) but not walk-summable, then some finite computation trees may be invalid (indefinite). This turns out to be the primary reason why belief propagation can fail to converge. Walk-summability on the original graph implies walk-summability (and hence validity) on all of its computation trees. But if the model is not walk-summable, then its computation tree may or may not be valid.

We characterize walk-summability of the computation trees as follows. Let $T_i^{(n)}$ be the nth computation tree rooted at some node i. We define $R_i^{(n)} \triangleq I - J_i^{(n)}$ where $J_i^{(n)}$ is the normalized information matrix for $T_i^{(n)}$ and I is an identity matrix. The nth computation tree $T_i^{(n)}$ is walk-summable (valid) if and only if $\rho(R_i^{(n)}) < 1$ due to the fact that $\rho(\bar{R}_i^{(n)}) = \rho(R_i^{(n)})$ for trees. We are interested in the validity of all finite computation trees, so we consider the quantity $\lim_{n\to\infty} \rho(R_i^{(n)})$. Lemma 23 guarantees the existence of this limit:

Lemma 23 *The sequence $\{\rho(R_i^{(n)})\}$ is monotonically increasing and bounded above by $\rho(\bar{R})$. Thus, $\lim_{n\to\infty} \rho(R_i^{(n)})$ exists, and is equal to $\sup_n \rho(R_i^{(n)})$.*

In the proof we use *k-fold graphs*, which we introduce in Appendix B. The proof appears in Appendix A. The limit in Lemma 23 is defined with respect to a particular root node and message schedule. The next lemma shows that for connected graphs, as long as the message schedule is proper, they do not matter.

Lemma 24 *For connected graphs and with proper message schedule, $\rho_\infty \triangleq \lim_{n\to\infty} \rho(R_i^{(n)})$ is independent of i. The limit does not change by using any other proper message schedule.*

This independence results from the fact that for large n the computation trees rooted at different nodes overlap significantly. Technical details of the proof appear in Appendix A. Using this lemma we suppress the dependence on the root node i from the notation to simplify matters. The limit ρ_∞ turns out to be critical for convergence of LBP variances:

Proposition 25 (LBP validity/variance convergence) *(i) If $\rho_\infty < 1$, then all finite computation trees are valid and the LBP variances converge to walk-sums over the back-tracking self-return walks. (ii) If $\rho_\infty > 1$, then the computation tree eventually becomes invalid and LBP is ill-posed.*

Proof. (i) Since $\rho_\infty = \lim_{n\to\infty} \rho(R^{(n)}) < 1$ and the sequence $\{\rho(R^{(n)})\}$ is monotonically increasing, then there exists $\delta > 0$ such that $\rho(R^{(n)}) \leq 1 - \delta$ for all n. This implies that all the computation trees $T^{(n)}$ are walk-summable and that variances monotonically increase (since weights of back-tracking walks are positive, see the discussion after Lemma 19). We have that $\lambda_{\max}(R^{(n)}) \leq 1 - \delta$, so $\lambda_{\min}(J^{(n)}) \geq \delta$ and $\lambda_{\max}(P^{(n)}) \leq \frac{1}{\delta}$. The maximum eigenvalue of a matrix is a bound on the maximum entry of the matrix, so $(P^{(n)})_{ii} \leq \lambda_{\max}(P^{(n)}) \leq \frac{1}{\delta}$. The variances are monotonically increasing and bounded above, hence they converge.

(ii) If $\lim_{n\to\infty} \rho(R^{(n)}) > 1$, then there exists an m such that $\rho(R^{(n)}) > 1$ for all $n \geq m$. This means that these computation trees $T^{(n)}$ are invalid, and that the variance estimates at some of the nodes

are negative. □

As discussed in Section 2.2, the LBP computation tree is valid if and only if the information parameters $\hat{J}_{i\backslash j}^{(n)}$ and $\hat{J}_i^{(n)}$ in (7), (9) computed during LBP iterations are strictly positive for all n. Hence, it is easily detected if the LBP computation tree becomes invalid. In this case, continuing to run LBP is not meaningful and will lead to division by zero (if the computation tree is singular) or to negative variances (if it is not positive definite).

Recall that the limit ρ_∞ is invariant to message order by Lemma 24. Hence, by Proposition 25, convergence of LBP variances is likewise invariant to message order (except possibly when $\rho_\infty = 1$). The limit ρ_∞ is bounded above by $\rho(\bar{R})$, hence walk-summability in G is a sufficient condition for well-posedness of the computation tree: $\rho_\infty \leq \rho(\bar{R}) < 1$. However, the bound is not tight in general (except for trees and single cycles). This is related to the phenomenon that the limit of the spectral radius of the finite computation trees can be less than the spectral radius of the infinite computation tree (which has no leaf nodes). See He et al. (2000) for analysis of a related discrepancy.

Means in non-WS models For the case where $\rho_\infty < 1 < \rho(\bar{R})$, the walk-sums for LBP variances converge absolutely (see proof of Proposition 25), but the walk-sums for the means do not. The reason is that LBP only computes a subset of the self-return walks for the variances but captures all the walks for the means. However, the series LBP computes for the means, corresponding to a particular ordering of walks, may still converge.

It is well known (Rusmevichientong and Van Roy, 2001) that once variances converge, the updates for the means follow a linear system. Consider (7) and (8) with $\hat{J}_{i\backslash j}$ fixed, then the LBP messages for the means $\Delta h = (\Delta h_{i\to j} \mid \{i,j\} \in E)$ follow a linear system update. For the parallel message schedule we can express this as:

$$\Delta h^{(n+1)} = L\, \Delta h^{(n)} + b \tag{15}$$

for some matrix L and some vector b. Convergence of this system depends on the spectral radius $\rho(L)$. However, it is difficult to analyze $\rho(L)$ since the matrix L depends on the converged values of the LBP variances. To improve convergence of the means, one can damp the message updates by modifying (8) as follows:

$$\Delta h_{i\to j}^{(n+1)} = (1-\alpha)\Delta h_{i\to j}^{(n)} + \alpha(-J_{ij}(\hat{J}_{i\backslash j}^{(n)})^{-1}\hat{h}_{i\backslash j}^{(n)}) \;\text{ with }\; 0 < \alpha \leq 1. \tag{16}$$

We have observed in experiments that for all the cases where variances converge we also obtain convergence of the means with enough damping of BP messages. We have also tried damping the updates for the ΔJ messages, but whether or not variances converge appears to be independent of damping. Apparently, it is the validity of the computation tree ($\rho_\infty < 1$) that is essential for convergence of both means and variances in damped versions of Gaussian LBP.

Example 3. We illustrate Proposition 25 on a simple example. Consider the 5-node cycle model from Figure 3(b). In Figure 8(b), for $\rho = .49$ we plot $\rho(R_n)$ vs. n (lower curve) and observe that $\lim_{n\to\infty}\rho(R_n) \approx .98 < 1$, and LBP converges. For $\rho = .51$ (upper curve), the model defined on the 5-node cycle is still valid but $\lim_{n\to\infty}\rho(R_n) \approx 1.02 > 1$ so LBP is ill-posed and does not converge.

As we mentioned, in non-walksummable models the series that LBP computes for the means is not absolutely convergent and may diverge even when variances converge. For our 4-cycle with a chord example in Figure 3(a), the region where variances converge but means diverge is very

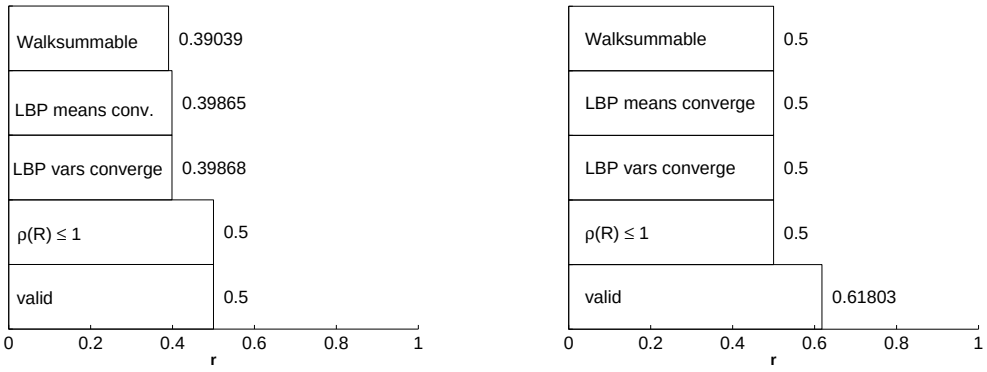


Figure 9: Critical regions for example models from Figure 3. (a) 4-cycle with a chord. (b) 5-cycle.

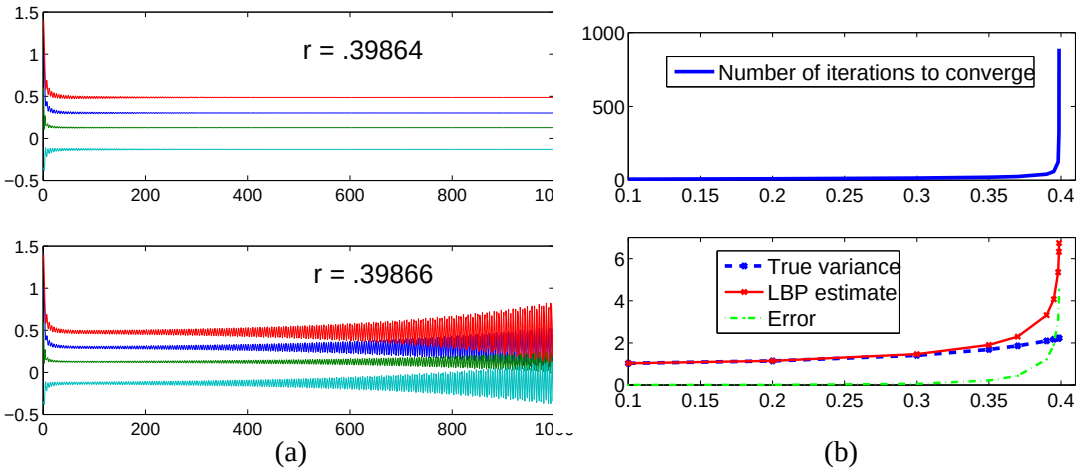


Figure 10: The 4-cycle with a chord example. (a) Convergence and divergence of the means near the LBP mean critical point. (b) Variance near the LBP variance critical point: (top) number of iterations for variances to converge, (bottom) true variance, LBP estimate and the error at node 1.

narrow, $r \approx .39865$ to $r \approx .39867$ (we use the parallel message schedule here; the critical point for the means is slightly higher using a serial schedule). In Figure 10(a) we show mean estimates vs. the iteration number on both sides of the LBP mean critical point for $r = 0.39864$ and for $r = 0.39866$. In the first case the means converge, while in the latter they slowly but very definitely diverge. The spectral radius of the linear system for mean updates in (15) for the two cases is $\rho(L) = 0.99717 < 1$ and $\rho(L) = 1.00157 > 1$ respectively. In the divergent example, all the eigenvalues of L have real components less than 1 (the maximum such real component is $0.8063 < 1$). Thus by damping we can force all the eigenvalues of L to enter the unit circle: the damped linear system is $(1-\alpha)I + \alpha L$. Using $\alpha = 0.9$ in (16) the means converge.

In Figure 10(b) we illustrate that near the LBP variance critical point, the LBP estimates become more difficult to obtain and their quality deteriorates dramatically. We consider the graph in Figure

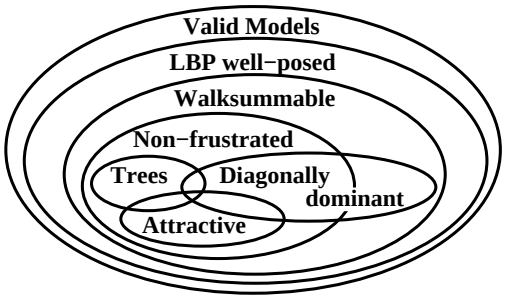


Figure 11: Venn diagram summarizing various subclasses of Gaussian models.

3(a) again as r approaches 0.39867, the critical point for the convergence of the variances. The picture shows that the number of iterations as well as the error in LBP variance estimates explode near the critical point. In the figure we show the variance at node 1, but similar behavior occurs at every node. In Figure 9, we summarize the critical points of both models from Figure 3.

6. Conclusion

We have presented a walk-sum interpretation of inference in Gaussian graphical models, which holds for a wide class of models that we call walk-summable. We have shown that walk-summability encompasses many classes of models which are considered "easy" for inference—trees, attractive, non-frustrated and diagonally dominant models—but also includes many models outside of these classes. A Venn diagram summarizing relations between these sets appears in Figure 11. We have also shown the equivalence of walk-summability to pairwise-normalizability.

We have established that in walk-summable models LBP is guaranteed to converge, for both means and variances, and that upon convergence the means are correct, whereas the variances only capture walk-sums over back-tracking walks. We have also used the walk-summability of valid (i.e., positive definite) models on trees to develop a more complete picture of LBP for non-walksummable models, relating variance convergence to validity of the LBP computation tree.

There are a variety of directions in which these results can be extended. One involves developing improved walk-sum algorithms that gather more walks than LBP does, to yield better variance estimates. Results along these lines—involving vectors of variables at each node as well as factor graph versions of LBP that group larger sets of variables—will be presented in a future publication. Another direction is to apply walk-sum analysis to other algorithms for Gaussian inference, for example, Chandrasekaran et al. are applying walk-sums to better understand the embedded trees algorithm (Sudderth et al., 2004).

Our current work is limited to Gaussian models, as walk-sums arise from the power series expansion for the matrix inverse. However, related expansions of correlations in terms of walks have been investigated for other models. Fisher (1967) developed an approximation to the pairwise correlations in Ising models based on self-avoiding walks. Brydges et al. (1983) use walk-sums for non-Gaussian classical and quantum spin-systems, where the weights of walks involve complicated multi-dimensional integrals. It would be very useful to develop ways to compute or approximate self-avoiding or non-Gaussian walk-sums efficiently and extend the walk-sum perspective to inference in a broader class of models.

Appendix A. Detailed Proofs

Proof of Proposition 1 *Proof of (i) ⇒ (ii).* We examine convergence of the matrix series in (ii) element-wise. First note that $(\bar{R}^l)_{ij}$ is an absolute walk-sum over all walks of length l from i to j:

$$(\bar{R}^l)_{ij} = \sum_{w:i\xrightarrow{l}j} |\phi(w)|$$

(there are a finite number of these walks so the sum is well-defined). Now, if (i) holds then using properties of absolute convergence we can order the sum $\sum_{w:i\to j}|\phi(w)|$ however we wish and it still converges. If we order walks by their length and then group terms for walks of equal lengths (each group has a finite number of terms) we obtain:

$$\sum_{w:i\to j}|\phi(w)| = \sum_l \sum_{w:i\xrightarrow{l}j}|\phi(w)| = \sum_l (\bar{R}^l)_{ij}\,. \tag{17}$$

Therefore, the series $\sum_l(\bar{R}^l)_{ij}$ converges for all i,j.

Proof of (ii) ⇒ (i). To show convergence of the sum $\sum_{w:i\to j}|\phi(w)|$ it is sufficient to test convergence for any convenient ordering of the walks. As shown in (17), $\sum_l(\bar{R}^l)_{ij}$ corresponds to one particular ordering of the walks which converges by (ii). Therefore, the walk-sums in (i) converge absolutely.

Proof of (ii) ⇔ (iii). This is a standard result in matrix analysis (Varga, 2000).

Proof of (iii) ⇔ (iv). Note that λ is an eigenvalue of $\bar{R}$ if and only if $1-\lambda$ is an eigenvalue of $I-\bar{R}$ ($\bar{R}x=\lambda x \Leftrightarrow (I-\bar{R})x=(1-\lambda)x$). Therefore, $\lambda_{\min}(I-\bar{R}) = 1-\lambda_{\max}(\bar{R})$. According to the Perron-Frobenius theorem, $\rho(\bar{R})=\lambda_{\max}(\bar{R})$ because $\bar{R}$ is non-negative. Thus, $\rho(\bar{R}) = 1-\lambda_{\min}(I-\bar{R})$ and we have that $\rho(\bar{R})<1 \Leftrightarrow \lambda_{\min}(I-\bar{R})>0$. □

Proof of Corollary 3 We will show that for any non-frustrated model there exists a diagonal D with $D_{ii}=\pm 1$, that is, a signature matrix, such that $DRD=\bar{R}$. Hence, R and $\bar{R}$ have the same eigenvalues, because $DRD=DRD^{-1}$ is a similarity transform which preserves the eigenvalues of a matrix. It follows that $I-R\succ 0$ implies $I-\bar{R}\succ 0$ and walk-summability of J by Proposition 1(iv).

Now we describe how to construct a signature similarity which makes R attractive for non-frustrated models. We show how to split the vertices into two sets V^+ and V^- such that negating V^- makes the model attractive. Find a spanning tree T of the graph G. Pick a node i. Assign it to V^+. For any other node j, there is a unique path to i in T. If the product of edge weights along the path is positive, then assign j to V^+, otherwise to V^-. Now, since the model is non-frustrated, all edges $\{j,k\}$ in G such that $j,k\in V^+$ are positive, all edges with $j,k\in V^-$ are positive, and all edges with $j\in V^+$ and $k\in V^-$ are negative. This can be seen by constructing the cycle that goes from j to i to k in T and crosses the edge $\{k,j\}$ to close itself. If $j,k\in V^+$ then the paths j to i and i to k have a positive weight, hence in order for the cycle to have a positive weight, the last step $\{k,j\}$ must also have a positive weight. The other two cases are similar. Now let D be diagonal with $D_{ii}=1$ for $i\in V^+$, and $D_{ii}=-1$ for $i\in V^-$. Then $DRD=\begin{bmatrix} R_{V^+} & -R_{V^+,V^-} \\ -R_{V^-,V^+} & R_{V^-}\end{bmatrix}\geq 0$, that is, $DRD=\bar{R}$. □

Proof of Proposition 4 *Proof of WS ⇒ (i).* WS is equivalent to $\rho(\bar{R})<1$ by Proposition 1. But $\rho(R)\leq\rho(\bar{R})$ by (11). Hence, $\rho(\bar{R})<1\Rightarrow\rho(R)<1$.

Proof of (i) $\Rightarrow$ (ii). Given $J = I - R$, it holds that $\lambda_{\min}(J) = 1 - \lambda_{\max}(R)$. Also, $\lambda_{\max}(R) \leq \rho(R)$. Hence, $\lambda_{\min}(J) = 1 - \lambda_{\max}(R) \geq 1 - \rho(R) > 0$ for $\rho(R) < 1$.

Proof of (i) $\Rightarrow$ (iii). This is a standard result in matrix analysis. $\square$

Proof of Proposition 10 Assume that G is connected (otherwise we apply the proof to each connected component, and the spectral radii are the maxima over the respective connected components). We prove that $\rho(\bar{R}) = \rho(\hat{R})$. By the Perron-Frobenius theorem, there exists a positive vector x such that $\bar{R}x = \rho(\bar{R})x$. Let $\hat{x} = (x;x)$. Then $\hat{R}\hat{x} = \rho(\bar{R})\hat{x}$ because

$$(\hat{R}\hat{x})_{\pm} = (R_+ + R_-)x = \bar{R}x = \rho(\bar{R})x.$$

Hence, $\rho(\bar{R})$ is an eigenvalue of $\hat{R}$ with positive eigenvector $\hat{x}$. First suppose that $\hat{G}$ is connected. Then, by the Perron-Frobenius theorem, $\rho(\bar{R}) = \rho(\hat{R})$ because $\hat{R}$ has a unique positive eigenvector which has eigenvalue equal to $\rho(\hat{R})$. Now, $\hat{J} = I - \hat{R} \succ 0 \Leftrightarrow \hat{J}$ is WS $\Leftrightarrow \rho(\hat{R}) < 1 \Leftrightarrow \rho(\bar{R}) \leq 1 \Leftrightarrow J = I - R$ is WS. If $\hat{G}$ is disconnected then $\hat{R}$ is a block-diagonal matrix with two copies of $\bar{R}$ (after relabeling the nodes), so $\rho(\hat{R}) = \rho(\bar{R})$. $\square$

Proof of Proposition 11 We partition walk-sums into sums over "even" and "odd" walks according to the number of negative edges crossed by the walk. Thus a walk w is even if $\phi(w) > 0$ and is odd if $\phi(w) < 0$. The graph $\hat{G}$ is defined so that every walk from i_+ to j_+ is even and every walk from i_+ to j_- is odd. Thus,

$$\begin{aligned} P_{ij} &= \sum_{\text{even } w:i\to j} \phi(w) + \sum_{\text{odd } w:i\to j} \phi(w) \\ &= \sum_{w:i_+\to j_+} \hat{\phi}(w) - \sum_{w:i_+\to j_-} \hat{\phi}(w) \\ &= \hat{P}_{i_+,j_+} - \hat{P}_{i_+,j_-}\,. \end{aligned}$$

The second part of the the proposition follows by similar logic. Now we classify a walk as even if $h_{w_0}\phi(w) > 0$ and as odd if $h_{w_0}\phi(w) < 0$. Note also that setting $\hat{h} = (h_+;h_-)$ has the effect that all walks with $h_{w_0} > 0$ begin in V_+ and all walks with $h_{w_0} < 0$ begin in V_-. Consequently, every even walk ends in V_+ and every odd walk ends in V_-. Thus,

$$\begin{aligned} \mu_i &= \sum_{\text{even } w:*\to i} h_*\phi(w) + \sum_{\text{odd } w:*\to i} h_*\phi(w) \\ &= \sum_{w:*\to i_+} \hat{h}_*\hat{\phi}(w) - \sum_{w:*\to i_-} \hat{h}_*\hat{\phi}(w) \\ &= \hat{\mu}_{i_+} - \hat{\mu}_{i_-} \;\square \end{aligned}$$

Proof of Proposition 12 Take J_1 and J_2 pairwise-normalizable. Take any $\alpha, \beta \geq 0$ such that at least one of them is positive. Then $\alpha J_1 + \beta J_2$ is also pairwise-normalizable simply by taking the same weighted combinations of each of the J_e matrices for J_1 and J_2. Setting $\beta = 0$ shows that $\mathcal{J}_{PN}$ is a cone, and setting $\beta = 1 - \alpha$ shows convexity. The cone is pointed since it is a subset of the cone of semidefinite matrices, which is pointed. $\square$

Proof of Proposition 13 *Proof of PN ⇒WS.* It is evident that any J matrix which is pairwise-normalizable is positive definite. Furthermore, reversing the sign of the partial correlation coefficient on edge e simply negates the off-diagonal element of J_e which does not change the value of $\det J_e$ so that we still have $J_e \succeq 0$. Thus, we can make all the negative coefficients positive and the resulting model $I - \bar{R}$ is still pairwise-normalizable and hence positive definite. Then, by Proposition 1(iv), $J = I - R$ is walk-summable.

Proof of WS ⇒PN. Given a walk-summable model $J = I - R$ we construct a pairwise-normalized representation of the information matrix. We may assume the graph is connected (otherwise, we may apply the following construction for each connected component of the graph). Hence, by the Perron-Frobenius theorem there exists a positive eigenvector $x > 0$ of $\bar{R}$ such that $\bar{R}x = \lambda x$ and $\lambda = \rho(\bar{R}) > 0$. Given (x, λ) we construct a representation $J = \sum_e [J_e]$ where for $e = \{i, j\}$ we set:

$$J_e = \begin{pmatrix} \frac{|r_{ij}|x_j}{\lambda x_i} & -r_{ij} \\ -r_{ij} & \frac{|r_{ij}|x_i}{\lambda x_j} \end{pmatrix}.$$

This is well-defined (there is no division by zero) since x and λ are positive. First, we verify that $J = \sum_{e \in E} [J_e]$. It is evident that the off-diagonal elements of the edge matrices sum to $-R$. We check that the diagonal elements sum to one:

$$\sum_e [J_e]_{ii} = \frac{1}{\lambda x_i} \sum_j |r_{ij}| x_j = \frac{(\bar{R}x)_i}{\lambda x_i} = \frac{(\lambda x)_i}{\lambda x_i} = 1.$$

Next, we verify that each J_e is positive definite. This matrix has positive diagonal and determinant

$$\det J_e = \left(\frac{|r_{ij}|x_j}{\lambda x_i} \right) \left(\frac{|r_{ij}|x_i}{\lambda x_j} \right) - (-r_{ij})^2 = r_{ij}^2 \left(\frac{1}{\lambda^2} - 1 \right) > 0.$$

The inequality follows from walk-summability because $0 < \lambda < 1$ and hence $\left(\frac{1}{\lambda^2} - 1\right) > 0$. Thus, $J_e \succ 0$. □

Proof of Proposition 14 Let $a_i = J_{ii} - \sum_{j \neq i} |J_{ij}|$. Note that $a_i > 0$ follows from diagonal dominance. Let $\deg(i)$ denote the degree of node i in G. Then, $J = \sum_{e \in E} [J_e]$ where for edge $e = \{i, j\}$ we set

$$J_e = \begin{pmatrix} |J_{ij}| + \frac{a_i}{\deg(i)} & J_{ij} \\ J_{ij} & |J_{ij}| + \frac{a_j}{\deg(j)} \end{pmatrix}$$

with all other elements of $[J_e]$ set to zero. Note that:

$$\sum_e [J_e]_{ii} = \sum_{j \in \mathcal{N}(i)} \left(|J_{ij}| + \frac{a_i}{\deg(i)} \right) = a_i + \sum_{j \in \mathcal{N}(i)} |J_{ij}| = J_{ii}\,.$$

Also, J_e has positive diagonal elements and has determinant $\det(J_e) > 0$. Hence, $J_e \succ 0$. Thus, J is pairwise-normalizable. □

Proof of Proposition 16 To calculate the walk-sum for multiple-revisit self-return walks in $T_{i\backslash j}$, we can use the single-revisit counterpart:

$$\gamma_{i\backslash j} = \phi(i \to i \mid T_{i\backslash j}) = \frac{1}{1 - \phi\left(i \overset{\backslash i}{\to} i \mid T_{i\backslash j}\right)} \ . \tag{18}$$

Now, we decompose the single-revisit walks in the subtree $T_{i\backslash j}$ in terms of the possible first step of the walk (i,k), where $k \in \mathcal{N}(i)\backslash j$. Hence,

$$\phi(i \overset{\backslash i}{\to} i \mid T_{i\backslash j}) = \sum_{k \in \mathcal{N}(i)\backslash j} \phi(i \overset{\backslash i}{\to} i \mid T_{k\to i}). \tag{19}$$

Using (14), (18), and (19), we are able to represent the walk-sum $\phi(j \overset{\backslash j}{\to} j \mid T_{i\to j})$ in $T_{i\to j}$ in terms of the walk-sums $\phi(i \overset{\backslash i}{\to} i \mid T_{k\to i})$ on smaller subtrees $T_{k\to i}$. This is the basis of the recursive calculation:

$$\alpha_{i\to j} = r_{ij}^2 \frac{1}{1 - \sum_{k \in \mathcal{N}(i)\backslash j} \alpha_{k\to i}} \ .$$

These equations look strikingly similar to the belief propagation updates. Combining (7) and (8) from Section 2.1 we have:

$$-\Delta J_{i\to j} = J_{ij}^2 \frac{1}{J_{ii} + \sum_{k \in \mathcal{N}(i)\backslash j} \Delta J_{k\to i}} \ .$$

It is evident that the recursive walk-sum equations can be mapped exactly to belief propagation updates. In normalized models $J_{ii} = 1$. We have the message update $\alpha_{i\to j} = -\Delta J_{i\to j}$, and the variance estimate in the subtree $T_{i\backslash j}$ is $\gamma_{i\backslash j} = \hat{J}^{-1}_{i\backslash j}$. □

Proof of Proposition 17 A multiple-revisit walk in $T_{i\backslash j}$ can be written in terms of single-visit walks:

$$\phi_h(* \to i \mid T_{i\backslash j}) = \left(h_i + \phi_h(* \overset{\backslash i}{\to} i \mid T_{i\backslash j})\right) \phi(i \to i \mid T_{i\backslash j}).$$

We already have $\gamma_{i\backslash j} = \phi(i \to i \mid T_{i\backslash j})$ from (18). The remaining term $\phi_h(* \overset{\backslash i}{\to} i \mid T_{i\backslash j})$ can be decomposed by the subtrees in which the walk lives:

$$\phi_h(* \overset{\backslash i}{\to} i \mid T_{i\backslash j}) = \sum_{k \in \mathcal{N}(i)\backslash j} \phi_h(* \overset{\backslash i}{\to} i \mid T_{k\to i}).$$

Thus we have the recursion:

$$\beta_{i\to j} = r_{ij}\gamma_{i\backslash j}(h_i + \sum_{k \in \mathcal{N}(i)\backslash j} \beta_{k\to i}).$$

To compare this to the Gaussian BP updates, let us combine (7) and (8) in Section 2.2:

$$\Delta h_{i\to j} = -J_{ij}\hat{J}^{-1}_{i\backslash j}\left(h_i + \sum_{k \in \mathcal{N}(i)\backslash j} \Delta h_{k\to i}\right).$$

Thus BP updates for the means can also be mapped exactly into recursive walk-sum updates via $\beta_{i\to j} = \Delta h_{i\to j}$. □

Proof of Lemma 18 First, we note that for every walk w which ends at the root node of $T_i^{(n)}$ there is a corresponding walk in G which ends at i. The reason is that the neighbors of a given node j in $T_i^{(n)}$ correspond to a subset of the neighbors of j in G. Hence, for each step (w_k, w_{k+1}) of the walk in $T_i^{(n)}$ there is a corresponding step in G.

Next, we show that every walk $w = (w_0, \ldots, w_l)$ in G is contained in $T_{w_l}^{(n)}$ for some n. First consider the parallel message schedule, for which the computation tree $T_{w_l}^{(n)}$ grows uniformly. Then for any walk in G that ends at w_l and has length n there is a walk in $T_{w_l}^{(n)}$ that ends at the root.

The intuition for other message schedules is that every step (i, j) of the walk will appear eventually in any proper message schedule $\mathcal{M}$. A formal proof is somewhat technical. First we unwrap the walk w into a tree T_w rooted at w_l in the following way: start at w_l, the end of the walk, and traverse the walk in reverse. First add the edge $\{w_l, w_{l-1}\}$ to T_w. Now, suppose we are at node w_k in T_w and the next step in w is $\{w_k, w_{k-1}\}$. If w_{k-1} is already a neighbor of w_k in T_w then set the current node in T_w to w_{k-1}. Otherwise create a new node w_{k-1} and add the edge to T_w. It is clear that loops are never made in this procedure, so T_w is a tree.

We now show for any proper message schedule $\mathcal{M}$ that T_w is part of the computation tree $T_{w_l}^{(n)}$ for some n. Pick a leaf edge $\{i_1, j_1\}$ of T_w. Since $\{\mathcal{M}^{(n)}\}$ is proper, there exist n_1 such that $(i_1, j_1) \in \mathcal{M}^{(n_1)}$. Now $(i_1, j_1) \in T_{i_1 \to j_1}^{(n_1)}$, and the edge appears at the root of $T_{i_1 \to j_1}^{(n_1)}$. Also, $T_{i_1 \to j_1}^{(n_1)} \subset T_{i_1 \to j_1}^{(m)}$ for $m > n_1$, so this holds for all subsequent steps as well. Now remove $\{i_1, j_1\}$ from T_w and pick another leaf edge $\{i_2, j_2\}$. Again, since $\{\mathcal{M}^{(n)}\}$ is proper, there exist $n_2 > n_1$ such that $(i_2, j_2) \in \mathcal{M}^{(n_2)}$. Remove $\{i_2, j_2\}$ from T_w, and continue similarly. At each such point n_k of eliminating some new edge $\{i_k, j_k\}$ of T_w, the whole eliminated subtree of T_w extending from $\{i_k, j_k\}$ has to belong to $T_{i_k \to j_k}^{(n_k)}$. Continue until just the root of T_w remains at step n. Now the computation tree $T_{w_l}^{(n)}$ (which is created by splicing together $T_{i \to j}^{(n)}$ for all edges (i, j) coming into the root of T_w) contains T_w, and hence it contains the walk w. □

Proof of Lemma 20 This result comes as an immediate corollary of Proposition 28, which states that $\rho(R_i^{(n)}) \leq \rho(\bar{R})$ (here $R_i^{(n)}$ is the partial correlation matrix for $T_i^{(n)}$). For WS models, $\rho(\bar{R}) < 1$ and the result follows. □

Proof of Lemma 23 The fact that the sequence $\{\rho(R_i^{(n)})\}$ is bounded by $\rho(\bar{R})$ is a nontrivial fact, proven in Appendix B using a *k-fold graph* construction. To prove monotonicity, note first that for trees $\rho(R_i^{(n)}) = \rho(\bar{R}_i^{(n)})$. Also, note that all of the variables in the computation tree $T_i^{(n)}$ are also present in T_i^{n+1}. We zero-pad $\bar{R}_i^{(n)}$ to make it the same size as $\bar{R}_i^{(n+1)}$ (this does not change the spectral radius). Then it holds that $\bar{R}_i^{(n)} \leq \bar{R}_i^{(n+1)}$ element-wise. Using (11), it follows that $\rho(\bar{R}_i^{(n)}) \leq \rho(\bar{R}_i^{(n+1)})$, establishing monotonicity. □

Proof of Lemma 24 Let $T_i^{(n)}(\mathcal{M})$ denote the nth computation tree under a proper message schedule $\mathcal{M}$ rooted at node i. We use the following simple extension of Lemma 18: Let $T_i^{(n)}(\mathcal{M}_1)$ be the nth computation tree rooted at i under message schedule $\mathcal{M}_1$. Take any node in $T_i^{(n)}(\mathcal{M}_1)$ which is a replica of node j in G. Then there exists m such that $T_i^{(n)}(\mathcal{M}_1) \subset T_j^{(m)}(\mathcal{M}_2)$, where $\mathcal{M}_2$ is another

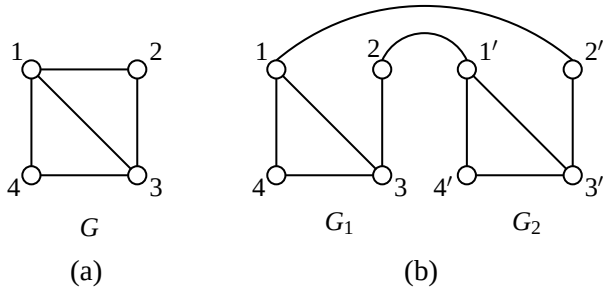


Figure 12: Illustration of (a) graph G and (b) a 2-fold graph of G.

message schedule. The proof parallels that of Lemma 18: the tree $T_i^{(n)}(\mathcal{M}_1)$ has a finite number of edges, and we use induction adding one edge at a time.

Consider message schedule $\mathcal{M}_1$. By Lemma 23, $\rho_i \triangleq \lim_{n\to\infty} \rho(R_i^{(n)}(\mathcal{M}_1))$ exists. For any ε pick an L such that for $n \geq L$ it holds that $|\rho(R_i^{(n)}(\mathcal{M}_1)) - \rho_i| \leq \frac{\varepsilon}{2}$. Pick a replica of node j inside $T_i^{(L)}(\mathcal{M}_1)$. Then using the property from the previous paragraph, there exists M such that $T_i^{(L)}(\mathcal{M}_1) \subset T_j^{(M)}(\mathcal{M}_2)$. Similarly there exists N such that $T_j^{(M)}(\mathcal{M}_2) \subset T_i^{(N)}(\mathcal{M}_1)$. It follows that $\overline{R}_i^{(L)}(\mathcal{M}_1) \leq \overline{R}_j^{(M)}(\mathcal{M}_2) \leq \overline{R}_i^{(N)}(\mathcal{M}_1)$, where we zero-pad the first two matrices to have the same size as the last one. Then, $\rho(\overline{R}_i^{(L)}(\mathcal{M}_1)) \leq \rho(\overline{R}_j^{(M)}(\mathcal{M}_2)) \leq \rho(\overline{R}_i^{(N)}(\mathcal{M}_1))$. Then it holds that $\rho_i - \frac{\varepsilon}{2} \leq \rho(\overline{R}_j^{(M)}(\mathcal{M}_2)) \leq \rho_i + \frac{\varepsilon}{2}$. Hence, $|\rho(\overline{R}_j^{(M)}(\mathcal{M}_2)) - \rho_i| \leq \varepsilon$, and $\lim_{n\to\infty} \rho(\overline{R}_j^{(n)}(\mathcal{M}_2)) = \rho_i$. □

Appendix B. K-fold Graphs and Proof of Boundedness of $\rho(R_i^{(n)})$.

Consider an arbitrary graph $G = (V, E)$. Suppose that we have a pairwise MRF defined on G with self potentials $\psi_i(x_i)$, for $v_i \in V$ and pairwise potentials $\psi_{ij}(x_i, x_j)$ for $(v_i, v_j) \in E$. We construct a family of K-fold graphs based on G as follows:

1. Create K disconnected copies G_k, $k \in \{1, .., K\}$ of G, with nodes $v_i^{(k)}$, and edges $(v_i^{(k)}, v_j^{(k)})$. The nodes and the edges of G_k are labeled in the same way as the ones of G. The potentials ψ_i and ψ_{ij} are copied to the corresponding nodes and edges in all G_k.

2. Pick some pair of graphs G_k, G_l, and choose an edge (v_i, v_j) in G. We flip the corresponding edges in G_k and G_l, edges $(v_i^{(k)}, v_j^{(k)})$ and $(v_i^{(l)}, v_j^{(l)})$ become $(v_i^{(k)}, v_j^{(l)})$ and $(v_i^{(l)}, v_j^{(k)})$. The pairwise potentials are adjusted accordingly.

3. Repeat step 2 an arbitrary number of times for a different pair of graphs G_k, or a different edge in G.

An illustration of the procedure appears in Figure 12. The original graph G is a 4-cycle with a chord. We create a 2-fold graph based on G by flipping the edges $(1,2)$ in G_1 and $(1', 2')$ in G_2.

Now we apply the K-fold graph construction to Gaussian MRF models. Suppose that we have a model with information parameters J and h on G. Suppose that J is normalized to have unit-diagonal. Let G^K be a K-fold graph based on G with the information matrix J^K (which is also

unit-diagonal by construction). Also, let $T_i^{(n)}$ be the nth computation tree for the original graph, and $J_i^{(n)}$ the corresponding information matrix (also unit-diagonal). Let $R = I - J$, $R^K = I^K - J^K$, and $R_i^{(n)} = I^{(n)} - J_i^{(n)}$ (here I, I^K, and $I^{(n)}$ are identity matrices of appropriate dimensions).

Lemma 26 (Spectral radii of R and R^K) *For any K-fold graph G^K based on G: $\rho(\bar{R}^K) = \rho(\bar{R})$.*

Proof. Suppose that G is connected (otherwise apply the proof to each connected component of G, and the spectral radius for G will be the maximum of the spectral radii for the connected components).

Then, by the Perron-Frobenius theorem there exists a vector $x > 0$ such that $\bar{R}x = \rho(\bar{R})x$. Create a K-fold vector x^K by copying entry x_i into each of the K corresponding entries of x^K. Then x^K is positive, and it also holds that $\bar{R}^K x^K = \rho(\bar{R})x^K$ (since the local neighborhoods in G and G^K are the same). Now $\bar{R}^K$ is a non-negative matrix, and x^K is a positive eigenvector, hence it achieves the spectral radius of $\bar{R}^K$ by the Perron-Frobenius theorem. Thus, $\rho(\bar{R}) = \rho(\bar{R}^K)$. □

The construction of a K-fold graph based on G has parallels with the computation tree on G. The K-fold graph is locally equivalent to G and the computation tree, except for its leaf nodes, is also locally equivalent to G. We show next that the computation tree $T_i^{(n)}$ is contained in some G^K for K large enough.

Lemma 27 (K-fold graphs and computation trees) *Consider a computation tree $T_i^{(n)}$ corresponding to graph G. There exists a K-fold graph G^K, which contains $T_i^{(n)}$ as a subgraph, for K large enough.*

Proof. We provide a simple construction of a K-fold graph, making no attempt to minimize K. Let $T_i^{(n)} = (V_n, E_n)$. Each node $v' \in V_n$ corresponds to some node $v \in V$ in G. We create a K-fold graph G^K by making a copy $G_{v'}$ of G for every node $v' \in T_i^{(n)}$. Hence $K = |V_n|$. For each edge $(u', v') \in E_n$ in the computation tree, we make an edge flip between nodes in graphs $G_{u'}$ and $G_{v'}$ that correspond to u and v in G. This operation is well-defined because edges in $T_i^{(n)}$ that map to the same edge in G do not meet. Thus, the procedure creates G^K which contains $T_i^{(n)}$ as a subgraph. □

Finally, we use the preceding lemmas to prove a bound on the spectral radii of the matrices $R_i^{(n)}$ for the computation tree $T_i^{(n)}$.

Proposition 28 (Bound on $\rho(R_i^{(n)})$) *For computation tree $T_i^{(n)}$: $\rho(R_i^{(n)}) \leq \rho(\bar{R})$.*

Proof. Consider a computation tree $T_i^{(n)}$. Recall that $\rho(R_i^{(n)}) = \rho(\bar{R}_i^{(n)})$, since $T_i^{(n)}$ is a tree. Use Lemma 27 to construct a K-fold graph G^K which has $T_i^{(n)}$ as a subgraph. Zero-padding $\bar{R}_i^{(n)}$ to have the same size as $\bar{R}^K$, it holds that $\bar{R}_i^{(n)} \leq \bar{R}^K$. Since $\bar{R}_i^{(n)} \leq \bar{R}^K$, using (11) and Lemma 26 we have: $\rho(R_i^{(n)}) \leq \rho(\bar{R}^K) = \rho(\bar{R})$. □

References

E. G. Boman, D. Chen, O. Parekh, and S. Toledo. On factor width and symmetric H-matrices. *Linear Algebra and its Applications*, 405, 2005.

D. C. Brydges, J. Frohlich, and A. D. Sokal. The random-walk representation of classical spin systems and correlation inequalities. *Communications in mathematical physics*, 91, 1983.

V. Chandrasekaran, J. Johnson, and A. Willsky. Walk-sum analysis and convergence of embedded subgraph algorithms. In preparation.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, 1999.

M. Fisher. Critical temperatures of anisotropic Ising lattices II, general upper bounds. *Physical Review*, 1967.

R. Godement. *Analysis I*. Springer-Verlag, 2004.

J. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.

L. He, X. Liu, and G. Strang. Laplacian eigenvalues of growing tees. In *Conf. on Math. Theory of Networks and Systems*, 2000.

R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

R. Horn and C. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

A. T. Ihler, J. W. Fisher III, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6, May 2005.

J. Johnson. Walk-summable Gauss-Markov random fields. Unpublished manuscript, available at `http://www.mit.edu/people/jasonj`, December 2001.

J. Johnson, D. Malioutov, and A. Willsky. Walk-sum interpretation and analysis of Gaussian belief propagation. In *Advances in Neural Information Processing Systems*, 2006.

B. Jones and M. West. Covariance decomposition in undirected Gaussian graphical models. *Biometrika*, 92(4), 2005.

S. Kirkland, J. J. McDonald, and M. Tsatsomeros. Sign patterns that require positive eigenvalues. *Linear and Multilinear Algebra*, 41, 1996.

S. Lauritzen. *Graphical Models*. Oxford Statistical Science Series. Oxford University Press, 1996.

C. Moallemi and B. Van Roy. Consensus propagation. In *Advances in Neural Information Processing Systems*, 2006a.

C. Moallemi and B. Van Roy. Convergence of the min-sum message passing algorithm for quadratic optimization. Technical Report, March 2006b.

J. Mooij and H. Kappen. Sufficient conditions for convergence of loopy belief propagation. In *Proc. Uncertainty in Artificial Intelligence*, 2005.

J. Pearl. *Probabilistic inference in intelligent systems*. Morgan Kaufmann, 1988.

K. Plarre and P. Kumar. Extended message passing algorithm for inference in loopy Gaussian graphical models. In *Ad Hoc Networks*, 2004.

W. Rudin. *Principles of Mathematical Analysis*. McGraw Hill, 3rd edition, 1976.

P. Rusmevichientong and B. Van Roy. An analysis of belief propagation on the turbo decoding graph with Gaussian densities. *IEEE Trans. Information Theory*, 48(2), 2001.

T. Speed and H. Kiiveri. Gaussian Markov distributions over finite graphs. *The Annals of Statistics*, 14(1), 1986.

E. Sudderth, M. Wainwright, and A. Willsky. Embedded trees: Estimation of Gaussian processes on graphs with cycles. *IEEE Trans. Signal Processing*, 52(11), November 2004.

S. Tatikonda and M. Jordan. Loopy belief propagation and Gibbs measures. In *Uncertainty in Artificial Intelligence*, 2002.

R. S. Varga. *Matrix iterative analysis*. Springer-Verlag, 2000.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. Information Theory*, 49(5), 2003.

Y. Weiss and W. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13, 2001.

J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring AI in the new millennium*, 2003.

Journal of Machine Learning Research 7 (2006) 2065-2086 Submitted 4/05; Revised 4/06; Published 10/06

Distance Patterns in Structural Similarity

Thomas Kämpke KAEMPKE@FAW-NEU-ULM.DE
Forschungsinstitut für Anwendungsorientierte Wissensverarbeitung/n FAW/n
Lise-Meitner-Str. 9
89081 Ulm, Germany

Editor: Peter Dayan

Abstract

Similarity of edge labeled graphs is considered in the sense of minimum squared distance between corresponding values. Vertex correspondences are established by isomorphisms if both graphs are of equal size and by subisomorphisms if one graph has fewer vertices than the other. Best fit isomorphisms and subisomorphisms amount to solutions of quadratic assignment problems and are computed exactly as well as approximately by minimum cost flow, linear assignment relaxations and related graph algorithms.

Keywords: assignment problem, best approximation, branch and bound, inexact graph matching, model data base

1. Introduction

Structural similarity is involved in a variety of pattern recognition problems when considered from an abstract perspective. The abstraction refers to measurements and observations whose specifics are ignored. One class of such problems is encountered in image processing, where a set of features or objects with topological interrelations is detected in several scenes. Whenever these are presumed to be similar according to position, proximity or else, the degree of similarity is of interest.

Structures are represented throughout by labeled graphs such as image graphs. In image graphs, vertices represent image edges, corners or regions of interest such as regions of constant intensity or homogenous texture. Graph edges represent relations such as neighborhoods or concept hierarchies. Edge labels represent distances, degrees of association or else.

Structural similarity is considered as similarity between two labeled graphs. Typical roles of the two graphs are that of a model graph from a model data base or a prototype data base and that of an instance graph representing an 'as is' structure which is encountered 'at run time'. The issue is then to determine the similarity between prototype and instance.

Similarity will be formulated as a best approximation problem. This involves minimization of squared distances which results in a quadratic assignment problem. The problem is approached by several algorithmic concepts including network algorithms with emphasis on linear assignment relaxations. Also, cost minimal flows of given strength will play a major role. The focus is on approximate algorithms for best graph approximation since the exact problem is NP-hard.

Besides image processing, structural similarity is encountered, for example, in document analysis and molecular graph search. However, the objective of this work is not to consider one particular real or potential application. Instead, common problem formulations and algorithms are presented.

The remainder of this work is organized as follows. Section 2 introduces the best approximation problem for edge-labeled graphs and reviews related work. Polynomial time approximation algorithms are stated in Section 3. Since the best graph approximation problem contains subgraph isomorphism as a special case, no exact algorithm can be expected to run in polynomial worst case time. The approximations are consequently based on linear assignment problems since the original problem is a quadratic assignment problem. Approximations have interesting side features such as being suited for grid computations. Section 4 contains two sketches of approaches for exact algorithms for the best graph approximation problem. One is based on flows, the other on branch and bound.

Unless otherwise stated, all graphs considered are undirected which means that edges have no preferred directions. Moreover, the graphs are simple which means that there is at most one edge between any two vertices and there are no loops so that no edge begins and ends in the same vertex. The edge labels themselves must allow to be subtractable from each other but are otherwise unconstrained. In particular, the edge labels themselves do not have to reflect any notion of similarity.

The 2-norm or Euclidean norm of any vector $z = (z_1, \ldots, z_n)$ of real numbers z_i is denoted by $||z|| = ||z||_2 = \sqrt{z_1^2 + \ldots + z_n^2}$. The number of elements of a finite set A is denoted by $|A|$.

2. Problem and Related Work

A distance pattern is understood to be an undirected graph with edge labels. The graph vertices denote objects or states and the edge labels denote distances, transition times etc. Though it may take quite some effort to generate these graphs in applications, this effort is ignored here and two such graphs are assumed to be given.

2.1 Problem Formulation

The best approximation of a labeled graph by another labeled graph is defined by a subisomorphism of the vertex set of the first graph to the vertex set of the second graph. Thereby edge labels of the first graph are approximated by corresponding edge labels of the second graph as minimum sum of squared differences.

Formally, two undirected graphs with edge labelings are given by $G_1 = (V_1, E_1)$ with $l_1 : E_1 \to \mathbb{R}$ and $G_2 = (V_2, E_2)$ with $l_2 : E_2 \to \mathbb{R}$. The first graph has $n = |V_1|$ vertices and the second graph has $m = |V_2|$ vertices with $n \leq m$. The best approximation of G_1 by G_2 is defined via an optimal approximating subisomorphism

$$\varphi^0 = argmin_{\varphi: V_1 \to V_2,\, \varphi \text{ invertible}} \sqrt{\sum_{\{v_i, v_j\} \in E_1} \Big(l_1(v_i, v_j) - l_2(\varphi(v_i), \varphi(v_j)) \Big)^2}.$$

The best approximation is given by the image of the first vertex set $\varphi^0(V_1)$ so that $\varphi^0(V_1) \subseteq V_2$. The minimum value is called the distance of the best approximation.

In order to be well defined, the problem entails a technical condition that, whenever two vertices of the first graph are joined by an edge, the images of the two vertices must be joined by an edge in the second graph. Without further restrictions to the subisomorphisms this implies that the second graph must be complete.

The objective of best approximation can be considered as Frobenius distance when both graphs are complete. The edge labelings then denote distance matrices D_1, D_2 with entries

$$D_1(v_i,v_j) = \begin{cases} l_1(v_i,v_j), & \text{for } v_i \neq v_j \\ 0, & \text{for } v_i = v_j \end{cases} \quad D_2(w_i,w_j) = \begin{cases} l_2(w_i,w_j), & \text{for } w_i \neq w_j \\ 0, & \text{for } w_i = w_j. \end{cases}$$

The best approximation objective can then be written as follows since counting over all edges amounts to counting twice over all vertex pairs. Pairs of identical vertices are negligible because they contribute value zero.

$$\begin{aligned} \sum_{\{v_i,v_j\}\in E_1} \Big(l_1(v_i,v_j) - l_2(\varphi(v_i),\varphi(v_j))\Big)^2 &= \frac{1}{2} \sum_{v_i,v_j\in V_1} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2 \\ &= \frac{1}{2}\,||(D_1(\cdot,\cdot)) - (D_2(\varphi(\cdot),\varphi(\cdot)))||_F^2. \end{aligned}$$

The Frobenius norm of any matrix is the square root of the sum of all squared entries (Golub and van Loan, 1985).

Distance matrices allow to consider the best approximation problem also for graphs which are not complete. Whenever one of the two given graphs is not complete, all missing edges are inserted. The complete graph is then labeled by the shortest path distances according to the original edge labeling. Original edge labels are preserved when these satisfy the triangle inequality. Original edge labels may be overwritten when these do not satisfy the triangle inequality.

The celebrated graph isomorphism problem is contained in the best approximation problem as the following special case. Suppose H_1 and H_2 are two unlabeled graphs with same number of vertices and arbitrary edge sets. Each graph is extended to the complete graph on its vertex set and receives the edge labels

$$l_i(e) := \begin{cases} 1, & \text{if edge } e \text{ belongs to original graph } H_i \\ 0, & \text{if edge } e \text{ does not belong to original graph } H_i, \end{cases}$$

$i = 1,2$. These graphs are denoted G_1 and G_2 respectively. The original graphs are isomorphic if and only if the best approximation of G_1 by G_2 and the best approximation of G_2 by G_1 both have distance zero.

The most trivial case of the best approximation problem is given for the smaller graph being the smallest possible. This is a two vertex graph with one edge only. The single edge graph is best approximated by that edge from the larger graph whose label comes closest to the label of the single-edge graph. A non-trivial example of the best approximation problem is given in Figures 1 and 2.

Best graph approximation can be considered as search for a minimum weight clique of given size in a suitably defined graph, the association graph or correspondence graph. This relation is such that the given clique size equals the size of the smaller graph and that no larger cliques exist in the association graph. Mnemonically, best graph approximation can thus be remembered as search for a minimum weight clique of maximum size.

The association graph of two graphs is defined as their product. Each vertex of one graph is paired with each vertex of the other graph and each such pair is identified with a vertex of the association graph. Two vertices of the association graph are joined by an edge if the two vertices stem from four distinct vertices of the original graphs. The construction is illustrated in Figure 3. The

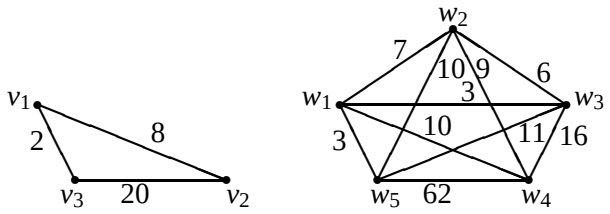


Figure 1: The left three vertex graph is best approximated by the triangle with edge labels $3, 10, 16$ in the larger graph. The subisomorphism is $\varphi(v_1) = w_1$, $\varphi(v_2) = w_4$ and $\varphi(v_3) = w_3$ with the two vertices w_2 and w_5 being unattained. The graph isomorphism is given explicitly in the next figure.

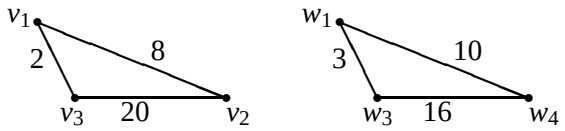


Figure 2: Original graph (left) and best approximating isomorphic substructure (right) for the situation of Figure 1. The squared distance between the two graphs is $(2-3)^2 + (8-10)^2 + (20-16)^2 = 21$.

cost of the edge between the vertices (v_i, w_j) and (v_k, w_l) is set equal to $(D_1(v_i, v_k) - D_2(w_j, w_l))^2$. The meaning of selecting any vertex of the form (v_i, w_j) is that the original vertex v_i is mapped to the original vertex w_j by a subisomorphism. These "associations" motivate the name association graph and the cost of a clique, which equals the cost sum over all edges between selected vertices, is the objective of graph approximation.

An alternative view of best graph approximation can be obtained from quadratic assignment problems such as the following

$$\begin{aligned} &\min_x x^T C x \\ &\text{such that } \sum_{i=1}^{n} x_{ij} \leq 1 \quad \forall\, j = 1, \ldots, m \\ &\qquad \sum_{j=1}^{m} x_{ij} = 1 \quad \forall\, i = 1, \ldots, n \\ &\qquad x_{ij} \in \{0, 1\} \quad \forall\, i, j. \end{aligned}$$

The binary variable x_{ij} attaining value one means that vertex v_i is assigned to vertex w_j and that variable attaining the value zero means that this assignment is not valid. The vector x has $n \cdot m$ coordinates and C is an $n \cdot m \times n \cdot m$ matrix denoting the cost incurred by pairwise assignments; the assignments $v_i \mapsto w_j$ and $v_k \mapsto w_l$ entail the cost $(D_1(v_i, v_k) - D_2(w_j, w_l))^2$. The cost matrix is

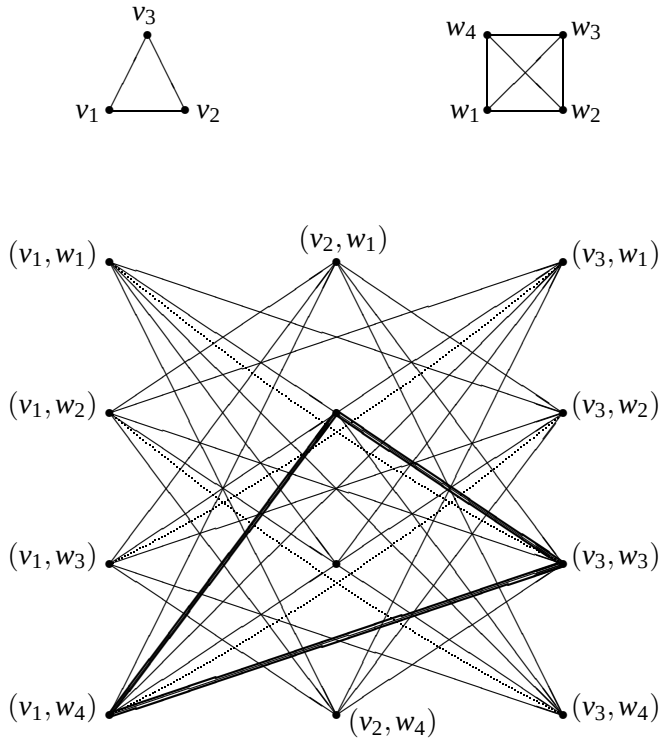


Figure 3: A "small" graph and a "large" graph (top) and their association graph (bottom). The indicated clique of the vertices (v_1,w_4), (v_2,w_2) and (v_3,w_3) denotes the subisomorphism $\varphi(v_1) = w_4$, $\varphi(v_2) = w_2$ and $\varphi(v_3) = w_3$.

computed as

$$C = \begin{pmatrix} (D_1(v_1,v_1) - D_2(\cdot,\cdot))^2 & \dots & (D_1(v_1,v_n) - D_2(\cdot,\cdot))^2 \\ \vdots & \vdots & \vdots \\ (D_1(v_n,v_1) - D_2(\cdot,\cdot))^2 & \dots & (D_1(v_n,v_n) - D_2(\cdot,\cdot))^2 \end{pmatrix}$$

where $D_2(\cdot,\cdot)$ is the $m \times m$ matrix of all distance values for the second graph and $(c-M)^2$, the square of a constant c minus a matrix M, is understood as a matrix of the size of M with each element denoting the squared distance from the constant so that $(c-M)^2 = ((c-m_{ab})^2)_{ab}$. The size of the cost matrix is unfortunately large as it already is a 15×15 matrix for the small graphs from Figure 1.

2.2 Related Work

The subisomorphism problem which is contained as special case of the best approximation problem must not be confused with that version of the SUBGRAPH ISOMORPHISM problem which is known to be NP-complete, see Garey and Johnson (1981, problem GT48). That problem does not admit edge labels and it considers the two operations of vertex removal and edge removal for the transition from the larger to the smaller graph. Even more, the LARGEST COMMON SUBGRAPH

problem also is NP-complete, see Garey and Johnson (1981, problem GT49). This problem allows edge removals only in order to find isomorphic subgraphs. Here, only vertex removals matter and edge removals are allowed only in so far as they are implied by vertex removals.

A widely used measure for so-called inexact graph matching is the edit distance between two unlabeled graphs. One graph is therefore modified by a minimum number of vertex insertions and deletions and by edge insertions and deletions. The concept in general as well as a particular focus on tree graphs is given in Wang et al. (1998). A simplification of the edit distance does not refer to the graphs themselves and isomorphism between them but to their degree histograms which are to be made equal by a minimum number of changes (Papadopoulos and Manolopoulos, 1999). The edit distance for subisomorphisms of labeled graphs is considered by Messmer and Bunke (1998a).

Subisomorphism with vertices and edges both carrying labels is considered by Hlaoui and Wang (2002). Conflicts of tentatively assigning several vertices of the smaller graph to one vertex of the larger graph are resolved in a hierarchical manner. The method is reported to be well suited for small graphs. Best graph approximation in terms of matching problems is considered by Gold and Rangarajan (1996). There, the problem is extended to a sequence of continuous surrogate problems. This leads to an iterative, matrix-based solution scheme which is controlled by a continuous parameter that intends to drive continuous relaxations to a discrete vertex assignment. While leaving slight uncertainties about the control of this parameter and while using a linear instead of a quadratic distance between edge labels, the method makes, like the approaches given below, use of the assignment problem. A quite different, probabilistic approach which makes use of potential functions is given in Caetano et al. (2005).

Best graph approximation can be considered as ”dual” to joint edge and label construction. This construction was developed for trees by Desper and Vingron (2002). Only distance information is required in order to build one tree with edge weights so that the given distances are approximated by path lengths between leaves of the tree. The construction is formulated as a least square approximation problem so that solution methods have a strong algebraic component.

Matching techniques for graphs whose vertices denote positions in space have been developed from the analogy of physical elasticity, compare Wiskott et al. (1997) and Wiskott and Malsburg (2002). In addition, the elasticity idea supports the generation of the model graph from examples. These physical methods are complemented by probabilistic methods for unlabeled graph subisomorphisms by Bengoetxea (2002).

Motivated by graphs that describe the structure of SQL data bases, an interactive fixpoint algorithm for similarity computing has been proposed by Melnik et al. (2002). The method is based on the assumption that adjacent vertices are more similar than non-adjacent vertices. The quality of the matching result is measured by the number of human adjustment steps that are eventually needed. For a similar data base purpose, case-based reasoning, similarity of graphs with character-string labels has been considered (Champin and Solnon, 2003). Similarity is measured by weighted counts of identical labels with vertex correspondence being generated by a greedy algorithm. This algorithm maximizes the similarity score in each iteration.

Best graph approximation relates given lists of numbers by vertex-edge incidences. When these are dropped, that is, when numbers are given as mere list entries of one list and when the numbers are viewed as Euclidean distances on the real line, a complete line graph may be searched for such that the given numbers form a coherent distance labeling. This is the NP-complete PARTIAL DIGEST problem from genomic mapping, see Skiena and Sundaram (1994). Whenever the best graph approximation problem refers to graphs with Euclidean distances on the line and whenever

the smaller graph is known to be contained in the larger graph, methods for PARTIAL DIGEST can help to identify the actual subgraph isomorphism.

Subgraph isomorphism for unlabeled graphs has been studied in Messmer and Bunke (1998b) motivated by symbol recognition problems. A polynomial time algorithm is given which requires preprocessing and exponential space in the worst case. Graph similarity has even been investigated for machine learning (Pope and Lowe, 1996). A quite sketchy outline of graph search methods over molecular graphs is presented in Shasha et al. (2002), while the perspective of distance methods for molecular similarity and superstructure retrieval is given in Kämpke (2004).

3. Approximate Algorithms

The best approximation problem obviously is finite and, thus, can, in principle, be solved by enumeration over all $\binom{m}{n}n!$ selections for admissible functions φ. Approximate algorithms of different types as well as a strategy for exact algorithms are given in the sequel. The present approximations mainly focus on linear assignment problems since the original problem is a quadratic assignment problem.

To illustrate the intuitive aim of best graph approximation, the squared approximation distance is rewritten for the special case of equally sized graphs. The best isomorphism then is a solution of the maximization problem

$$\max_{\varphi:V_1\to V_2,\ \varphi \text{ invertible}} \sum_{\{v_i,v_j\}\in E_1} D_1(v_i,v_j)\cdot D_2(\varphi(v_i),\varphi(v_j)).$$

The sum can be considered as an inner product over edges. When all edges of the first graph are sorted increasingly then the maximization is obtained by an isomorphism that maintains monotonicity "as far as possible". This is motivated by the well known inner product maximization $\sum_{i=1}^{N} a_i b_{\pi(i)}$ problem over all permutations π. The solution is a permutation with $b_{\pi(1)} \leq \ldots \leq b_{\pi(N)}$ whenever all coordinates of the first vector are sorted as $a_1 \leq \ldots \leq a_N$, compare with Hardy et al. (1948). A monotonicity preserving isomorphism does generally not exist for the edge labels.

3.1 Distance Lists

A heuristic procedure for best subgraph isomorphism can be devised on local, that is, vertex-oriented decisions. These are based on distance lists. The distance list of a vertex contains the labels of all edges that are incident with the vertex. Any distance list can also be considered as a vector. The distance list of a vertex v is denoted by $distlist(v)$. A distance list in a complete graph with edge labels $D(\cdot,\cdot)$ is given by $distlist(v) = (D(v,v_i))_{v_i\in V-\{v\}}$. For the ease of comparability, all vertex lists are sorted increasingly. Distance lists generalize vertex degrees since they count the "ones" when unlabeled graphs receive the binary edge labeling as given above for the embedding of the graph isomorphism problem.

The distance lists for the three-vertex graph from Figure 1 are given by $distlist(v_1) = (2,8)$, $distlist(v_2) = (8,20)$ and $distlist(v_3) = (2,20)$. The five-vertex graph of the same figure has the distance lists $distlist(w_1) = (3,3,7,10)$, $distlist(w_2) = (6,7,7,9)$, $distlist(w_3) = (3,6,8,16)$, $distlist(w_4) = (9,10,16,62)$ and $distlist(w_5) = (3,7,8,62)$.

Viewing distance lists as vectors allows to consider Euclidean distances between distance lists. This only requires standard notions for vector distances as long as distance lists have the same number of coordinates. The lists being sorted makes these differences meaningful. Whenever two

distance lists have different numbers of coordinates, a proper selection is made from the larger distance list. In the foregoing example, selections of two out of the four coordinates from the distance lists of the five-vertex graph are made.

The approximation of a distance list by a selection from a larger distance list can be formulated as a weighted or cost minimal assignment problem. Therefore, two distance lists $distlist(v) = (x_1,\ldots,x_{n-1})$ and $distlist(w) = (y_1,\ldots,y_{m-1})$, $n \leq m$, are endowed with a complete bipartite graph. Each coordinate receives one vertex and each coordinate of one distance list is connected by an edge to each coordinate of the other distance list. No two coordinates of the same list are connected. The edge connecting coordinates x_i and y_j is labeled by the squared difference of the list entries $(x_i - y_j)^2$, compare with Figure 4.

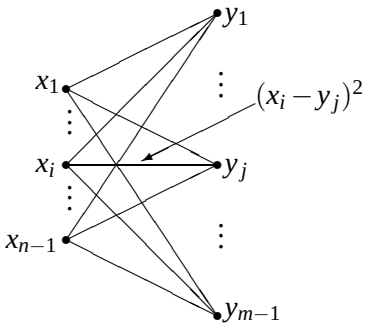


Figure 4: Complete bipartite graph for two distance lists. The vertices correspond to the coordinates of the distance lists rather than to the vertices of the original graphs. Only one of the $(n-1)\cdot(m-1)$ edge labels is sketched.

Any approximation of the first distance list by the second distance list amounts to a selection of $n-1$ distinct coordinates or indices from the second list. The approximation objective can be formulated as a linear function of the edge labels

$$\min_{1\leq j_1<\ldots<j_{n-1}\leq m-1} \sum_{i=1}^{n-1} (x_i - y_{j_i})^2.$$

Alternatively, the best approximation problem for distance lists can be formulated as cost minimal perfect matching problem and as a cost minimal integral flow problem with flow value set to level $n-1$, compare with Section 4.

The best approximation of a distance list $distlist(v)$ by the distance list $distlist(w)$ is denoted as the projection $pr(distlist(w),\ distlist(v))$. The squared approximation error equals $DL(v,w) = ||pr(distlist(w),\ distlist(v)) - distlist(v)||^2$. Samples of distance lists, their best approximations and approximation errors are given in the following table whose data refer to Figure 1.

	$distlist(v_i)$	$distlist(w_j)$	$pr(distlist(w_j), distlist(v_i))$	$DL(v_i, w_j)$
$i=1$	$(2,8)$	$(3,3,7,10)$	$(3,7)$	$(2-3)^2+(8-7)^2=2$
	$(2,8)$	$(6,7,9,10)$	$(6,7)$ or $(6,9)$	$(2-6)^2+(8-7)^2=17$
	$(2,8)$	$(3,6,11,16)$	$(3,6)$	$(2-3)^2+(8-6)^2=5$
	$(2,8)$	$(9,10,16,62)$	$(9,10)$	$(2-9)^2+(8-10)^2=53$
	$(2,8)$	$(3,10,11,62)$	$(3,10)$	$(2-3)^2+(8-10)^2=5$
$i=2$	$(8,20)$	$(3,3,7,10)$	$(7,10)$	$(8-7)^2+(20-10)^2=101$
	$(8,20)$	$(6,7,9,10)$	$(7,10)$ or $(9,10)$	$(8-7)^2+(20-10)^2=101$
	$(8,20)$	$(3,6,11,16)$	$(6,16)$	$(8-6)^2+(20-16)^2=20$
	$(8,20)$	$(9,10,16,62)$	$(9,16)$	$(8-9)^2+(20-16)^2=17$
	$(8,20)$	$(3,10,11,62)$	$(10,11)$	$(8-10)^2+(20-11)^2=85$
$i=3$	$(2,20)$	$(3,3,7,10)$	$(3,10)$	$(2-3)^2+(20-10)^2=101$
	$(2,20)$	$(6,7,9,10)$	$(6,10)$	$(2-6)^2+(20-10)^2=116$
	$(2,20)$	$(3,6,10,16)$	$(3,16)$	$(2-3)^2+(20-16)^2=17$
	$(2,20)$	$(9,10,16,62)$	$(9,16)$	$(2-9)^2+(20-16)^2=65$
	$(2,20)$	$(3,10,11,62)$	$(3,10)$	$(2-3)^2+(20-10)^2=101$

3.2 Best Approximations by Distance Lists

The idea of cost minimal assignments can be carried over from distance lists of single vertices to the whole graph. This results in an efficient heuristic algorithm for best graph approximation. Again, the approximation problem is formulated as cost minimal assignment problem over a complete bipartite graph. One set of vertices corresponds to the smaller graph and the other to the larger graph. The edges are labeled by the errors of best distance list approximations. The general situation is sketched in Figure 5.

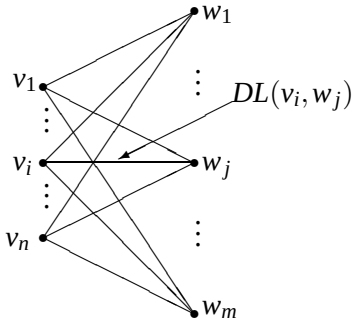


Figure 5: Complete bipartite graph for a heuristic solution of best graph approximation. The edge labels refer to best approximations of distance lists.

A greedy heuristic for best graph approximation can now be based on iterative decisions according to minimum distance list errors.

ApproxDistList

1. Input complete labeled graphs G_1, G_2 with $|V_1| \leq |V_2|$.
   Initialization. Computation of distance list errors $DL(v,w)$ for all $v \in V_1$, $w \in V_2$. $A = V$, $B = W$.
2. While $A \neq \emptyset$ do
   (a) Computation of $W(v) = argmin_{w \in B} DL(v,w)$ and $level(v,B) = DL(v,W(v))$ and for all $v \in A$.
   (b) Selection of $v_0 = argmin_{v \in A} level(v,B)$.
   (c) $\varphi(v_0) = w(v_0)$ with $w(v_0) \in W(v_0)$.
   (d) $A = A - \{v_0\}$.
   (e) $B = B - \{\varphi(v_0)\}$.
3. Output subisomorphism $\varphi(\cdot)$ on V_1.

The level computations in step 2(a) determine the best fit decision that can be made without taking back prior decisions. The sets $W(v)$ indicate all vertices from the second graph that attain the minimum. Ties for selections in step 2(b) as well as for selections from the set $W(v_0)$ in step 2(c) are broken arbitrarily. While the sets A and B of unassigned vertices decrease along the iterations of the algorithm, the distance lists to consider become fewer but not smaller. Thus, the edge labels $DL(v,w)$ do not have to be updated along the iterations.

The greedy procedure applied to the graphs of Figure 1 can be traced with the data from the table of Section 3.1. The resulting subisomorphism is given by selecting the minima for each vertex from the smaller graph. This is exactly the solution indicated by Figure 2.

The foregoing algorithm need not solve the cost minimum assignment problem exactly. An optimal solution of this problem (which still need not lead to the best graph approximation since the assignment problem is only an approximative encoding for best graph approximation problem) can be found by any weighted assignment algorithm for bipartite graphs. These algorithms are typically based on transformations to cost minimum flow problems. Therefore, all vertices of the smaller graph are connected to an extra source vertex and all vertices of the larger graph are connected to an extra sink vertex. All the extra edges receive a unit capacity on the flow. All edges become oriented edges or arcs as indicated in the flow network in Figure 6.

Among the cost minimal flows of strength n there is one with all integer values. This flow amounts to a cost minimal assignment of all vertices from the smaller graph. The out of kilter algorithm allows to compute the desired cost minimum flow, see Ahuja et al. (1993).

A different solution for cost minimal assignment problems can be obtained from straightforward transformations to cost minimal perfect matching problems by introducing additional vertices for the smaller graph. Both sets of the vertex partition then have the same size. All dummy vertices for the smaller graph are connected to all vertices from the larger graph by edges with zero cost. A matching is a set of edges such that any two edges do not have a common vertex. A matching is perfect if each vertex of the graph is covered by an edge.

A polynomial time algorithm for cost minimal perfect matchings can be based on augmenting paths that are constructed by shortest paths. Implementations thereof are available in the LEDA system, see Mehlhorn and Näher (2000, Chapter 7). More recent scaling algorithms are given in Ahuja et al. (1993).

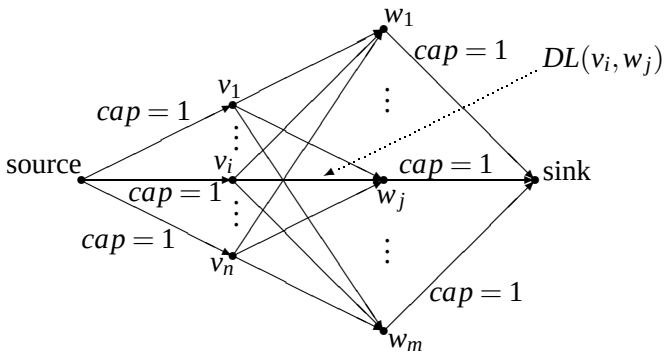


Figure 6: Flow network for a distance label heuristic. The edges that are incident either to the source or the sink carry capacities but no cost coefficients while the edges with cost coefficients do not carry capacities.

3.3 Direct Methods

Direct methods for graph approximation will use the original edge labels and the unaltered best approximation errors for all fitting assessments.

3.3.1 SEQUENTIAL ASSIGNMENTS

A subisomorphism can be constructed by sequentially assigning vertices from the smaller graph to the larger graph so that the sum of squared label distances over all new edge pairs is minimal. The first assignment may stem from two best matching edges. No assignment is ever revised by the following procedure.

SeqAssign

1. Input complete labeled graphs G_1, G_2 with $|V_1| \leq |V_2|$.
   Initialization. Computation of $(e_0, f_0) = argmin_{e \in E_1, f \in E_2} (D_1(e) - D_2(f))^2$.
   Selection of one vertex v_1 of the two vertices incident with e_0.
   Selection of one vertex w_1 of the two vertices incident with f_0.
   $\varphi(v_1) = w_1$.
   Labeling all other vertices from V_1 by $v_2, \ldots, v_n$.
   $B = V_2 - \{\varphi(v_1)\}$.

2. For $i = 2, \ldots, n$ do

   (a) Computation of $w_0 = argmin_{w \in B} \sum_{j=1}^{i-1} (D_1(v_j, v_i) - D_2(\varphi(v_j), w)^2$.

   (b) $\varphi(v_i) = w_0$.

   (c) $B = B - \{\varphi(v_i)\}$.

3. Output subisomorphism $\varphi(\cdot)$ on V_1.

3.3.2 IMPROVEMENTS

Whenever a subisomorphism is not optimal or not known to be optimal, improvements can be aimed at by swapping two vertex assignments or by swapping an assigned with an unassigned vertex. Swaps of both types can easily be evaluated.

For notational ease the vertices are numbered such that $\varphi(v_k) = w_k$, $k = 1,\ldots,n$, for some given subisomorphism $\varphi : V_1 \to V_2$. First, two vertices v_i, v_j, $1 \leq i \neq j \leq n$ are considered for swapping their assignments while all other assignments are preserved.

$$\varphi'(v_k) = \begin{cases} w_j & \text{if } k = i \\ w_i & \text{if } k = j \\ w_k & \text{if } k \neq i, j. \end{cases}$$

The new subisomorphism leads to a smaller approximation error if and only if

$$\sum_{k=1,k\neq i,j}^{n} \Big(l_1(v_i,v_k) - l_1(v_j,v_k)\Big) \cdot \Big(l_2(w_j,w_k) - l_2(w_i,w_k)\Big) > 0.$$

This condition being true is denoted by $\mathsf{Imp(i,j)} = \mathsf{true}$. Second, one vertex v_i, $1 \leq i \leq n$ is considered for changing its assignment to an unassigned vertex $w_0 \in V_2 - \varphi(V_1)$, that is, the present subisomorphism is compared to the new subisomorphism

$$\varphi'(v_k) = \begin{cases} w_0 & \text{if } k = i \\ w_k & \text{if } k \neq i. \end{cases}$$

The new subisomorphism leads to a smaller approximation error if and only if

$$\sum_{k=1,k\neq i}^{n} l_2^2(w_i,w_k) - l_2^2(w_0,w_k) > 2 \sum_{k=1,k\neq i}^{n} l_1(v_i,v_k) \cdot \Big(l_2(w_i,w_k) - l_2(w_0,w_k)\Big).$$

This condition being true is denoted by $\mathsf{Imp(i,}w_0\mathsf{)} = \mathsf{true}$. The improvement conditions results in the following procedure.

Imp

1. Input complete labeled graphs G_1, G_2.
   Subisomorphism $\varphi : V_1 \to V_2$ with $\varphi(v_k) = w_k$, $k = 1,\ldots,n$.
2. While $\mathsf{Imp(i,j)} = \mathsf{true}$ or $\mathsf{Imp(i,}w_0\mathsf{)} = \mathsf{true}$ for some w_0 do
   (a) If $\mathsf{Imp(i,j)} = \mathsf{true}$ then $\varphi(v_i) = w_j$ and $\varphi(v_j) = w_i$
   else $\varphi(v_i) = w_0$
   (b) Vertex relabeling such that $\varphi(v_k) = w_k$, $k = 1,\ldots,n$.
3. Output swap-improved subisomorphism $\varphi : V_1 \to V_2$.

Swaps for triples, quadruples etc. can be considered instead of pairwise swaps. Though the complexity of evaluating such swaps increases only little, the number of swap candidates increases by one order of m for each size increase of the swap candidates.

3.4 Relaxation Method

The previous methods can all be considered as primally feasible which means that all assignments actually are subisomorphisms though not necessarily optimal. The subisomorphism constraint may tentatively be relaxed in analogy to so-called dual optimization techniques. Starting from some promising structure, a sequence of changes will be made that eventually attain feasibility and that tend to incur as little additional cost as possible per step.

3.4.1 INITIAL STRUCTURE

A promising initial structure is constructible by enumerating the first vertex set. Each vertex from that set is paired with all vertices from the second vertex set which amounts to considering their common vertices in the association graph. The label sum of all edges which emanate from each of these common vertices is minimized under the choice of the second vertex. This can be expressed as the following nested minimization:

$$\mu(i) = argmin_{j=1,\ldots,m} \sum_{k=1,k\neq i}^{n} \min_{l=1,\ldots,m} \Big(D_1(v_i,v_k) - D_2(w_j,w_l) \Big)^2 .$$

This minimization implies a function from the first vertex set into the second vertex set by $v_i \mapsto w_{\mu(i)}$. The inner minimizations are independent of each other and, thus, may lead to overassignments which means that the same index l is attained as minimum for different outer indices. The analogue is true for the outer minimization. The presence of overassignments implies that the overall function is not a subisomorphism. However, the independence of the minimizations makes them easy to compute and the resulting cost value provides a lower bound for the cost of best graph approximation. The edges for summation in one minimization of the initial relaxation are sketched in Figure 7. It is worth noticing that the minimization for the initial structure may lead to even

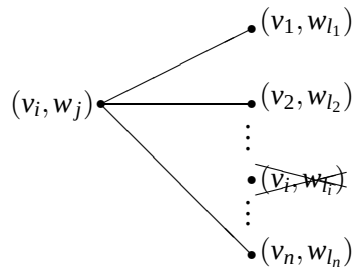


Figure 7: Vertices and edges of the association graph that are considered while computing the value $\mu(i)$ for the initial structure of the relaxation method.

smaller objective values than the distance lists.

3.4.2 IMPROVEMENT

After initialization, the relaxation method proceeds by iteratively selecting an overassigned vertex and redirecting or backtracking at least one assignment to a yet unassigned vertex. Several vertex assignments may be altered in each iteration. Each iteration is organized by computing a wave front of node potentials that emanates from an overassigned vertex.

The edges for the wave front computations are directed. All edges of the current structure are oriented as backward edges from the second vertex set to the first vertex set and all other edges are oriented as forward edges from the first to the second vertex set, see Figure 8. Not all edge labels are initially known in quite a contrast to the ordinary dual method for assignment problems. Actually, node potentials will be computed according to edge transitions rather than by explicitly given edge labels.

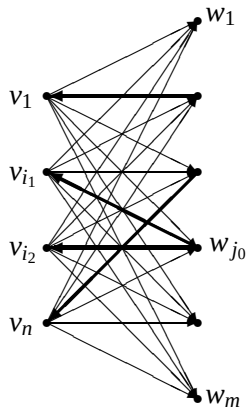


Figure 8: Forward edges (thin) and four backward edges (bold) for the wave front computations to resolve the double assignment of vertex w_{j_0}. The wave front begins in that vertex and ends in a suitable vertex from the second set from which no edge leads back into the first set.

The relaxation method assigns tentative and permanent labels to graph nodes in analogy to the Dijkstra algorithm. The labels are potentials which equal the cost of functions from the first vertex set or a subset thereof into the second vertex set. Such functions need not be one-to-one. Formally, the potential of a function φ is computable as $cost(\varphi) = \sum_{v \in dom(\varphi)} \sum_{v' \in dom(\varphi) - \{v\}} (D_1(v,v') - D_2(\varphi(v), \varphi(v')))^2$, where $dom(\varphi)$ denotes the domain of function φ. Whenever a function is altered by deleting an assignment like $v_4 \mapsto w_3$ it is denoted by $\varphi - (v_4 \mapsto w_3)$, when then the assignment $v_4 \mapsto w_8$ is inserted, the function is denoted by $\varphi - (v_4 \mapsto w_3) + (v_4 \mapsto w_8)$ etc. Backward edges amount to deleting assignments and forward edges amount to inserting assignments.

NoPo

1. Input Structure φ, overassigned vertex $w_0 \in V_2$.
   Initialization $L = V_1 \cup V_2 - \{w_0\}$, $m(l) = \infty \; \forall l \in L$, and $m(w_0) = cost(\varphi)$.
2. While $(V_2 - \varphi(V_1) \subseteq L)$ do:
   (a) Selection of $u = argmin_{l \in L} m(l)$.
   (b) $L = L - \{u\}$.
   (c) $\forall z \in L \cap S(u)$ do:
      i. $\varphi_z = \varphi_u + (u \mapsto z)$ if $z \in V_2$ and $u \in V_1$
         $\varphi_z = \varphi_u - (u \mapsto z)$ if $z \in V_1$ and $u \in V_2$.

ii. Computation of $cost(\varphi_z)$.
iii. If $cost(\varphi_z) < m(z)$ then $m(z) = cost(\varphi_z)$.

3. Termination. Output φ_z for that $z \in V_2 - \varphi(V_1)$ which received its permanent label most recently.

The set $S(u)$ denotes the set of all immediate successors of vertex u which is the set of all vertices to which an edge points from vertex u. The list L contains all vertices that are tentatively labeled. The graph vertices which are not contained in the list are permanently labeled. The algorithm terminates as soon as the first yet unassigned vertex from the second set receives a permanent label.

It may occur during wave front propagation that an already assigned vertex from the second set is permanently labeled. This means that an assignment of the input function φ is revised. The node potential algorithm terminates with exactly one additional vertex assignment from the second graph. The algorithm is applied repeatedly until all overassignments are eliminated.

The improvement algorithm **Imp** from Section 3.3.2 can be obtained from the mode potential algorithm **NoPo** by starting at a vertex that is attained exactly once (with all vertices of the second set being attained at most once). The search for cost reductions proceeds along wave fronts of length two or four and by allowing the wave front to return to its origin.

3.5 Grid Computing Methods

The recently celebrated framework of grid computing is based on the idea of a system which coordinates distributed resources using standard, open, general purpose protocols and interfaces to deliver nontrivial qualities of services (Foster and Kesselman, 2004). Though the distribution of a computational problem into subproblems is not an inherent feature of grid computing in general, it is here considered as exactly that. The breakdown of a computational problem into subproblems that amend to partial computations without any communication between them is here called grid distribution. Grids with several thousand computing nodes have already become feasible.

The lack of any communication between computations means that neither intermediate results nor data are shared. Whenever common data are required, they are physically copied and stored separately before computations begin in order to avoid any access collision. The avoidance of intermediate result communication is a trivial concept. But this makes distributed computations feasible from a practical perspective. Multiple execution of certain operations is the price to be paid. The computational subproblems are generated, distributed and possibly queued by a master. The master also collects the individual computing results and aggregates them to the final computing result.

Best graph approximation lends to grid computing in a straightforward manner. To this end, the strategy of pivoting is here proposed for grid distribution. The idea is that of selecting a vertex from the larger graph as pivot element. This means that best graph approximation is tentatively reduced to only those subisomorphisms which attain that vertex and the optimal of such pivot-subisomorphisms is computed exactly or approximately. Eventually, all vertices of the second graph are chosen as pivot elements and the pivoting-subisomorphism with smallest objective value is reported as the best one. Heuristics which make use of assignment problems are particularly suited for pivoting.

Any subisomorphism which attains the pivot element $w_{j_0} \in V_2$ from some vertex $v_{i_0} \in V_1$ is denoted by $\varphi_{i_0 \mapsto j_0}$. A candidate for the best subisomorphism of this type will be computed and the

best φ_{j_0} of these over all vertices from the first graph is selected by independent computations. Then, the best φ_0 of these over all vertices from the second graph is centrally computed. Schematically this is denoted as

$$\underbrace{\varphi_{i_0 \mapsto j_0} \overset{\min_{i_0 \in \{1,\ldots,n\}}}{\longrightarrow} \varphi_{j_0}}_{grid} \quad pass\ result \longrightarrow \quad \varphi_{j_0} \underbrace{\overset{\min_{j_0 \in \{1,\ldots,m\}}}{\longrightarrow} \varphi_0}_{central}.$$

The weighted assignment problems that are solved for each of the subisomorphisms $\varphi_{i_0 \mapsto j_0}$ is sketched in Figure 9.

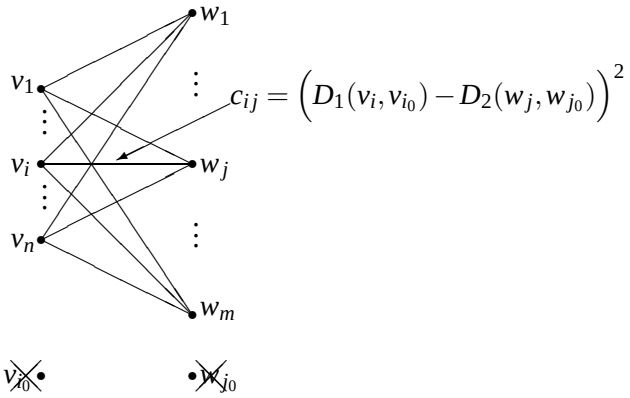


Figure 9: Complete bipartite graph for pivot vertex and preselected vertex from the first graph.

The resulting algorithm that has to be executed at the grid nodes is as follows.

GridPivot

1. Input complete labeled graphs G_1, G_2 with $n \leq m$ and $j_0 \in \{1, \ldots, m\}$.
2. Computations
   (a) Computation of $\varphi_{i_0 \mapsto j_0}$ as minimal weighted assignment for all $i_0 \in \{1, \ldots, n\}$.
   (b) Selection of φ_{j_0} with minimum objective of $\varphi_{i_0 \mapsto j_0}$ over all $i_0 \in \{1, \ldots, n\}$.
3. Output subisomorphism φ_{j_0} on V_1.

The minimization in step 2(b) adheres to the original objective of best graph approximation and no longer to the objective functions of the assignment problems from step 2(a). The computations in step 2(a) can be coupled by using the optimal assignment for one problem—for one value of i_0—as an initial assignment for the next problem—for the next value of i_0. This is feasible for primal methods as well as for dual methods such as the relaxation method, see above.

The grid distribution of the complete problem into subproblems and the selection φ_0 of the best of their results is conceptually obvious.

4. Towards Exact Methods

Since the focus is on practical algorithms, the descriptions of exact algorithmic solutions of best graph approximations is kept to an informal level. First, flows are extended and second, a branch and bound method is sketched.

4.1 Flows

Best graph approximation can be formulated as a cost minimal flow problem in loose analogy to the distance list heuristic. However, the graph is more complicated and additional constraints are necessary. These include submodular edge capacities and integrality conditions for the flow through some edges.

The main problem of transforming the best graph approximation into a flow problem is that subisomorphisms refer to vertices while the costs refer to edge pairs. The cost issue is therefore dealt by introducing a biquadratic number of network vertices that represent all possible edge pairings induced by the vertex assignments. Each pair of distinct vertices from the smaller graph may correspond to each pair of distinct vertices from the larger graph. The incurred cost is an edge label with the edge connecting a network vertex (v_i, v_j, w_k, w_l) with the sink in the flow network.

A subisomorphism in the network is specified by all considering each vertex v_i of the smaller graph and all its possible assignments by introducing the network vertices $(v_i, w_1), \ldots, (v_i, w_m)$. These are connected by edges. Since each vertex of the second graph is attained at most once by any subisomorphism, the edges into the network vertices $(v_1, w_j), \ldots, (v_n, w_j)$ have one common capacity constraint. The joint flow into all these vertices is bounded by one. The situation is depicted by Figure 10.

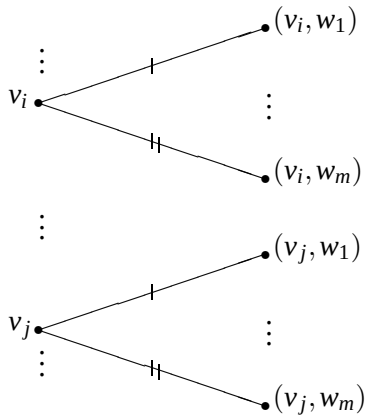


Figure 10: The m vertices $(v_i, w_1), \ldots, (v_i, w_m)$ together allow an inflow of strength one only for each of the vertices v_i. Edges with common capacity constraints are indicated with identical number of ticks.

Common edge capacities are known as submodular flow constraints, see Fujishige (1991). The flow from the source vertex to each of the graph vertices is bounded by one and the flow out of each vertex (v_i, w_j) is bounded by $\frac{1}{n-1}$. The reason for this bound is that each vertex of the smaller graph

is incident with all $n-1$ other vertices and thus $n-1$ edges of the smaller graph are incident with each vertex. The complete construction is illustrated in Figure 11 for the problem from Figure 1.

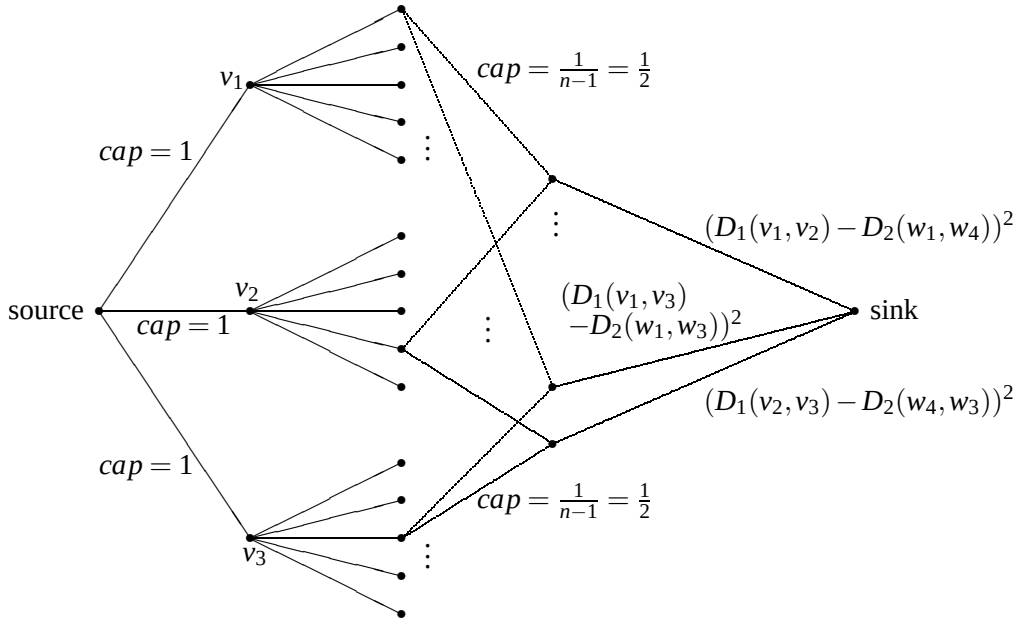


Figure 11: Part of graph for an exact solution of the best approximation problem. The submodular flow constraints as well as arc orientations are not indicated. Arcs are directed in the general direction "from left to right". The arcs for which cost labels are specified refer to the solution for the problem from Figure 1.

A best graph approximation amounts to a cost minimal flow of strength n from source to sink such that the flows along the edges between all v_i and (v_i, w_j) are integer. The other constraints then imply that the flows are binary over these edges.

4.2 Branch and Bound

The complicated structure of the foregoing flow problem motivates to organize an exact best graph approximation by branch and bound. Subisomorphisms will be built up sequentially by either pruning or refining a partial subisomorphism. A partial subisomorphism is a subisomorphism defined over a subset of the vertex set of the smaller graph. The domain of a partial subisomorphism is denoted by $A(V_1)$ and the partial subisomorphism itself is denoted by $\varphi|_{A(V_1)}$. The special case of the partial subisomorphism being defined over the complete vertex set of the smaller graph is denoted by $\varphi = \varphi|_{V_1}$.

A lower bound for the approximation distance of a partial subisomorphism can be obtained by independently minimizing distances between unassigned and assigned vertices. Formally, the lower bound is given by

$$\sum_{\{v_i,v_j\}\in E_1} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2$$

$$\begin{aligned} &\geq \sum_{\{v_i,v_j\}\in E_1, v_i,v_j\in A(V_1)} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2 + \sum_{v_0\in A(V_1)} c(v_0)^2 \\ &=: Val(\varphi|_{A(V_1)}), \end{aligned}$$

where $c(v_0) = \min_{v\in V_1 - A(V_1), w\in V_2 - \varphi(A(V_1))} |D_1(v_0,v) - D_2(\varphi(v_0),w)|$ for all $v_0 \in A(V_1)$.

Improved lower bounds can be constructed by cost minimal assignments in analogy to the heuristic distance list constructions of Section 3. Independent minimization over unassigned vertices is replaced by joint minimization. The vertex set of the bipartite graph for the assignment problem consist of both sets of unassigned vertices which are $V_1 - A(V_1)$ and $V_2 - \varphi(A(V_1))$. The cost values of the edges are given by squared errors of distance list approximations $DL(v,w)$, see Figure 12.

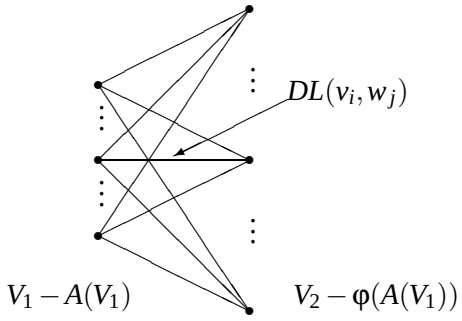


Figure 12: Assignment problem for improved lower bound of partial subisomorphism $\varphi|_{A(V_1)}$.

The improved lower bound is

$$\begin{aligned} &\sum_{\{v_i,v_j\}\in E_1} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2 \\ &\geq \sum_{\{v_i,v_j\}\in E_1, v_i,v_j\in A(V_1)} \Big(D_1(v_i,v_j) - D_2(\varphi(v_i),\varphi(v_j))\Big)^2 + val(\varphi|_{A(V_1)}) \\ &=: Val^*(\varphi|_{A(V_1)}), \end{aligned}$$

where $val(\varphi|_{A(V_1)})$ is the minimum cost value of the lower bounding assignment problem from Figure 12. The bounding strategy of a branch and bound algorithm for best graph approximation can now be readily specified as follows.

Bound
Case $A(V_1) \neq V_1$.
If $Val^*(\varphi|_{A(V_1)}) \leq M$ then refine $\varphi|_{A(V_1)}$
else ignore $\varphi|_{A(V_1)}$. (prune or bound).
Case $A(V_1) = V_1$.
If $Val^*(\varphi) = M$ then $L_{opt} = L_{opt} \cup \{\varphi\}$.
If $Val^*(\varphi) < M$ then $L_{opt} = \{\varphi\}$ and $M = Val^*(\varphi)$.

The value M is the approximation distance of the best subisomorphism found so far and L_{opt} is a list of all these best subisomorphisms. This results in the following branch and bound approach.

The algorithm operates on a list U of unexplored partial subisomorphisms until this list becomes empty. The initial setting of this list consists of partial subisomorphisms that make exactly one assignment.

B+B

1. Input graphs G_1, G_2 with distances D_1, D_2.
   Initialization. M cost of arbitrary subisomorphism. $U = \{\varphi|_{v_1}(v_1) = w_1, \ldots, \varphi|_{v_1}(v_1) = w_m\}$.
   $L_{opt} = \emptyset$.
2. While $U \neq \emptyset$ do
   (a) Selection $\varphi|_{A(V_1)} \in U$.
   (b) $U = U - \{\varphi|_{A(V_1)}\}$.
   (c) Computation of $Val^*(\varphi|_{A(V_1)})$.
   (d) (Bound)
   If $A(V_1) = V_1$ then
   If $Val^*(\varphi) = M$ then $L_{opt} = L_{opt} \cup \{\varphi\}$.
   If $Val^*(\varphi) < M$ then $L_{opt} = \{\varphi\}$ and $M = Val^*(\varphi)$.
   (e) (Branch)
   If $Val^*(\varphi|_{A(V_1)}) < M$ then $U = U \cup \bigcup_{w_0 \in V_2 - \varphi(A(V_1))} \{\varphi|_{A(V_1)\cup\{v_0\}}$ with $\varphi|_{A(V_1)\cup\{v_0\}}(v_0) = w_0\}$ for one $v_0 \in A(V_1)$.
3. Output list of optimal subisomorphisms L_{opt}.

The branch and bound procedure is informal in so far as the selection step 2(a) and the branching step 2(e) leave many ways of specialization. Removal is specified implicitly here which means that a partial subisomorphism selected in step 2(a) is removed anyway in step 2(b). Refinements of the partial subisomorphism are possibly added to U in the branching step. Whenever a partial subisomorphism is not pruned by the bounding step, the next iteration of step 2 may or may not select a refinement of this partial subisomorphism to continue with.

5. Conclusion

Best graph approximation has been formulated for labeled graphs in analogy to isomorphism for unlabeled graphs. Polynomial time approximation algorithms in terms of linear assignment problems have been given and exact algorithms have been outlined. All algorithms are independent from application domains.

Whenever an instance graph is to be matched to several instead of one model graph, this can obviously be done sequentially. The best match is given by the minimum over all approximation distances and a ranking of the matching results is given by increasingly sorted approximation distances.

References

Ravindra, K. Ahuja, Thomas L. Magnanti and James Orlin. *Network Flows*. Prentice Hall, Englewood Cliffs, 1993.

Endika Bengoetxea. *Inexact graph matching using estimation of distribution algorithms*. Ph.D. dissertation, University of the Basque Country, San Sebastian, 2002.

Tiberio S. Caetano, Terry Caelli, Dale Schuurmanns and Dante A.C Barone. Graphical models and point pattern matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, forthcoming.

Pierre-Antoine Champin and Christine Solnon. Measuring the similarity of labeled graphs. In *Proceedings of the Fifth International Conference on Case-Based Reasoning*, pages 80-95, Springer, LNCS 2689, Berlin, 2003.

Richard Desper and Martin Vingron. Tree fitting: topological reconstruction from ordinary least-squares edge length estimates. *Journal of Classification*, 19:87-112, 2002.

Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint to a new Computing Infrastructure*. 2nd ed., Elsevier, Amsterdam, 2004.

Satoru Fujishige. *Submodular Functions and Optimization*. North Holland, Amsterdam, 1991.

Michael R. Garey and David S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1981.

Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 18:377-388, 1996.

Gene H. Golub and Charles F. van Loan. *Matrix Computations*. 4th printing, John Hopkins University Press, Baltimore, 1985.

Godefrey H. Hardy, John E. Littlewood and George Polya. *Inequalities*. Cambridge University Press, Cambridge, 1948.

Adel Hlaoui and Shengrui Wang. A new algorithm for inexact graph matching. In *Proceedings of the International Conference on Pattern Recognition ICPR'02*, pages 180-183, Quebec, 2002.

Thomas Kämpke. Scalable distance similarity of chemical structures. *Combinatorial Chemistry and High Throughput Screening*. 7:11-21, 2004.

Sergey Melnik, Hector Garcia-Molina and Erhard Rahm. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 16th International Conference on Data Engineering ICDE*, 12 pages, San Diego, 2002.

Kurt Mehlhorn and Stephan Näher. *LEDA A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, 2000.

Bruno T. Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 20:493-504, 1998.

Bruno T. Messmer and Horst Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition* 32:1979-1998, 1999.

Apostolos N. Papadopoulos and Yannis Manolopoulos. Structure-based similarity search with graph histograms. In *Proceedings of the 10th International Workshop on Database and Expert System Applications DEXA*, pages 174-178, 1999.

Arthur R. Pope and David G. Lowe. Learning appearance models for object recognition. In *Proceedings of International Workshop on Object Representation for Computer Vision*, pages 201-219, Springer, Berlin, 1996.

Dennis Shasha, Jason T.L. Wang and Rosalba Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the Symposium on Principles of Database Systems*, pages 39-52, 2002.

Steven S. Skiena and Gopalakrishnan Sundaram. A partial digest approach to restriction site mapping. *Bulletin of Mathematical Biology*, 56:275-294, 1994.

Jason T.L. Wang, Bruce A. Shapiro, Dennis Shasha, Kaizhong Zhang and Kathleen M. Currey. An algorithm for finding the largest approximately common substructures of two trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:889-895, 1998.

Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, Christoph Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 19:775-779, 1997.

Laurenz Wiskott and Christoph Malsburg. Labeled bunch graphs for image analysis. US patent no. 6,356,659, 2002.

Journal of Machine Learning Research 7 (2006) 2087-2123 Submitted 2/05; Revised 6/06; Published 10/06

A Hierarchy of Support Vector Machines for Pattern Detection

Hichem Sahbi HS385@CAM.AC.UK
Machine Intelligence Laboratory
Department of Engineering
University of Cambridge
Trumpington Street
Cambridge, CB2 1PZ, UK

Donald Geman GEMAN@JHU.EDU
Center for Imaging Science
302 Clark Hall
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218, USA

Editor: Pietro Perona

Abstract

We introduce a computational design for pattern detection based on a tree-structured network of support vector machines (SVMs). An SVM is associated with each cell in a recursive partitioning of the space of patterns (hypotheses) into increasingly finer subsets. The hierarchy is traversed coarse-to-fine and each chain of positive responses from the root to a leaf constitutes a detection. Our objective is to design and build a network which balances overall error and computation.

Initially, SVMs are constructed for each cell with no constraints. This "free network" is then perturbed, cell by cell, into another network, which is "graded" in two ways: first, the number of support vectors of each SVM is reduced (by clustering) in order to adjust to a pre-determined, increasing function of cell depth; second, the decision boundaries are shifted to preserve all positive responses from the original set of training data. The limits on the numbers of clusters (virtual support vectors) result from minimizing the mean computational cost of collecting all detections subject to a bound on the expected number of false positives.

When applied to detecting faces in cluttered scenes, the patterns correspond to poses and the free network is already faster and more accurate than applying a single pose-specific SVM many times. The graded network promotes very rapid processing of background regions while maintaining the discriminatory power of the free network.

Keywords: statistical learning, hierarchy of classifiers, coarse-to-fine computation, support vector machines, face detection

1. Introduction

Our objective is to design and build a "pattern detection" system based on a tree-structured network of increasingly complex support vector machines (SVMs) (Boser et al., 1992; Osuna et al., 1997). The methodology is general, and could be applied to any classification task in machine learning in which there are natural groupings among the patterns (classes, hypotheses). The application which motivates this work is to detect and localize all occurrences in a scene of some particular object category based on a single, grey-level image. The particular example of detecting faces against

cluttered backgrounds provides a running illustration of the ideas where the groupings are based on pose continuity.

Our optimization framework is motivated by natural trade-offs among invariance, selectivity (background rejection rate) and the cost of processing the data in order to determine all detected patterns. In particular, it is motivated by the amount of computation involved when a single SVM, dedicated to a reference pattern (e.g., faces with a nearly fixed position, scale and tilt), is applied to many data transformations (e.g., translations, scalings and rotations). This is illustrated for face detection in Fig 1; a graded network of SVMs achieves approximately the same accuracy as a pattern-specific SVM but with order 100 to 1000 times fewer kernel evaluations, resulting from the network architecture as well as the reduced number of support vectors.

To design and construct such a graded network, we begin with a hierarchical representation of the space of patterns (e.g., poses of a face) in the form of a sequence of nested partitions, one for each level in a binary tree (Fleuret and Geman, 2001; Fleuret, 1999; Sahbi et al., 2002; Jung, 2001; Blanchard and Geman, 2005; Amit et al., 2004; Gangaputra and Geman, 2006a). Each cell - distinguished subset of patterns - encodes a simpler, sub-classification task and is assigned a binary classifier. The leaf cells represent the resolution at which we desire to "detect" the true pattern(s). There is also a "background class," for example, a complex and heterogeneous set of non-distinguished patterns, which is statistically dominant (i.e., usually true). A pattern is "detected" if the classifier for every cell which covers it responds positively.

Initially, SVMs are constructed for each cell in the standard way (Boser et al., 1992) based on a kernel and training data – positive examples (from a given cell) and negative examples ("background"). This is the "free network," or "f-network" $\{f_t\}$, where t denotes a node in the tree hierarchy. The "graded network," or "g-network" $\{g_t\}$, is indexed by the same hierarchy, but the number of intervening terms in each g_t is fixed in advance (by clustering those in f_t as in Schölkopf et al., 1998), and grows with the level of t. (From here on, the vectors appearing g_t will be referred to as "support vectors" even though, technically, they are constructed from the actual support vectors appearing in f_t.) Moreover, the decision boundaries are shifted to preserve all positive responses from the original set of training data; consequently, the false negative (missed detection) rate of g_t is at most that of f_t and any pattern detected by the f-network is also detected by the g-network. But the g-network will be far more efficient.

The limits on the numbers of support vectors result from solving a constrained optimization problem. *We minimize the mean computation necessary to collect all detections subject to a constraint on the rate of false detections.* (In the application to face detection, a false detection refers to finding a face amidst clutter.) Mean computation is driven by the background distribution. This also involves a model for how the selectivity of an SVM depends on complexity, which is assumed proportional to the number of support vectors, and invariance, referring to the "scope" of the underlying cell in the hierarchy.

In the free network, the complexity of each SVM decision function depends in the usual way on the underlying probability distribution of the training data. For instance, the decision function for a linearly separable training set might be expressed with only two support vectors, whereas the SVMs induced from complex tasks in object recognition usually involve many support vectors (Osuna et al., 1997). For the f-network, the complexity generally *decreases* as a function of depth due to the progressive simplification of the underlying tasks. This is illustrated in Fig 2 (left) for face detection; the classifiers f_t were each trained on 8000 positive examples and $50,000$ negative examples. Put differently, complexity increases with invariance.

Figure 1: Comparison between a single SVM (top row) dedicated to a nearly fixed pose and our designed network (bottom row) which investigates many poses simultaneously. The sizes of the three images are, left to right, 520×739, 462×294 and 662×874 pixels. The network achieves approximately the same accuracy as the pose-specific SVM but with order 100-1000 times fewer kernel evaluations. Some statistics comparing efficiency are given in Table 1.

Consider an SVM f in the f-network with N support vectors and dedicated to a particular hypothesis cell; this network is slow, but has high selectivity and few false negatives. The corresponding SVM g has a specified number n of support vectors with $n \leq N$. It is intuitively apparent that g is less selective; this is the price for maintaining the false negative rate and reducing the number of kernel evaluations. In particular, if n is very small, g will have low selectivity (cf. Fig 2 (right)). In general, of course, with no constraints, the fraction of support vectors provides a rough measure of the difficulty of the problem; here, however, we are *artificially* reducing the number of support vectors, thereby limiting the selectivity of the classifiers in the g-network.

Building *expensive* classifiers at the upper levels ($n \approx N$) leads to intensive early processing, even when classifying *simple* background patterns (e.g., flat areas in images), so the overall mean

Figures	"Mona Lisa"			"Singers"			"Star Trek"		
	1 SVM	**f-net**	**g-net**	**1 SVM**	**f-net**	**g-net**	**1 SVM**	**f-net**	**g-net**
# **Subimages Processed**	$2 \,.\, 10^5$	$2 \,.\, 10^3$	$2 \,.\, 10^3$	$5 \,.\, 10^4$	$8 \,.\, 10^2$	$8 \,.\, 10^2$	$2 \,.\, 10^5$	$4 \,.\, 10^3$	$4 \,.\, 10^3$
# **Kernel Evaluations**	$5 \,.\, 10^7$	10^7	$3 \,.\, 10^4$	$2 \,.\, 10^7$	$7 \,.\, 10^6$	10^4	$8 \,.\, 10^7$	$2 \,.\, 10^7$	$5 \,.\, 10^4$
Processing Time (s)	172.45	28.82	0.53	55.87	17.83	0.26	270.1	48.92	0.87
# **Raw Detections**	3	3	4	12	14	15	19	20	20

Table 1: Comparisons among i) a single SVM dedicated to a small set of hypotheses (in this case a constrained pose domain), ii) the f-network and iii) our designed g-network, for the images in Fig 1. For the single SVM, the position of the face is restricted to a 2×2 window, its scale to the range $[10, 12]$ pixels and its orientation to $[-5^0, +5^0]$; the original image is downscaled 14 times by a factor of 0.83 and for each scale the SVM is applied to the image data around each non-overlapping 2×2 block. In the case of the f and g-networks, we use the coarse-to-fine hierarchy and the search strategy presented here.

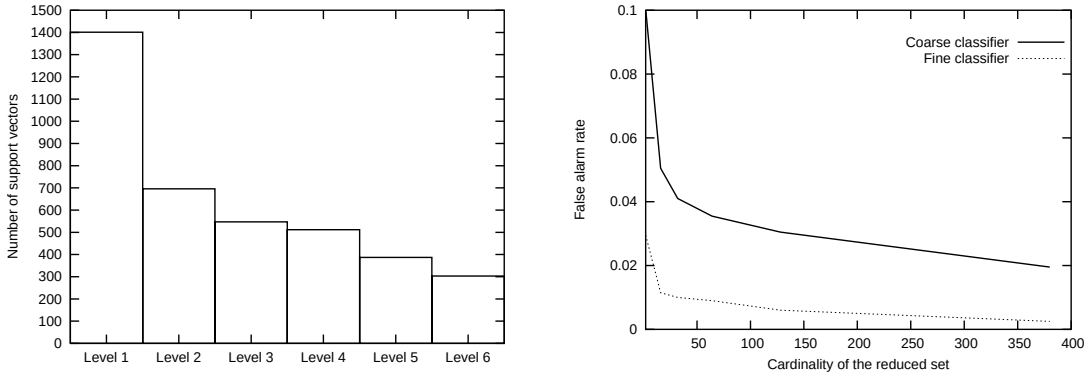


Figure 2: Left: The average number of support vectors for each level in an f-network built for face detection. The number of support vectors is decreasing due to progressive simplification of the original problem. Right: False alarm rate as a function of the number of support vectors using two SVM classifiers in the g-network with different pose constraints.

cost is also very large (cf. Fig 3, top rows). As an alternative to building the g-network, suppose we simply replace the SVMs in the upper levels of f-network with *very simple* classifiers (e.g., linear SVMs); then many background patterns will reach the lower levels, resulting in an overall loss of efficiency (cf. Fig 3, middle rows).

We focus in between these extremes and build $\{g_t\}$ to achieve a certain trade-off between cost and selectivity (cf. Fig 3, bottom rows). Of course, we cannot explore all possible designs so a model-based approach is necessary: The false alarm rate of each SVM is assumed to vary with complexity and invariance in a certain way. This functional dependence is consistent with the one proposed in Blanchard and Geman (2005), where the computational cost of a classifier is modeled as the product of an increasing function of scope and an increasing function of selectivity.

Finally, from the perspective of computer vision, especially image interpretation, the interest of this paper is the proposed architecture for aggregating binary classfiers such as SVMs for organized

Maximum Level Reached	1	2	3	4	5	6
# Samples (f-network)	1697	56	4	1	0	2
# Samples (heuristic)	936	555	135	17	54	63
# Samples (g-network)	1402	336	2	0	3	17

# **Kernel Evaluations**	$2-10$	$10-10^2$	10^2-10^3	10^3-10^4	10^4-10^5	10^5-10^6
# Samples (f-network)	0	0	0	1697	58	5
# Samples (heuristic)	936	755	67	2	0	0
# Samples (g-network)	1402	340	18	0	0	0

Figure 3: In order to illustrate varying trade-offs among cost, selectivity and invariance, and to demonstrate the utility of a principled, global analysis, we classified 1760 subimages of size 64×64 extracted from the image shown above using three different types of SVM hierarchies of depth six. In each case, the hierarchy was traversed coarse-to-fine. For each hierarchy type and each subimage, the upper table shows the distribution of the deepest level visited and the lower table shows the distribution of cost in terms of the total number of kernel evaluations. In both tables: Top row: The unconstrained SVM hierarchy ("f-network") with a Gaussian kernel at all levels; the SVMs near the top are very expensive (about 1400 support vectors at the root; see Fig 2) resulting in high overall cost. Middle row: An ad hoc solution: the same f-network, except with linear SVMs (which can be assumed to have only two support vectors) at the upper three levels in order to reduce computation; many images reach deep levels. Bottom row: The constrained SVM hierarchy ("g-network"), globally designed to balance error and computation; the number of (virtual) support vectors grows with depth.

scene parsing. For some problems, dedicating a single classifier to each hypothesis, or a cascade (linear chain) of classifiers to a small subset of hypotheses (see Section 2), and then training with existing methodology (even off-the-shelf software) might suffice, in fact provide state-of-the-art performance. This seems to the case for example with frontal face detection as long as large training sets are available, at least thousands of faces and sometimes billions of negative examples, for learning long, powerful cascades. However, those approaches are either very costly (see above) or may not scale to more ambitious problems involving limited data, or more complex and varied interpretations, because they rely too heavily on brute-force learning and lack the structure necessary to hardwire efficiency by simultaneously exploring multiple hypotheses.

We believe that hierarchies of classifiers provide such a structure. In the case of SVMs, which may require extensive computation, we demonstrate that building such a hierarchy with a global design which accounts for both cost and error is superior to either a single classifier applied a great many times (a form of template-matching) or a hierarchy of classifiers constructed independently, node-by-node, without regard to overall performance. We suspect that the same demonstration could be carried out with other "base classifiers" as long as there is a natural method for adjusting the amount of computation; in fact, the global optimization framework could be applied to improve other parsing strategies, such as cascades.

The remaining sections are organized as follows: A review of coarse-to-fine object detection, including related work on cascades, is presented in Section 2. In Section 3, we discuss hierarchical representation and search in general terms; decomposing the pose space provides a running example of the ideas and sets the stage for our main application - face detection. The f-network and g-network are defined in Section 4, again in general terms and the statistical framework and optimization problem are laid out in Section 5. This is followed in Section 6 by a new formulation of the "reduced set" method (Burges, 1996; Schölkopf et al., 1998), which is used to construct an SVM of specified complexity. These ideas are illustrated for a pose hierarchy in Section 7, including a specific instance of the model for chain probabilities and the corresponding minimization of cost subject to a constraint on false alarms. Experiments are provided in Section 8, where the g-network is applied to detect faces in standard test data, allowing us to compare our results with other methods. Finally, some conclusions are drawn in Section 9.

2. Coarse-to-Fine Object Detection

Our work is motivated by difficulties encountered in inducing semantic descriptions of natural scenes from image data. This is often computationally intensive due to the large amount of data to be processed with high precision. Object detection is such an example and has been widely investigated in computer vision; see for instance Osuna et al. (1997); Fleuret and Geman (2001); Kanade (1977); Schneiderman and Kanade (2000); Sung (1996); Viola and Jones (2001) for work on face detection. Nonetheless, there is as yet no system which matches human accuracy; moreover, the precision which is achieved often comes at the expense of run-time performance or a reliance on massive training sets.

One approach to computational efficiency is *coarse-to-fine processing*, which has been applied to many problems in computer vision, including object detection (Fleuret and Geman, 2001; Viola and Jones, 2001; Geman et al., 1995; Baker and Nayar, 1996; Amit and Geman, 1999; Rowley, 1999; Heisele et al., 2001), matching (Borgefors, 1988; Huttenlocher and Rucklidge, 1993; Gee and Haynor, 1996), optical flow (Battiti and Koch, 1991), tracking (Sobottka and Pittas, 1996) and

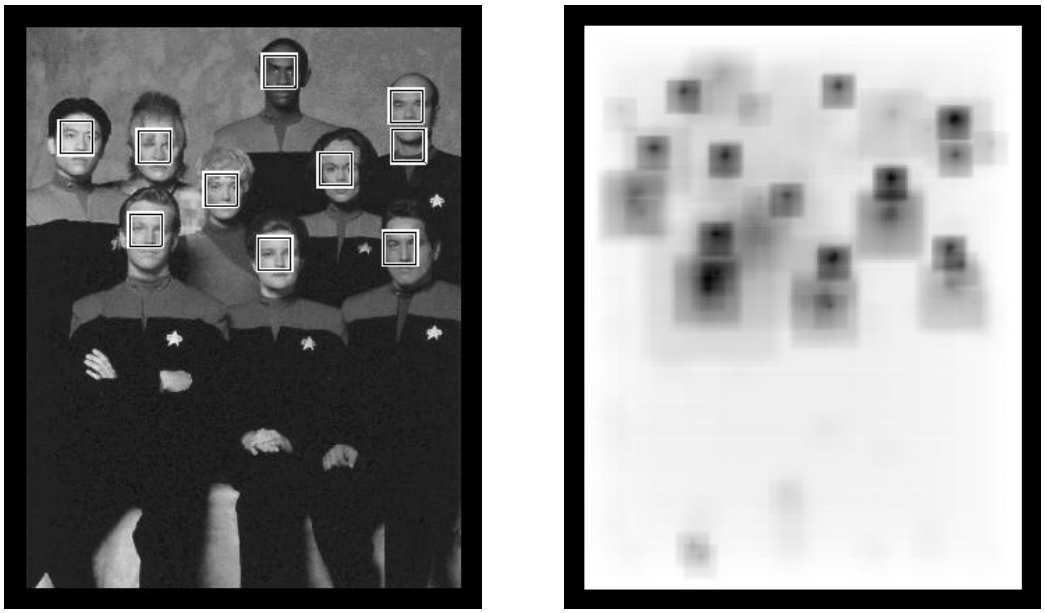


Figure 4: Left: Detections using our system. Right: The darkness of a pixel is proportional to the amount of local processing necessary to collect all detections.

other tasks such as compression, registration, noise reduction and estimating motion and binocular disparity. In the case of object detection, one strategy is to focus rapidly on areas of interest by finding characteristics which are common to many instantiations; in particular, background regions are quickly rejected as candidates for further processing (see Fig 4).

In the context of finding faces in cluttered scenes, Fleuret and Geman (2001) developed a fast, coarse-to-fine detector based on simple edge configurations and a hierarchical decomposition of the space of poses (location, scale and tilt). (Similar, tree-structured recognition strategies appear in Geman et al. (1995); Baker and Nayar (1996).) One constructs a family of classifiers, one for each cell in a recursive partitioning of the pose space and trained on a sub-population of faces meeting the pose constraints. A face is declared with pose in a leaf cell if all the classifiers along the chain from root to leaf respond positively. In general, simple and uniform structures in the scene are quickly rejected as face locations (i.e., very few classifiers are executed before all possible complete chains are eliminated) whereas more complex regions, for instance textured areas and face-like structures, require deeper penetration into the hierarchy. Consequently, the overall cost to process a scene is dramatically lower than looping over many individual poses, a form of template- matching (cf. Fig 1).

Work on cascades (Viola and Jones, 2001; Elad et al., 2002; Eveland et al., 2005; Keren et al., 2001; Socolinsky et al., 2003; Romdhani et al., 2001; Kienzle et al., 2004; Wu et al., 2005) is also motivated by an early rejection principle to exploit skewed priors (i.e., background domination). In that work, as in ours, the time required to classify a pattern (e.g., an input subimage) depends on the resemblance between that pattern and the objects of interest. For example, Viola and Jones (2001) developed an accurate, real-time face detection algorithm in the form of a cascade of boosted classifiers and computationally efficient feature detection. Other variations, such as those in Wu

et al. (2005); Romdhani et al. (2001) for face detection, and the cascade of inner products in Keren et al. (2001) for object identification, employ very simple linear classifiers. In nearly all cases the individual node learning problems are treated heuristically; an exception is Wu et al. (2005), where, for each node, the classifiers are designed to solve a (local) optimization problem constrained by desired (local) error rates.

There are several important differences between our work and cascades. Cascades are coarse-to-fine in the sense of background filtering whereas our approach is coarse-to-fine both in the sense of hierarchical pruning of the background class and representation of the space of hypotheses. In particular, cascades operate in a more or less brute-force fashion because every pose (e.g., position, scale and tilt) must be examined separately. In comparing the two strategies, especially our work with cascades of SVMs for face detection as in Kienzle et al. (2004); Romdhani et al. (2001), there is then a trade-off between very fast early rejection of individual hypotheses (cascades) and somewhat slower rejection of collections of hypotheses (tree-structured pruning).

No systematic comparison with cascades has been attempted. Moving beyond an empirical study would require a model for how cost scales with other factors, such as scope and selectivity. One such model was proposed in Blanchard and Geman (2005), in which the computational cost $C(f)$ of a binary classifier f dedicated to a set A of hypotheses (against a universal "background" alternative) is expressed as

$$C(f) = \Gamma(|A|) \times \Psi(1-\delta)$$

where δ is false positive rate of the classifier f (so $1-\delta$ is what we have called the selectivity) and Γ and Ψ are increasing functions with Γ subadditive and Ψ convex. (Some empirical justification for this model can be found in Blanchard and Geman (2005).) One can then compare the cost of testing a "small" set A of hypotheses (e.g., all poses over a small range of locations, scales and tilts, as in cascades) versus a "large" set $B \supset A$ (e.g., many poses simultaneously, as here). Under this cost model, and equalizing the selectivity, the subadditivity of Γ would render the test dedicated to B cheaper than doing the test dedicated to A approximately $\frac{|A|}{|B|}$ times, even ignoring the inevitable reduction in selectivity due to repeated tests.

More importantly, perhaps, it is not clear that cascades will scale to more ambitious problems involving many classes and instantiations since repeatedly testing a coarse set of hypotheses will lack selectivity and repeatedly testing a narrow one will require a great many implementations.

Finally, to our knowledge, the work presented in this paper is the first to consider a global construction of the system in an optimization framework. In particular, no global criteria appear in either Fleuret and Geman (2001) or Viola and Jones (2001); in the former, the edge-based classifiers are of roughly constant complexity whereas in the latter the complexity of the classifiers along the cascade is not explicitly controlled.

3. Hierarchical Representation and Search

Let Λ denote a set of "patterns" or "hypotheses" of interest. Our objective is to determine which, if any, of the hypotheses $\lambda \in \Lambda$ is true, the alternative being a statistically dominant "background" hypothesis $\{0\}$, meaning that most of the time 0 is the true explanation. Let Y denote the true state; $Y = 0$ denotes the background state. Instead of searching separately for each $\lambda \in \Lambda$, consider a coarse-to-fine search strategy in which we first try to exploit common properties ("shared features") of *all* hypotheses to "test" simultaneously for all $\lambda \in \Lambda$, that is, test the compound hypothesis $H : Y \in \Lambda$ against the alternative $H_0 : Y = 0$. If the test is negative, we stop and declare background; if

the test is positive, we separately test two disjoint subsets of Λ against H_0; and so forth in a nested fashion.

The tests are constructed to be very conservative in the sense that each false negative error rate is very small, that is, given that $Y \in A$, we are very unlikely to declare background if $A \subset \Lambda$ is the subset of hypotheses tested at a given stage. The price for this small false negative error is of course a non-negligible false positive error, particularly for testing "large" subsets A. However, this procedure is highly efficient, particularly under the background hypothesis. This "divide-and-conquer" search strategy has been extensively examined, both algorithmically (see for example Fleuret and Geman, 2001; Amit et al., 2004; Gangaputra and Geman, 2006a) and mathematically (Blanchard and Geman, 2005; Fleuret, 1999; Jung, 2001).

Note: There is an alternate formulation in which Y is directly modeled as a *subset* of Λ with $Y = \emptyset$ corresponding to the background state. In this case, at each node of the hierarchy, we are testing a hypothesis of the form $H : Y \cap A \neq \emptyset$ vs the alternative $Y \cap A = \emptyset$. In practice, the two formulations are essentially equivalent; for instance, in face detection, we can either "decompose" a set of "reference" poses which can represent at most one face and then execute the hierarchical search over subimages or collect all poses into one hierarchy with virtual tests near the root; see Section 7.1. We shall adopt the simpler formulation in which $Y \in \Lambda \cup \{0\}$.

Of course in practice we do all the splitting and construct all the "tests" in advance. (It should be emphasized that we are not constructing a decision tree; in particular, we are recursively partitioning the space of interpretations not features and, when the hierarchy is processed, a data point can travel down many branches and arrive at none of the leaves.) Then, on line, we need only execute the tests in the resulting hierarchy coarse-to-fine. Moreover, the tests are simply standard classifiers induced from training data - examples of $Y \in A$ for various subsets of A and examples of $Y = 0$. In particular, in the case of object detection, the classifiers are constructed from the usual types of image features, such as averages, edges and wavelets (Sahbi et al., 2002).

The nested partitions are naturally identified with a tree T. There is a subset Λ_t for each node t of T, including the root ($\Lambda_{root} = \Lambda$) and each leaf $t \in \partial T$. We will write $t = (l,k)$ to denote the k'th node of T at depth or level l. For example, in the case of a *binary* tree T with L levels, we then have:

$$\begin{cases} \Lambda_{1,1} = \Lambda \\ \Lambda_{l,k} = \Lambda_{l+1,2k-1} \cup \Lambda_{l+1,2k} \\ \Lambda_{l+1,2k-1} \cap \Lambda_{l+1,2k} = \emptyset \end{cases} \qquad l \in \{1,...,L-1\}, \quad k \in \left\{1,...,2^{l-1}\right\}.$$

The hierarchy can be manually constructed (as here, copying the one in Fleuret and Geman, 2001) or, ideally, learned.

Notice that the leaf cells $\Lambda_t, t \in \partial T$, needn't correspond to individual hypotheses. Instead, they represent the finest "resolution" at which we wish to estimate Y. More careful disambiguation among candidate hypotheses may require more intense processing, perhaps involving online optimization. It then makes sense to modify our definition of Y to reflect this possible coarsening of the original classification problem: the possible "class" values are then $\{0,1,...,2^{L-1}\}$, corresponding to "background" ($\{0\}$) and the 2^{L-1} "fine" cells at the leaves of the hierarchy.

Example: The Hierarchy for Face Detection. Here, the *pose* of an object refers to parameters characterizing its geometric appearance in the image. Since we are searching for instances of a

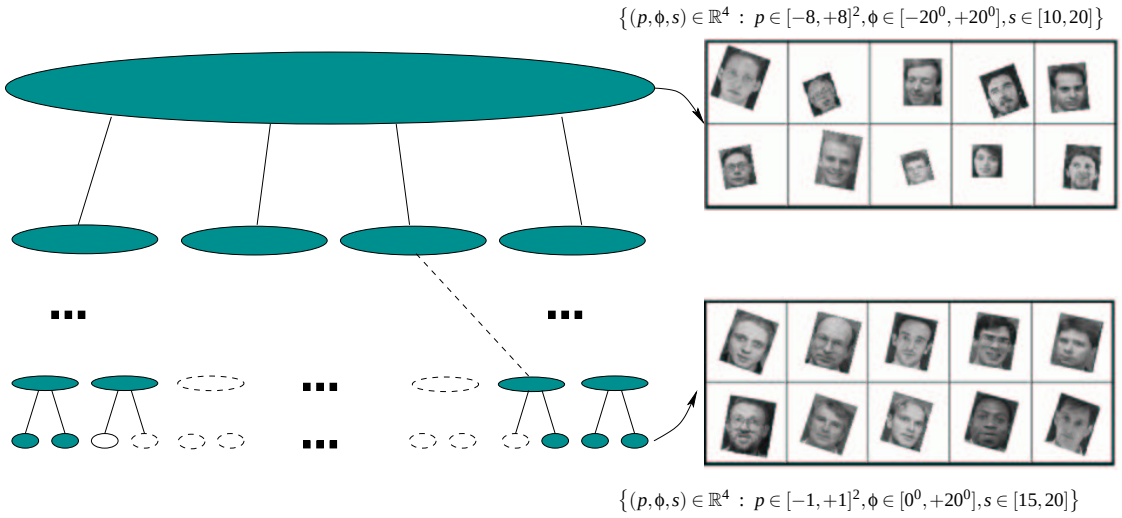


Figure 5: An illustration of the pose hierarchy showing a sample of faces at the root cell and at one of the leaves.

one object class – faces – the family of hypotheses of interest is a set of poses Λ. Specifically, we focus attention on the position, tilt and scale of a face, denoted $\theta = (p, \phi, s)$, where p is the midpoint between the eyes, s is the distance between the eyes and ϕ is the angle with the line orthogonal to the segment joining the eyes. We then define

$$\Lambda = \left\{(p,\phi,s) \in \mathbb{R}^4 \ : \ p \in [-8,+8]^2, \phi \in [-20^0,+20^0], s \in [10,20]\right\}.$$

Thus, we regard Λ as a "reference set" of poses in the sense of possible instantiations of a single face within a given 64×64 image assuming that the position is restricted to a subwindow (e.g., an 16×16 centered in the subimage) and the scale to the stated range. The "background hypothesis" is "no face" (with pose in Λ). The leaves of T do not correspond to individual poses $\theta \in \Lambda$; for instance, the final resolution on position is a 2×2 window. Hence, each "object hypothesis" is a small collection of fine poses.

The specific hierarchy used in our experiments is illustrated in Fig (5). It has six levels $(L = 6)$, corresponding to three quaternary splits in location (four 8×8 blocks, etc.) and one binary split both on tilt and scale. Therefore, writing ν_l for the number of cells in T at depth l: $\nu_1 = 1$, $\nu_2 = 4^1$, $\nu_3 = 4^2 = 16$, $\nu_4 = 4^3 = 64$, $\nu_5 = 2\ 4^3 = 128$ and $\nu_6 = 2^2\ 4^3 = 256$.

This is the same, manually-designed, pose hierarchy that was used in Fleuret and Geman (2001). The partitioning based on individual components, as well as the splitting order, is entirely ad hoc. The important issue of how to automatically design or learn the "divide-and-conquer" architecture is not considered here. Very recent work on this topic appears in Fan (2006) and Gangaputra and Geman (2006a).

Search Strategy:

Consider coarse-to-fine search in more detail. Let X_t be the test or classifier associated with node t, with $X_t = 1$ signaling the acceptance of $H_t : Y \in \Lambda_t$ and $X_t = 0$ signaling the acceptance

of $H_0 : Y = 0$. Also, let $\omega \in \Omega$ represent the underlying data or "pattern" upon which the tests are based; hence the true class of ω is $Y(\omega)$ and $X_t : \Omega \longrightarrow \{0,1\}$.

The result of coarse-to-fine search applied to ω is a *subset* $\mathbf{D}(\omega) \subset \Lambda$ of "detections", possibly empty, defined to be all $\lambda \in \Lambda$ for which $X_t(\omega) = 1$ for every test which "covers" λ, that is, for which $\lambda \in \Lambda_t$. Equivalently, $\mathbf{D}$ is the union over all $\Lambda_t, t \in \partial T$ such that $X_t = 1$ *and the test corresponding to every ancestor of* $t \in \partial T$ *is positive*, that is, all "complete chains of ones" (cf. Fig 6, B).

Both breadth-first and depth-first coarse-to-fine search lead to the same set $\mathbf{D}$. Breadth-first search is illustrated in Fig (6, C): Perform $X_{1,1}$; if $X_{1,1} = 0$, stop and declare $\mathbf{D} = \emptyset$; if $X_{1,1} = 1$, perform *both* $X_{2,1}$ *and* $X_{2,2}$ and stop only if both are negative; etc. Depth-first search explores the sub-hierarchy rooted at a node t before exploring the brother of t. In other words, if $X_t = 1$, we *visit recursively* the sub-hierarchies rooted at t; if $X_t = 0$ we "cancel" all the tests in this sub-hierarchy. In both cases, *a test is performed if and only if all its ancestors are performed and are positive*. (These strategies are not the same if our objective is only to determine whether or not $\mathbf{D} = \emptyset$; see the analysis in Jung (2001).)

Notice that $\mathbf{D} = \emptyset$ if and only if there is a "null covering" of the hierarchy in the sense of a collection of negative responses whose corresponding cells cover all hypotheses in Λ. The search is terminated upon finding such a null covering. Thus, for example, if $X_{1,1} = 0$, the search is terminated as there cannot be a complete chain of ones; similarly, if $X_{2,1} = 0$ and $X_{3,3} = X_{3,4} = 0$, the search is terminated.

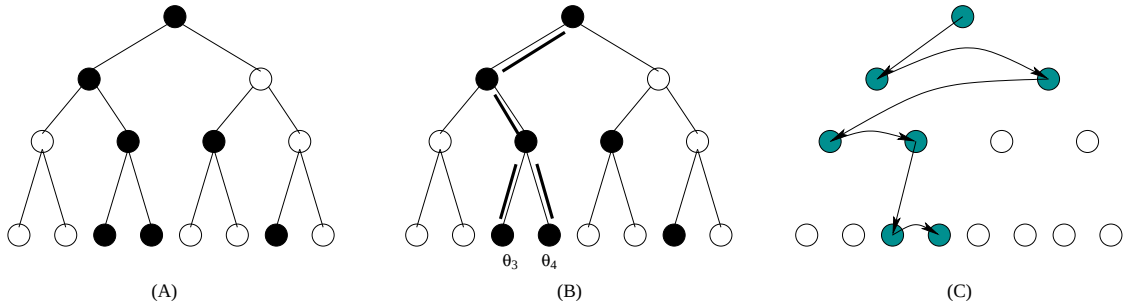


Figure 6: A hierarchy with fifteen tests. (A) The response to an input image were all the tests to be performed; the positive tests are shown in black and negative tests in white. (B) There are two complete chains of ones; in the case of object detection, the detected pose is the average over those in the two corresponding leaves. (C) The breadth-first search strategy with the executed tests are shown in color; notice that only seven of the tests would actually be performed.

Example: The Search Strategy for Face Detection. Images ω are encoded using a vector of wavelet coefficients; in the remainder of this paper we will write x to denote this vector of coefficients computed on a given 64×64 subimage. If $\mathbf{D}(x) \neq \emptyset$, the estimated pose of the face detected in ω is obtained by averaging over the "pose prototypes" of each leaf cell represented in $\mathbf{D}$, where the pose prototype of Λ_t is the midpoint (cf. Fig 6, B).

A scene is processed by visiting *non-overlapping* 16×16 blocks, processing the surrounding image data to extract the features (wavelet coefficients) and classifying these features using the

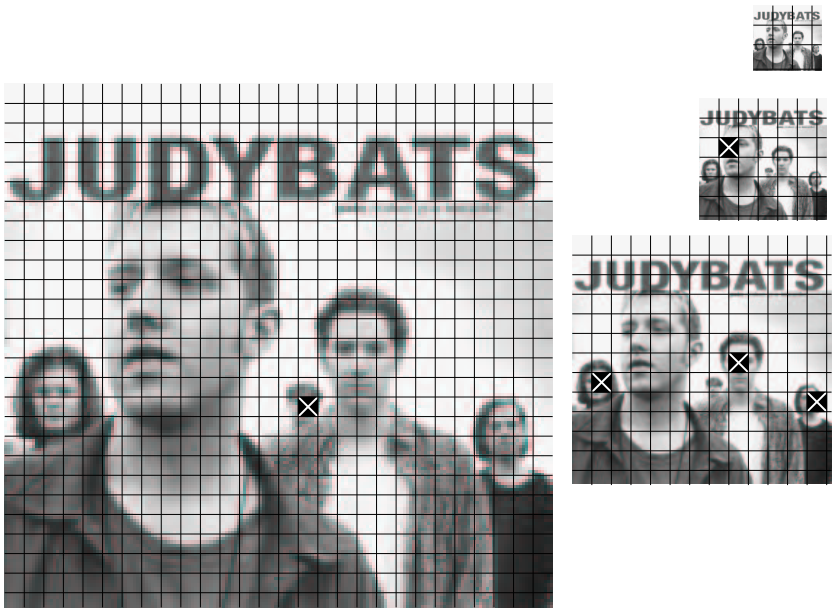


Figure 7: Multi-scale search. The original image (on the left) is downscaled three times. For each scale, the base face detector visits each *non-overlapping* 16×16 block, and searches the surrounding image data for all faces with position in the block, scale anywhere in the range $[10,20]$ and in-plane orientation in the range $[-20^o,+20^o]$.

search strategy described earlier in this section. This process makes it possible to detect all faces whose scale s lies in the interval $[10,20]$ and whose tilt belongs to $[-20^o,+20^o]$. Faces at scales $[20,160]$ are detected by repeated down-sampling (by a factor of 2) of the original image, once for scales $[20,40]$, twice for $[40,80]$ and thrice for $[80,160]$(cf. Fig 7). Hence, due to high invariance to scale in the base detector, only four scales need to be investigated altogether.

Alternatively, we can think of an extended hierarchy over all possible poses, with initial branching into disjoint 16×16 blocks and disjoint scale ranges, and with virtual tests in the first two layers which are passed by all inputs. Given color or motion information (Sahbi and Boujemaa, 2000), it might be possible to design a test which handles a set of poses larger than Λ; however, our test at the root (accounting simultaneously for all poses in Λ) is already quite coarse.

4. Two SVM Hierarchies

Suppose we have a training set $\mathcal{T} = \{(\omega_1, y_1), ..., (\omega_n, y_n)\}$. In the case of object detection, each ω is some 64×64 subimage taken, for example, from the Web, and either belongs to the "object examples" $\mathcal{L}$ (subimages ω for which $Y(\omega) \neq 0$) or "background examples" $\mathcal{B}$ (subimages for which $Y(\omega) = 0$).

All tests $X_t, t \in T$, are based on SVMs. We build one hierarchy, the *free network* or *f-network* for short, with no constraints, that is, in the usual way from the training data once a kernel and any other parameters are specified (Boser et al., 1992). The other hierarchy, the *graded network* or *g-network*, is designed to meet certain error and complexity specifications.

4.1 The f-network

Let f_t be an SVM dedicated to separating examples of $Y \in \Lambda_t$ from examples of $Y = 0$. (In our application to face detection, we train f_t based on face images ω with pose in Λ_t.) The corresponding test is simply $1_{\{f_t > 0\}}$. We refer to $\{f_t, t \in T\}$ as the *f-network*. In practice, the number of support vectors decreases with the depth in the hierarchy since the classification tasks are increasingly simplified; see Fig 2, left. We assume the false negative rate of f_t is very small for each t; in other words, $f_t(\omega) > 0$ for nearly all patterns ω for which $Y(\omega) \in \Lambda_t$. Finally, denote the corresponding data-dependent set of detections of the f-network by $\mathbf{D}_f$.

4.2 The g-network

The *g-network* is based on the same hierarchy $\{\Lambda_t\}$ as the f-network. However, for each cell Λ_t, a simplified SVM decision function g_t is built by reducing the complexity of the corresponding classifier f_t. The set of hypotheses detected by the g-network is denoted by $\mathbf{D}_g$. The targeted complexity of g_t is determined by solving a constrained minimization problem (cf. Section 5).

We want g_t to be *both* efficient *and* respect the constraint of a negligible false negative rate. As a result, for nodes t near the root of T the false positive rate of g_t will be higher than that of the corresponding f_t since low cost comes at the expense of a weakened background filter. Put differently, we are willing to sacrifice selectivity for efficiency, but not at the expense of missing (many) instances of our targeted hypotheses. Thus, for both networks, a positive test by no means signals the presence of a targeted hypothesis, especially for the very computationally efficient tests in the g-network near the top of the hierarchy.

Instead of imposing an absolute constraint on the false negative error, we impose one *relative* to the f-network, referred to as the *conservation hypothesis*: For each $t \in T$ and $\omega \in \Omega$:

$$f_t(\omega) > 0 \Rightarrow g_t(\omega) > 0.$$

This implies that an hypothesis detected by the f-network is also detected by the g-network, namely

$$\mathbf{D}_f(\omega) \subset \mathbf{D}_g(\omega), \ \forall \omega \in \Omega.$$

Consider two classifiers g_t and g_s in the g-network and suppose node s is deeper than node t. With the *same* number of support vectors, g_t will generally produce more false alarms than g_s since more invariance is expected of g_t (cf. Fig 2, right). In constructing the g-network, all classifiers at the same level will have the same number of support vectors and are then expected to have approximately the same false alarm rate (cf. Fig 8).

In the following sections, we will introduce a model which accounts for both the overall mean cost and the false alarm rate. This model is inspired by the trade-offs among selectivity, cost and invariance discussed above. The proposed analysis is performed under the assumption that there exists a convex function which models the false alarm rate as a function of the number of support vectors and the degree of "pose invariance".

5. Designing the g-network

Let P be a probability distribution on Ω. Write $\Omega = \mathcal{L} \cup \mathcal{B}$, where $\mathcal{L}$ denotes the set of all possible patterns for which $Y(\omega) > 0$, that is, $Y \in \{1, ..., 2^{L-1}\}$, hence a targeted pattern, and $\mathcal{B} = \mathcal{L}^c$

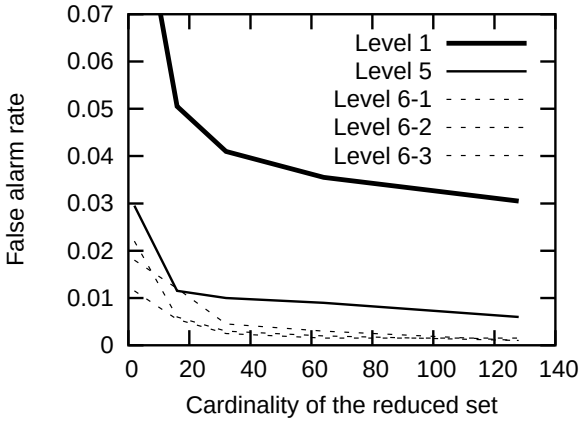


Figure 8: For the root cell, a particular cell in the fifth level and three particular pose cells in the sixth level of the g-network, we built SVMs with varying numbers of (virtual) support vectors. All curves show false alarm rates with respect to the number of (virtual) support vectors. For the sixth level, and in the regime of fewer than 10 (virtual) support vectors, the false alarm rates show considerable variation, but have the same order of magnitude. These experiments were run on background patterns taken from 200 images including highly textured areas (flowers, houses, trees, etc.)

contains all the background patterns. Define $P_0(.) = P(.|Y=0)$ and $P_1(.) = P(.|Y>0)$, the conditional probability distributions on background and object patterns, respectively. Throughout this paper, we assume that $P(Y=0) >> P(Y>0)$, which means that the presence of the targeted pattern is considered to be a rare event in data sampled under P.

Face Detection Example (cont): We might take P to be the empirical distribution on a huge set of 64×64 subimages taken from the Web. Notice that, given a subimage selected at random, the probability to have a face present with location near the center is very small.

Relative to the problem of deciding $Y = 0$ vs $Y \neq 0$, that is, deciding between "background" and "object" (some hypothesis in Λ), the two error rates for the f-network are $P_0(\mathbf{D}_f \neq \emptyset)$, the false positive rate, and $P_1(\mathbf{D}_f = \emptyset)$, the false negative rate. The total error rate is, $P_0(\mathbf{D}_f \neq \emptyset)P(Y=0) + P_1(\mathbf{D}_f = \emptyset)\ P(Y>0)$. Clearly this total error is largely dominated by the false alarm rate.

Recall that for each node $t \in T$ there is a subset Λ_t of hypotheses and an SVM classifier g_t with n_t support vectors. The corresponding test for checking $Y \in \Lambda_t$ against the background alternative is $1_{\{g_t>0\}}$. Our objective is to provide an optimization framework for specifying $\{n_t\}$.

5.1 Statistical Model

We now introduce a *statistical model* for the behavior of the g-network. Consider the event that a background pattern traverses the hierarchy up to node t, namely the event $\bigcap_{s\in\mathcal{A}_t}\{g_s > 0\}$, where $\mathcal{A}_t$ denotes the set of ancestors of t – the nodes from the parent of t to the root, inclusive. We will

assume that the probability of this event under P_0, namely

$$P_0(g_s > 0, s \in \mathcal{A}_t), \tag{1}$$

depends only on the level of t in the hierarchy (and not on the particular set $\mathcal{A}_t$) as well as on the numbers $n_1, ..., n_{l-1}$ of support vectors at levels one through $l-1$, where t is at level l. These assumptions are reasonable and roughly satisfied in practice; see Sahbi (2003).

The probability in (1) that a background pattern reaches depth l, that is, there is a chain of positive responses of length $l-1$, is then denoted by $\delta(l-1;\mathbf{n})$, where $\mathbf{n}$ denotes the sequence $(n_1, ..., n_L)$. Naturally, we assume that $\delta(l;\mathbf{n})$ is decreasing in l. In addition, it is natural to assume that $\delta(l;\mathbf{n})$ is a decreasing function of each $n_j, 1 \leq j \leq l$. In Section 7 we will present an example of a two-dimensional parametric family of such models.

There is an equivalent, and useful, reformulation of these joint statistics in terms of *conditional false alarm rates* (or conditional selectivity). One specifies a model by prescribing the quantities

$$P_0(g_{root} > 0),\ P_0(g_t > 0 | g_s > 0, s \in \mathcal{A}_t) \tag{2}$$

for all nodes t with $2 \leq l(t) \leq L$. Clearly, the probabilities in (1) determine those in (2) and vice-versa.

Note: We are not specifying a probability distribution on the entire family of variables $\{g_t, t \in T\}$, equivalently, on all labeled trees. However, it can be shown that any (decreasing) sequence of positive numbers $p_1, ..., p_L$ for the chain probabilities is "consistent" in the sense of providing a well-defined distribution on *traces*, the labeled subtrees that can result from coarse-to-fine processing, which necessarily are labeled "1" at all internal nodes; see Gangaputra and Geman (2006b).

In order to achieve efficient computation (at the expense of extra false alarms relative to the f-network), we choose $\mathbf{n} = (n_1, ..., n_L)$ to solve a constrained minimization problem based on the mean total computation in evaluating the g-network and a bound on the expected number of detected background patterns:

$$\begin{array}{ll} \min\limits_{\mathbf{n}} & C(n_1, ..., n_L) \\ \text{s.t.} & E_0(|\mathbf{D}_g|) \ \leq \ \mu \end{array} \tag{3}$$

where E_0 refers to expectation with respect to the probability measure P_0. We first compute this expected cost, then consider the constraint in more detail and finally turn to the problem of choosing the model.

Note: In our formulation, we are assuming that overall average computation is well-approximated by estimating total computation under the background probability by itself rather than with respect to a mixture model which accounts for object instances. In other words, we are assuming that background processing accounts for most of the work. Of course, in reality, this is not strictly the case, especially at the lower levels of the hierarchy, at which point evidence has accrued for the presence of objects and the conditional likelihoods of object and background are no longer extremely skewed in favor of the latter. However, computing under a mixture model would severely complicate the analysis. Moreover, since extensive computation is rarely performed, we believe our approximation is valid; whereas an expanded analysis might somewhat change the design of the lower levels, it would not appreciably reduce overall cost.

5.2 Cost of the g-network

Let c_t indicate the cost of performing g_t and assume

$$c_t = a\, n_t + b.$$

Here a represents the cost of kernel evaluation and b represents the cost of "preprocessing" – mainly extracting features from a pattern ω (e.g., computing wavelet coefficients in a subimage). We will also assume that all SVMs at the same level of the hierarchy have the same number of support vectors, and hence approximately the same cost.

Recall that ν_l is the number of nodes in T at level l; for example, for a binary tree, $\nu_l = 2^{l-1}$. The global cost is then:

$$\begin{aligned} Cost &= \sum_t 1_{\{g_t \text{ is performed}\}}\, c_t \\ &= \sum_{l=1}^{L} \sum_{k=1}^{\nu_l} 1_{\{g_{l,k} \text{ is performed}\}}\, c_{l,k} \end{aligned} \tag{4}$$

since g_t is performed in the coarse-to-fine strategy if and only if $g_s > 0\ \forall s \in \mathcal{A}_t$, we have, from equation (4), with $\delta(0;\mathbf{n}) = 1$,

$$\begin{aligned} \mathcal{C}(n_1,...,n_L) &= E_0\,(Cost) \\ &= \sum_{l=1}^{L} \sum_{k=1}^{\nu_l} P_0\,(\{g_{l,k} \text{ is performed}\})\ c_{l,k} \\ &= \sum_{l=1}^{L} \sum_{k=1}^{\nu_l} \delta(l-1;\mathbf{n})\, c_{l,k} \\ &= \sum_{l=1}^{L} \nu_l\, \delta(l-1;\mathbf{n})\, c_l \\ &= a \sum_{l=1}^{L} \nu_l\, \delta(l-1;\mathbf{n})\, n_l + b \sum_{l=1}^{L} \nu_l\, \delta(l-1;\mathbf{n}). \end{aligned}$$

The first term is the SVM cost and the second term is the total preprocessing cost. In the application to face detection we shall assume the preprocessing cost – the computation of Haar wavelet coefficients for a given subimage – is small compared with kernel evaluations, and set $a = 1$ and $b = 0$. Hence,

$$\mathcal{C}(n_1,...,n_L) = n_1 + \sum_{l=2}^{L} \nu_l\, n_l\, \delta(l-1;\mathbf{n}).$$

5.3 Penalty for False Detections

Recall that $\mathbf{D}_g(\omega)$ – the set of detections – is the union of the sets Λ_t over all *terminal* nodes t for which there is a complete chain of positive responses from the root to t. For simplicity, we assume that $|\Lambda_t|$ is the same for all terminal nodes t. Hence $|\mathbf{D}_g|$ is proportional to the total number of complete chains:

$$|\mathbf{D}_g| \propto \sum_{t \in \partial T} 1_{\{g_t > 0\}} \prod_{s \in \mathcal{A}_t} 1_{\{g_s > 0\}}.$$

It follows that

$$\begin{aligned} E_0|\mathbf{D}_g| &\propto E_0 \sum_{t\in\partial T} 1_{\{g_t>0\}} \prod_{s\in\mathcal{A}_t} 1_{\{g_s>0\}} \\ &= \sum_{t\in\partial T} \delta(L;\mathbf{n}) \\ &= \nu_L \delta(L;\mathbf{n}) \end{aligned}$$

By the Markov inequality,

$$P_0(\mathbf{D}_g \neq \emptyset) = P_0(|\mathbf{D}_g| \geq 1) \leq E_0|\mathbf{D}_g|.$$

Hence bounding the mean size of $\mathbf{D}_g$ also yields the same bound on the false positive probability. However, we cannot calculate $P_0(|\mathbf{D}_g| \geq 1)$ based only on our model $\{\delta(l;\mathbf{n})\}_l$ since this would require computing the probability of a *union* of events and hence a model for the dependency structure *among* chains.

Finally, since we are going to use the SVMs in the f-network to build those in the g-network, the number of support vectors n_l for each SVM in the g-network at level l is bounded by the corresponding number, N_l, for the f-network. (Here, for simplicity, we assume that N_t is roughly constant in each level; otherwise we take the minimum over the level.)

Summarizing, our constrained optimization problem (3) becomes

$$\min_{n_1,\ldots,n_L} \; n_1 + \sum_{l=2}^{L} \nu_l \, n_l \, \delta(l-1;\mathbf{n}) \;\; \text{s.t.} \;\; \begin{cases} \nu_L \delta(L;\mathbf{n}) \leq \mu \\ 0 < n_l \leq N_l. \end{cases} \tag{5}$$

5.4 Choice of Model

In practice, one usually stipulates a *parametric family* of statistical models (in our case the chain probabilities) and estimates the parameters from data. Let $\{\delta(l;\mathbf{n},\beta)\}, \beta \in B\}$ denote such a family where β denotes a parameter vector, and let $\mathbf{n}^* = \mathbf{n}^*(\beta)$ denote the solution of (5) for model β. We propose to choose β by comparing population and empirical statistics. For example, we might select the model for which the chain probabilities best match the corresponding relative frequencies when the g-network is constructed with $\mathbf{n}^*(\beta)$ and run on sample background data. Or, we might simply compare predicted and observed numbers of background detections:

$$\beta^* \doteq \arg\min_{\beta\in B} |E_0\left(|\mathbf{D}_g|;\mathbf{n}^*(\beta)\right) - \hat{\mu}_0(\mathbf{n}^*(\beta))|$$

where $\hat{\mu}_0(\mathbf{n}^*(\beta))$ is the average number of detections observed with the g-network constructed from $\mathbf{n}^*(\beta)$. In Section 7 we provide a concrete example of this model estimation procedure.

6. Building the g-network

Since the construction is node-by-node, we can assume throughout this section that t is fixed and that $\Lambda = \Lambda_t$ and $f = f_t$ are given, where f is an SVM with N_f support vectors. Our objective is to build an SVM $g = g_t$ with two properties:

- g is to have $N_g < N_f$ support vectors, where $N_g = n^*_{l(t)}$ and $\mathbf{n}^* = (n^*_1,\ldots,n^*_L)$ is the optimal design resulting from the upcoming reformulation (8) of (5) which incorporates our model for $\{\delta(l:\mathbf{n})\}$; and

- The "conservation hypothesis" is satisfied; roughly speaking this means that detections under f are preserved under g.

Let $x(\omega) \in \mathbb{R}^q$ be the feature vector; for simplicity, we suppress the dependence on ω. The SVM f is constructed as usual based on a kernel K which implicitly defines a mapping Φ from the input space $\mathbb{R}^q$ into a high-dimensional Hilbert space $\mathcal{H}$ with inner product denoted by $\langle\ \rangle$. Support vector training (Boser et al., 1992) builds a maximum margin hyperplane (w_f, b_f) in $\mathcal{H}$. Re-ordering the training set as necessary, let $\left\{\Phi(v^{(1)}), ..., \Phi(v^{(N_f)})\right\}$ and $\{\alpha_1, ..., \alpha_{N_f}\}$ denote, respectively, the support vectors and the training parameters. The normal w_f of the hyperplane is $w_f = \sum_{i=1}^{N_f} \alpha_i\, y^{(i)}\, \Phi(v^{(i)})$.

The decision function in $\mathbb{R}^q$ is non-linear:

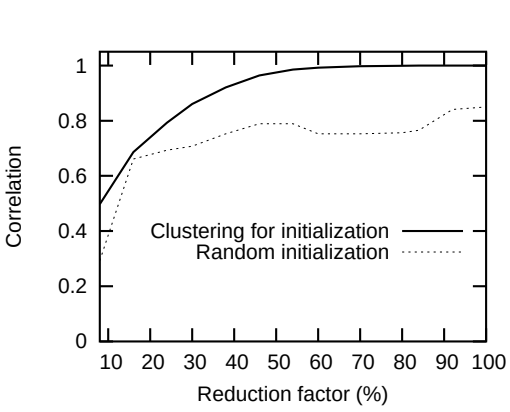
$$f(x) = \langle w_f, \Phi(x) \rangle + b_f = \sum_{i=1}^{N_f} \alpha_i\, y^{(i)}\, K(v^{(i)}, x) + b_f.$$

The parameters $\{\alpha_i\}$ and b_f can be adjusted to ensure that $\|w_f\|^2 = 1$.

The objective now is to determine (w_g, b_g), where

$$g(x) = \langle w_g, \Phi(x) \rangle + b_g = \sum_{k=1}^{N_g} \gamma_k\, K(z^{(k)}, x) + b_g.$$

Here $\mathcal{Z} = \left\{z^{(1)}, ..., z^{(N_g)}\right\}$ is the reduced set of support vectors (cf. Fig 9, right), $\gamma = \left\{\gamma_1, ..., \gamma_{N_g}\right\}$ the underlying weights, and the labels $y^{(k)}$ have been absorbed into γ.



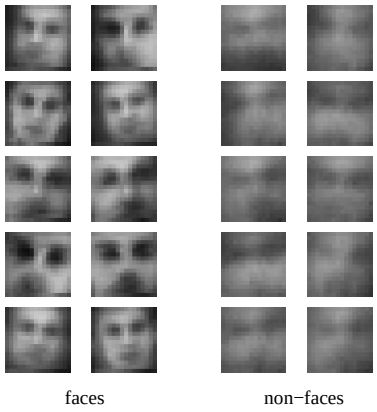


Figure 9: Left: The correlation is illustrated with respect to the reduction factor ($\frac{N_g}{N_f} \times 100$). These experiments are performed on a root SVM classifier using the Gaussian kernel (σ is set to 100 using cross-validation). Right: Visual appearance of some face and "non-face" virtual support vectors.

6.1 Determining w_g: Reduced Set Technique

The method we use is a modification of the *reduced set technique* (RST) (Burges, 1996; Burges and Schölkopf, 1997). Choose the new normal vector to satisfy:

$$w_g^* = \arg \min_{w_g : \|w_g\|=1} \langle w_g - w_f, w_g - w_f \rangle.$$

Using the kernel trick, and since $w_g = \sum_{k=1}^{N_g} \gamma_k \, \Phi(z^{(k)})$, the new optimization problem becomes:

$$\min_{\mathcal{Z},\gamma} \quad \sum_{k,l} \gamma_k \gamma_l \, K(z^{(k)}, z^{(l)}) \; + \; \sum_{i,j} \alpha_i \alpha_j \, y^{(i)} \, y^{(j)} \, K(v^{(i)}, v^{(j)}) - 2 \sum_{k,i} \gamma_k \alpha_i \, y^{(i)} \, K(z^{(k)}, v^{(i)}). \qquad (6)$$

For some kernels (for instance the Gaussian), the function to be minimized is not convex; consequently, with standard optimization techniques such as conjugate gradient, the normal vector resulting from the final solution $(\mathcal{Z}, \gamma)$ is a poor approximation to w_f (cf. Fig 9, left). This problem was analyzed in Sahbi (2003), where it is shown that, in the context of face detection, a good initialization of the minimization process can be obtained as follows: First cluster the initial set of support vectors $\{\Phi(v^{(1)}), ..., \Phi(v^{(N_f)})\}$, resulting in N_g centroids, each of which then represents a dense distribution of the original support vectors. Next, each centroid, which is expressed as a linear combination of original support vectors, is replaced by one support vector which best approximates this linear combination. Finally, this new reduced set is used to initialize the search in (6) in order to improve the final solution. Details may be found in Sahbi (2003) and the whole process is illustrated in Section 7.

6.2 Determining b_g: Conservation Hypothesis

Regardless of how b_g is selected, g is clearly less powerful than f. However, in the hierarchical framework, particularly near the root, the two types of mistakes (namely not detecting patterns in Λ and detecting background) are not equally problematic. Once a distinguished pattern is rejected from the hierarchy it is lost forever. Hence we prefer to severely limit the number of missed detections at the expense of additional false positives; hopefully these background patterns will be filtered out before reaching the leaves.

We make the assumption that the classifiers in the f-network have a very low false negative rate. Ideally, we would choose b_g such that $g(x(\omega)) > 0$ for every $\omega \in \Omega$ for which $f(x(\omega)) > 0$. However, this results in an unacceptably high false positive rate. Alternatively, we seek to minimize

$$P_0 (\, g \, \geq \, 0 \mid f \, < \, 0 \,)$$

subject to

$$P_1 (\, g \, < \, 0 \mid f \, \geq 0 \,) \leq \varepsilon.$$

Since we do not know the joint law of (f, g) under either P_0 or P_1, these probabilities are estimated empirically: for each b_g calculate the conditional relative frequencies using the training data and then choose the optimal b_g based on these estimates.

7. Application to Face Detection

We apply the general construction of the previous sections to a particular two-class problem – face detection – which has been widely investigated, especially in the last ten years. Existing methods

include artificial neural networks (Schneiderman and Kanade, 2000; Sung, 1996; Rowley et al., 1998; Féraud et al., 2001; Garcia and Delakis, 2004), networks of linear units (Yang et al., 2000), support vector machines (Osuna et al., 1997; Evgeniou et al., 2000; Heisele et al., 2001; Romdhani et al., 2001; Kienzle et al., 2004), Bayesian inference (Cootes et al., 2000), deformable templates (Miao et al., 1999), graph-matching (Leung et al., 1995), skin color learning (Hsu et al., 2001; Sahbi and Boujemaa, 2000), and more rapid techniques such as boosting a cascade of classifiers (Viola and Jones, 2001; Li and Zhang, 2004; Wu et al., 2005; Socolinsky et al., 2003; Elad et al., 2002) and hierarchical coarse-to-fine processing (Fleuret and Geman, 2001).

The face hierarchy was described in Section 3. We now introduce a specific model for (1), the probability of a chain under the background hypothesis, and finally the solution to the resulting instance of the constrained optimization problem expressed in (8) below. The probability model links the cost of the SVMs to their underlying level of invariance and selectivity. Afterwords, in Section 8, we illustrate the performance of the designed g-network in terms of speed and error on both simple and challenging face databases including the CMU and the MIT datasets.

7.1 Chain Model

Our model family is $\{\delta(l;\mathbf{n},\beta),\beta\in B\}$, where $\delta(l;\mathbf{n},\beta)$ is the probability of a chain of "ones" of depth $l-1$. These probabilities are determined by the conditional probabilities in (2). Denote these by

$$\delta(1;\mathbf{n},\beta)=P_0(g_{root}>0)$$

and

$$\delta(l\mid 1,...,l-1;\mathbf{n},\beta)=P_0(g_t>0|g_s>0,s\in\mathcal{A}_t).$$

Specifically, we take:

$$\begin{aligned}\delta(1;\mathbf{n},\beta) &= \frac{1}{\beta_1 n_1} && \beta_1>0\\ \delta(l\mid 1,...,l-1;\mathbf{n},\beta) &= \frac{\beta_1 n_1+...+\beta_{l-1}n_{l-1}}{\beta_1 n_1+...+\beta_{l-1}n_{l-1}+\beta_l n_l} && ,\quad \beta_1,...,\beta_l>0.\end{aligned}$$

Loosely speaking, the coefficients $\beta=\{\beta_j \quad j=1,...,L\}$ are inversely proportional to the degree of "pose invariance" expected from the SVMs at different levels. At the upper, highly invariant, levels l of the g-network, minimizing computation yields relatively small values of β_l and vice-versa at the lower, pose-dedicated, levels. The motivation for this functional form is that the conditional false alarm rate $\delta(l\mid 1,...,l-1;\mathbf{n},\beta)$ should be *increasing* as the number of support vectors in the upstream levels $1,...,l-1$ increases. Indeed, when $g_s>0$ for all nodes s upstream of node t, and when these SVMs have a large number of support vectors and hence are very selective, the background patterns reaching node t resemble faces very closely and are likely to be accepted by the test at t. Of course, fixing the numbers of support vectors upstream, the conditional selectivity (that is, one minus the false positive error rate) at level l grows with n_l. *Notice also that the model does not anticipate exponential decay, corresponding to independent tests (under P_0) along the branches of the hierarchy.*

Using the marginal and the conditional probabilities expressed above, the probability $\delta(l;\mathbf{n},\beta)$ to have a chain of ones from the root cell to any particular cell at level l is easily computed:

$$\delta(l;\mathbf{n},\beta) = \frac{1}{\beta_1 n_1}\,\frac{\beta_1\, n_1}{\beta_1\, n_1+\beta_2\, n_2}...\frac{\beta_1\, n_1+...+\beta_{l-1}n_{l-1}}{\beta_1 n_1+...+\beta_l n_l}$$

$$= \left(\sum_{j=1}^{l} \beta_j n_j \right)^{-1}. \tag{7}$$

Clearly, for any $\mathbf{n}$ and β, these probabilities decrease as l increases.

7.2 The Optimization Problem

Using (7), the constrained minimization problem (5) becomes:

$$\begin{array}{ll} \min\limits_{n_1,\ldots,n_L} & \left[n_1 + \sum\limits_{l=2}^{L} \left(\sum\limits_{i=1}^{l-1} \beta_i n_i \right)^{-1} \nu_l \; n_l \right] \\ \text{s.t.} & \left\{ \begin{array}{l} \nu_L \left(\sum\limits_{i=1}^{L} \beta_i n_i \right)^{-1} \leq \mu \\ 0 < n_l \leq N_l. \end{array} \right. \end{array} \tag{8}$$

This problem is solved in two steps:

- **Step I:** Start with the solution for a binary network (i.e., $\nu_{l+1} = 2\nu_l$). This solution is provided in Appendix A.
- **Step II:** Pass to a dyadic network using the solution to the binary network, as shown in Sahbi (2003, p. 127).

7.3 Model Selection

We use a simple function with two degrees of freedom to characterize the growth of $\beta_1, \ldots, \beta_L$:

$$\beta_l = \Psi_1^{-1} \; \exp\{\Psi_2 (l-1)\} \tag{9}$$

where $\Psi = (\Psi_1, \Psi_2)$ are positive. Here Ψ_1 represents the degree of pose invariance at the root cell and Ψ_2 is the rate of the decrease of this invariance. Let $\mathbf{n}^*(\Psi)$ denote the solution to (8) for β given by (9) and suppose we restrict $\Psi \in Q$, a discrete set. (In our experiments, $|Q| = 100$ corresponding to ten choices for each parameter Ψ_1 and Ψ_2, namely Ψ_1 ranges from 0.0125 to 0.2 and Ψ_2 ranges from 0.1 to 1.0, both in equal steps.) In other words, $\mathbf{n}^*(\Psi)$ are the optimal numbers of support vectors found when minimizing total computation (8) under the false positive constraint for a given fixed $\beta = \{\beta_1, \ldots, \beta_L\}$ determined by (9). Then Ψ^* is selected to minimize the discrepancy between the model and empirical conditional false positive rates:

$$\min_{\Psi \in Q} \sum_{l=1}^{L} \left| \delta(l \mid 1, \ldots, l-1; \mathbf{n}^*(\Psi), \Psi) - \hat{\delta}(l \mid 1, \ldots, l-1; \mathbf{n}^*(\Psi)) \right| \tag{10}$$

where $\hat{\delta}(l \mid 1, \ldots, l-1; \mathbf{n}^*(\Psi))$ is the underlying empirical probability of observing $g_t > 0$ given that $g_s > 0$ for all ancestors s of t, averaged over all nodes t at level l, when the g-network is built with $\mathbf{n}^*(\Psi)$ support vectors.

In practice, it takes 3 days (on a 1-Ghz pentium-III) to implement this design, which includes building the f-network (SVM training), solving the minimization problem (8) for different instances

Algorithm: *Design of the g-network.*

- Build the f-network using standard SVM training and learning set $\mathcal{L} \cup \mathcal{B}$
- **for** $(\Psi_1, \Psi_2) \in Q$ **do**
  - $\beta_l \leftarrow \Psi_1^{-1} \exp\{\Psi_2(l-1)\}$, $l = 1, ..., L$
  - Compute $\mathbf{n}^*(\Psi)$ using (8).
  - Build the g-network using the reduced set method and the specified costs $\mathbf{n}^*(\beta)$.
  - Compute the model and empirical conditional probabilities $\delta(l \mid 1, ..., l-1; \mathbf{n}^*(\Psi), \Psi)$ and $\hat{\delta}(l \mid 1, ..., l-1; \mathbf{n}^*(\Psi))$.

end

- $\Psi^* \leftarrow$ (10)
- The specification for the g-network is $\mathbf{n}^*(\Psi^*)$

of $\Psi = (\Psi_1, \Psi_2)$ and applying the reduced set technique (6) for each sequence of costs $n^*(\Psi)$ in order to build the g-network.

When solving the constrained minimization problem (8) (cf. Appendix A), we find the optimal numbers $n_1^*, ..., n_6^*$ of support vectors, rounded to the nearest even integer, are given by:

$$\mathbf{n}^* = \{2,\ 2,\ 2,\ 4,\ 8,\ 22\},$$

(cf. table 2), corresponding to $\Psi^* = ((\Psi_1^{-1})^*, \Psi_2^*) = (7.27, 0.55)$, resulting in $\beta^* = \{7.27,\ 25.21,\ 87.39,\ 302.95,\ 525.09,\ 910.11\}$. For example, we estimate

$$P_0(g_{root} > 0) = \frac{1}{2 \times 7.27} = 0.069$$

the false positive rate at the root.

The empirical conditional false alarms were estimated on background patterns taken from 200 images including highly textured areas (flowers, houses, trees, etc.). The conditional false positive rates for the model and the empirical results are quite similar, so that the cost in the objective function (8) approximates effectively the observed cost. In fact, when evaluating the objective function in (8), the average cost was 3.379 kernel evaluations per pattern whereas in practice this average cost was 3.196 per pattern taken from scenes including highly textured areas.

Again, the coefficients β_l^* and the complexity n_l^* of the SVM classifiers are increasing as we go down the hierarchy, which demonstrates that the best architecture of the g-network is low-to-high in complexity.

7.4 Features and Parameters

Many factors intervene in fitting our cost/error model to real observations (the conditional false alarms), including the size of the training sets and the choice of features, kernels and other parameters, such as the bound on the expected number of false alarms. Obviously the nature of the resulting g-network can be sensitive to variations of these factors. We have only used wavelet features, the Gaussian kernel, selecting the parameters by cross-validation, and the very small ORL database.

Whereas we have not done systematic experiments to analyze sensitivity, it is reasonable to suppose that having more data or more powerful features would increase performance. For instance,

Ψ_1	Ψ_2	L_1 error	Number of support vectors per level					
	$\Psi_2 = .70$	0.980	1.03	1.08	1.42	1.87	4.98	16.375
$\Psi_1 = .2000$	$\Psi_2 = .55$	0.809	1.49	2.35	3.69	4.99	11.64	32.12
	$\Psi_2 = .30$	1.207	2.70	8.13	17.45	24.36	42.89	93.50
	$\Psi_2 = .70$	1.129	1.00	1.08	1.46	1.98	5.40	18.12
$\Psi_1 = .1875$	$\Psi_2 = .55$	0.750	1.39	2.20	3.46	4.68	10.91	30.12
	$\Psi_2 = .30$	1.367	2.53	7.62	16.36	22.83	40.20	87.62
	$\Psi_2 = .70$	0.995	1.00	1.18	1.70	2.46	7.20	25.50
$\Psi_1 = .1750$	$\Psi_2 = .55$	0.766	1.30	2.05	3.23	4.37	10.19	28.12
	$\Psi_2 = .30$	1.401	2.36	7.12	15.27	21.32	37.56	81.87
	$\Psi_2 = .70$	0.981	1.00	1.29	2.00	3.13	9.84	37.00
$\Psi_1 = .1625$	$\Psi_2 = .55$	0.752	1.21	1.91	3.00	4.06	9.46	26.12
	$\Psi_2 = .30$	1.428	2.19	6.61	14.18	19.80	34.86	76.00
	$\Psi_2 = .70$	0.839	1.00	1.41	2.38	4.03	13.74	55.12
$\Psi_1 = .1500$	$\Psi_2 = .55$	0.605	1.11	1.76	2.77	3.75	8.74	24.12
	$\Psi_2 = .30$	1.339	2.02	6.10	13.09	18.27	32.17	70.12
	$\Psi_2 = .70$	1.137	1.00	1.56	2.87	5.30	19.72	85.12
Ψ_1=.1375	**Ψ_2=.55**	**0.557**	**1.02**	**1.61**	**2.54**	**3.43**	**8.01**	**22.12**
	$\Psi_2 = .30$	1.580	1.85	5.59	11.99	16.74	29.47	64.25
	$\Psi_2 = .70$	1.230	1.00	1.75	3.53	7.16	29.29	137.12
$\Psi_1 = .1250$	$\Psi_2 = .55$	0.808	1.00	1.71	2.89	4.21	10.56	30.62
	$\Psi_2 = .30$	1.441	1.69	5.08	10.91	15.23	26.83	58.50
	$\Psi_2 = .70$	NS	-	-	-	-	-	-
$\Psi_1 = .1000$	$\Psi_2 = .55$	0.958	1.00	2.21	4.69	8.60	27.23	93.37
	$\Psi_2 = .30$	0.687	1.35	4.06	8.72	12.18	21.44	46.75
	$\Psi_2 = .70$	NS	-	-	-	-	-	-
$\Psi_1 = .0750$	$\Psi_2 = .55$	NS	-	-	-	-	-	-
	$\Psi_2 = .30$	1.242	1.01	3.05	6.55	9.14	16.11	35.12

Table 2: A sample of the simulation results. Shown, for selected values of (Ψ_1, Ψ_2), are the L_1 error in (10) and also the numbers of support vectors which minimize cost. In practice 10×10 possible values of Ψ_1 and Ψ_2 are considered ($\Psi_1 \in [0, 0.2]$ and $\Psi_2 \in [0.1, 1.0]$). (*NS* stands for "no solution", L_1 refers to the sum of absolute differences, and the bold line is the optimal solution.)

with highly discriminating features, the separation between the positive and negative examples used for training the f-network might be sufficient to allow even linear SVMs to produce accurate decision boundaries, in which case very few support vectors might be required in the f-network. The leave-one-out error bound (see for instance Vapnik, 1998) would then suggest low error rates. Accordingly, in principle, the g-network could be designed with few (virtual) support vectors while satisfying the false alarm bound in (3). The features we use – the Haar wavelet coefficients – are generic and not especially powerful. The only other ones we tried were Daubechies wavelets, which

were abandoned due to extensive computation; their performance is unknown. Similar arguments apply to the choice of kernels and their parameters; for instance the scale of the Gaussian kernel controls influences both the error rate and the number of support vectors in the f-network (and also in the g-network.)

8. Experiments

All the training images of faces are based on the Olivetti database of 400 gray level pictures – ten frontal images for each of forty individuals. The coordinates of the eyes and the mouth of each picture were labeled manually. Most other methods (see below) use a far larger training set, in fact, usually ten to one hundred times larger. In our view, the smaller the better in the sense that the number of examples is a measure of performance along with speed and accuracy. Nonetheless, this criterion is rarely taken into account in the literature on face detection (and more generally in machine learning).

In order to sample the pose variation within Λ_t, for each face image in the original Olivetti database, we synthesize 20 images of 64×64 pixels with randomly chosen poses in Λ_t. Thus, a set of $8,000$ faces is synthesized for each pose cell in the hierarchy. Background information is collected from a set of $1,000$ images taken from 28 different topical databases (including auto-racing, beaches, guitars, paintings, shirts, telephones, computers, animals, flowers, houses, tennis, trees and watches), from which $50,000$ subimages of 64×64 pixels are randomly extracted.

Given coarse-to-fine search, the "right" alternative hypothesis at a node is "path-dependent". That is, the appropriate "negative" examples to train against at a given node are those data points which pass all the tests from the root to the parent of the node. As with cascades, this is what we do in practice; more precisely, we merge a fixed collection of background images with a "path-dependent" set (for details see Sahbi, 2003, chap. 4).

Each subimage, either a face or background, is encoded using the 16×16 low frequency coefficients of the Haar wavelet transform computed efficiently using the integral image (Sahbi, 2003; Viola and Jones, 2001). Thus, only the coefficients of the third layer of the wavelet transform are used; see Chapter 2 of Sahbi (2003). The set of face and background patterns belonging to Λ_t are used to train the underlying SVM f_t in the f-network (using a Gaussian kernel).

8.1 Clustering Detections

Generally, a face will be detected at several poses; similarly, false positives will often be found in small clusters. In fact, every method faces the problem of clustering detections in order to provide a reasonable estimate of the "false alarm rate," rendering comparisons somewhat difficult.

The search protocol was described in Section 7.1. It results in a set of detections $\mathbf{D}_g$ for each non-overlapping 16×16 block in the original image and each such block in each of three downsampled images (to detect larger faces). All these detections are initially collected. Evidently, there are many instances of two "nearby" poses which cannot belong to two distinct, fully visible faces. Many ad hoc methods have been designed to ameliorate this problem. We use one such method adapted to our situation: For each hierarchy, we sum the responses of the SVMs at the leaves of each complete chain (i.e., each detection in $\mathbf{D}_g$) and remove all the detections from the aggregated list unless this sum exceeds a learned threshold τ, in which case $\mathbf{D}_g$ is represented by a single "average" pose. In other words, we declare that a block contains the location of a face if the *aggregate SVM score* of the classifiers in the leaf-cells of complete chains is above τ. In this way, the false negative rate does

not increase due to pruning and yet some false positives are removed. Incompatible detections can and do remain.

Note: One can also implement a "voting" procedure to arbitrate among such remaining but incompatible detections. This will further reduce the false positive rate but at the expense of some missed detections. We shall not report those results; additional details can be found in Sahbi (2003). Our main intention is to illustrate the performance of the g-network on a real pattern recognition problem rather than to provide a detailed study of face detection or to optimize our error rates.

8.2 Evaluation

We evaluated the g-network in term of precision and run-time in several large scale experiments involving still images, video frames (TF1) and standard datasets of varying difficulty, including the CMU+MIT image set; some of these are extremely challenging. All our experiments were run under a 1-Ghz pentium-III mono-processor containing a 256 MB SDRM memory, which is today a standard machine in digital image processing.

The Receiver Operator Characteristic (ROC) curve is a standard evaluation mechanism in machine perception, generated by varying some free parameter (e.g., a threshold) in order to investigate the trade-off between false positives and false negatives. In our case, this parameter is the threshold τ for the aggregate SVM score of complete chains discussed in previous section. Several points on the ROC curve are given for the TF1 and CMU+MIT test sets whereas only a single point is reported for easy databases (such as FERET).

8.2.1 FERET AND TF1 DATASETS

The FERET database (FA and FB combined) contains $3,280$ images of single and frontal views of faces. It is not very difficult: The detection rate is 98.8 % with 245 false alarms and examples are shown in the top of Fig 10. The average run time on this set using a 1Ghz is 0.28 (s) for images of size 256×384.

The TF1 corpus involves a News-video stream of 50 minutes broadcasted by the French TV channel TF1 on May 5th, 2002. (It was used for a video segmentation and annotation project at INRIA and is not publicly available.) We sample the video at one frame each $4(s)$, resulting into 750 good quality images containing 1077 faces. Some results are shown on the bottom of Fig 10 and the performance is described in Table 8.2.1 for three points on the ROC curve. The *false alarm rate* is the total number of false detections divided by the total number of hierarchies traversed, that is, the total number of 16×16 blocks visited in processing the entire database.

8.2.2 ARF DATABASE AND SENSITIVITY ANALYSIS

The full ARF database contains 4000 images on ten DVDs; eight of these DVDs – $3,200$ images with faces of 100 individuals against uniform backgrounds – are publicly available at (http://rvl1.ecn.purdue.edu/$\sim$aleix/aleix_face_DB.html). This dataset is still very challenging due to large differences in expression and lighting, and especially to partial occlusions due to scarves, sunglasses, etc. The g-network was run on this set; sample results are given in the rows 2-4 of Fig 10. Among the 10 face images for a given person, two images show the person with sunglasses, three with scarves and five with some variation in the facial expression and/or strong lighting effects. Our face detection rate is only 78.79 % with 3 false alarms. Among the missed faces, 32.12 %

Figure 10: Sample detections on three databases: FERET (top), ARF (middle), TF1 (bottom).

are due to occlusion of the mouth, 56 % due to occlusion of the eyes (presence of sun glasses) and 11.88 % due to face shape variation and lighting effects.

8.2.3 CMU+MIT DATASET

The CMU subset contains frontal (upright and in-plane rotated) faces whereas the MIT subset contains lower quality face images. Images with an in-plane rotation of more than 20^0 were removed, as well as "half-profile" faces in which the nose covers one cheek. This results in a subset of 141 images from the CMU database and 23 images from the MIT test set. These 164 images contain 556 faces. A smaller subset was considered in Rowley et al. (1998) and in Viola and Jones (2001),

Threshold	# Missed faces	Detection rate	# False alarms	False alarm rate	Average run-time
$\tau = 0$	017	98.4 %	333	1/9,632	0.351(s)
$\tau = 1$	109	89.8 %	143	1/22,430	0.357(s)
$\tau = 2$	151	85.9 %	096	1/33,411	0.343(s)

Table 3: Performance on the TF1 database of 750 frames with 1077 faces. The three rows correspond to three choices of the threshold for clustering detections. The false alarm rate is given as the number of background pattern declared as faces over the total number of background patterns. The average run-time is reported for this corpus on images of size 500×409.

namely 130 images containing 507 faces, although in the former study other subsets, some accounting for half-profiles, were also considered (see Table 4).

	Sahbi & Geman	Viola and Jones (2001)	Rowley et al. (1998)
# of images	164	130	130
# of faces	556	507	507
False alarms	112	95	95
Detection rate	89.61 %	90.8 %	89.2 %
Time (384×288)	$0.20(s)$	$\frac{1}{3} \times 0.20(s)$	$5 \times 0.20(s)$

Table 4: Comparison of our work with other methods which achieve high performance.

The results are given in Table 4. The g-network achieves a detection rate of 89.61 % with 112 false alarms on the 164 images. These results are very comparable to those in Rowley et al. (1998); Viola and Jones (2001): for 95 false alarms, the detection rate in Viola and Jones (2001) was 90.8 % and in Rowley et al. (1998) it was 89.2 %. Put another way, we have an equivalent number of false alarms with a larger test set but a slightly smaller detection rate; see Table 4. Our performance could very likely be improved by utilizing a larger training set, exhibiting more variation than the Olivetti set, as in Viola and Jones (2001); Rowley et al. (1998); Schneiderman and Kanade (2000), where training sets of sizes 4916, 1048 and 991 images, respectively, are used.

Scenes are processed efficiently; see Fig 11. The run-time depends mainly on the size of the image and its complexity (number of faces, presence of face-like structures, texture, etc). Our system processes an image of 384×288 pixels (the dimensions reported in cited work) in 0.20(s); this is an average obtained by measuring the total run time on a sample of twenty images of varying sizes and computing the equivalent number of images (approximately 68) of size 384×288. This average is about three times slower than in Viola and Jones (2001), approximately five times faster than the fast version in Rowley et al. (1998) and 200 times faster than in Schneiderman and Kanade (2000). Notice that, for tilted faces, the fast version of Rowley's detector spends 14(s) on images of 320×240 pixels.

Figure 11: Detections using the CMU+MIT test set. More results can be found on http://www-rocq.inria.fr/who/Hichem.Sahbi/Web/face_results/

Threshold	Detection rate	# False alarms	False alarms rate
$\tau = 0$	92.95 %	312	1/2,157
$\tau = 0.5$	89.61 %	112	1/6,011
$\tau = 1$	87.2 %	096	1/7,013
$\tau = 10$	34.94 %	004	1/168,315

Table 5: Evaluation of our face detector on the CMU+MIT databases.

8.2.4 HALLUCINATIONS IN TEXTURE

Performance degrades somewhat on highly texture scenes. Some examples are provided in Fig 12. Many features are detected, triggering false positives. However, there does not appear to an "explosion" of hallucinated faces, at least not among the roughly 100 such scenes we processed, of which only a few had order ten detections (two of these are shown in Fig 12).

9. Summary

We presented a general method for exploring a space of hypotheses based on a coarse-to-fine hierarchy of SVM classifiers and applied it to the special case of detecting faces in cluttered images. As opposed to a single SVM dedicated to a template, or even a hierarchical platform for coarse-to-fine template-matching, but with no restrictions on the individual classifiers (the f-network), the proposed framework (the g-network) allows one to achieve a desired balance between computation and error. This is accomplished by controlling the number of support vectors for each SVM in the hierarchy; we used the reduced set technique here, but other methods could be envisioned. The design of the network is based on a model which accounts for cost, selectivity and invariance. Naturally, this requires assumptions about the cost of an SVM and the probability that any given SVM will be evaluated during the search.

We used one particular statistical model for the likelihood of a background pattern reaching a given node in the hierarchy, and one type of error constraint, but many others could be considered. In particular, the model we used is not realistic when the likelihood of an "object" hypothesis becomes comparable with that of the "background" hypothesis. This is in fact the case at deep levels of the hierarchy, at which point the conditional selectivity of the classifiers should ideally be calculated with respect to *both* object and background probabilities. A more theoretical approach to these issues, especially the cost/selectivity/invariance tradeoff, can be found in Blanchard and Geman (2005), including conditions under which coarse-to-fine search is optimal.

Extensive experiments on face detection demonstrate the huge gain in efficiency relative to either a dedicated SVM or an unrestricted hierarchy of SVMs, while at the same time maintaining precision. Efficiency is due to both the coarse-to-fine nature of scene processing, rejecting most background regions very quickly with highly invariant SVMs, *and* to the relatively low cost of most of the SVMs which are ever evaluated.

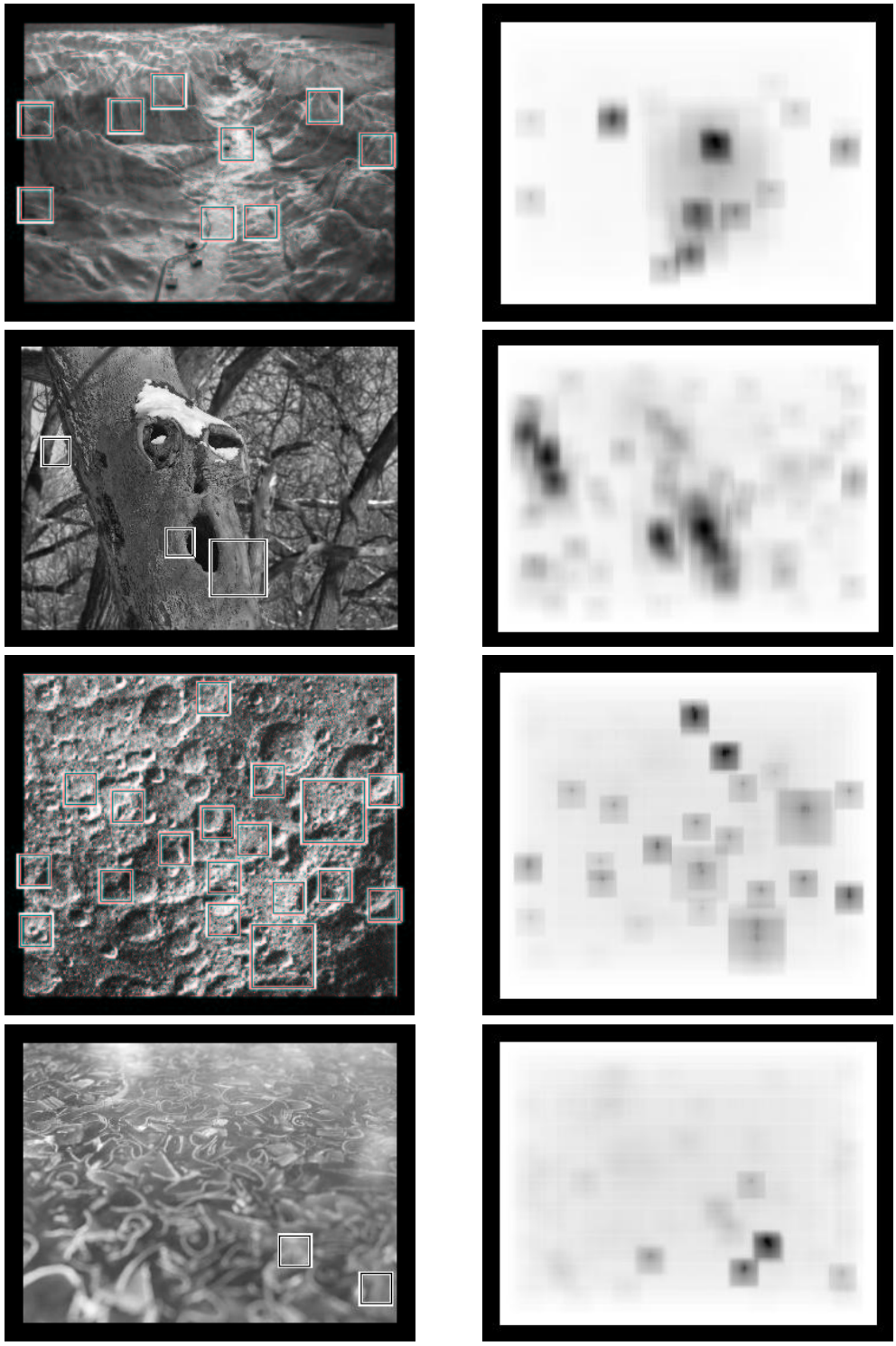


Figure 12: Left: Detections on highly textured scenes. (We thank Larry Jackal for the second ("face in a tree") image.) Right: The darkness of a pixel is proportional to the amount of local processing necessary to collect all detections. The average number of kernel evaluations, per block visited, are respectively 11, 6, 14 and 4.

Acknowledgments

The authors are grateful to both referees for very thoughtful and comprehensive reviews, which represent the reviewing process at its best. The research of the second author was supported in part by NSF grants ITR 0219016 and ITR 0427223 and by ARO grant DAAD19-02-1-0337.

Appendix A.

In this appendix, we will show how an approximate solution of the constrained minimization problem (8) can be obtained for the case of a binary hierarchy (i.e., $\nu_l = 2^{l-1}$). An extension to any arbitrary hierarchy can be found in Sahbi (2003).

Suppose β is fixed and consider the optimization problem in (8). Clearly, the *unconstrained* problem is degenerate, minimized by choosing $n_l \equiv 0$; indeed this minimizes cost. We start by minimizing cost for a *fixed value* of n_L and for real-valued $n_l, l = 1,...,n_{L-1}$. In this case, the values of $n_1, n_2, ..., n_{L-1}$ which satisfy $\frac{\partial \mathcal{C}}{\partial n_l} = 0$, $l = 1,...,L-1$, are given by:

$$n_l = \begin{cases} \left(2^{\frac{L(L-1)}{2}} \left(\prod_{i=1}^{L-1} \beta_i \right)^{-1} n_L \right)^{1/L} & \text{if } l = 1 \\ 2^{-\frac{l(l-1)}{2}} \left(\prod_{i=1}^{l-1} \beta_i \right) \beta_l^{-1} \, n_1^{l-1} \, (\beta_l n_1 - 2^{l-1}) & l \in \{2,...,L-1\} \\ n_L & l = L. \end{cases} \tag{11}$$

The proof is straightforward:

$$\begin{aligned} \frac{\partial \mathcal{C}}{\partial n_1} &= 1 - \sum_{l=2}^{L} \left[\frac{\beta_1 2^{l-1} n_l}{(\sum_{i=1}^{l-1} \beta_i n_i)^2} \right] \\ \frac{\partial \mathcal{C}}{\partial n_j} &= \frac{2^{j-1}}{(\sum_{i=1}^{j-1} \beta_i n_i)} - \sum_{l=j+1}^{L} \left[\frac{\beta_j 2^{l-1} n_l}{(\sum_{i=1}^{l-1} \beta_i n_i)^2} \right], j \in \{2, L-1\}. \end{aligned} \tag{12}$$

We have:

$$\frac{\partial \mathcal{C}}{\partial n_{j+1}} = 0 \quad \Rightarrow \quad \sum_{l=j+2}^{L} \left[\frac{2^{l-1} n_l}{(\sum_{i=1}^{l-1} \beta_i n_i)^2} \right] = \frac{2^j}{(\sum_{i=1}^{j} \beta_i n_i)} \frac{1}{\beta_{j+1}}$$

$$\frac{\partial \mathcal{C}}{\partial n_j} = 0 \quad \Rightarrow \quad \frac{2^{j-1}}{(\sum_{i=1}^{j-1} \beta_i n_i)} - \beta_j \frac{2^j n_{j+1}}{(\sum_{i=1}^{j} \beta_i n_i)^2} - \beta_j \sum_{l=j+2}^{L} \left[\frac{2^{l-1} n_l}{(\sum_{i=1}^{l-1} \beta_i n_i)^2} \right] = 0.$$

The above two equations imply:

$$\frac{2^{j-1}}{(\sum_{i=1}^{j-1}\beta_i n_i)} - \beta_j \frac{2^j\, n_{j+1}}{(\sum_{i=1}^{j}\beta_i n_i)^2} - \beta_j \frac{2^j}{\beta_{j+1}\,(\sum_{i=1}^{j}\beta_i n_i)} = 0,\ j \neq 1. \tag{13}$$

Suppose n_1 is known; we show by a recursion that the general term n_l is given by (11): Combining $\frac{\partial \mathcal{C}}{\partial n_1} = 0$ and $\frac{\partial \mathcal{C}}{\partial n_2} = 0$, we obtain:

$$1 - \frac{\beta_1\, 2\, n_2}{\beta_1^2\, n_1^2} - \frac{2\,\beta_1}{\beta_2\,\beta_1\, n_1} = 0 \quad \Rightarrow \quad n_2 = \frac{1}{2}\,\frac{\beta_1}{\beta_2}\, n_1\,(\beta_2 n_1 - 2).$$

Assume for $2 \leq j \leq l (l \in \{2, L-2\})$

$$n_j = 2^{-\frac{j(j-1)}{2}} \left(\prod_{i=1}^{j-1}\beta_i\right) \beta_j^{-1}\, n_1^{j-1}\,(\beta_j n_1 - 2^{j-1}). \tag{14}$$

We now demonstrate that:

$$n_{l+1} = 2^{-\frac{(l+1)l}{2}} \left(\prod_{i=1}^{l}\beta_i\right) \beta_{l+1}^{-1}\, n_1^{l}\,(\beta_{l+1} n_1 - 2^{l}). \tag{15}$$

By (14), $\forall j \in \{2, ..., l\}$, we have:

$$\begin{aligned}\left(\sum_{i=1}^{j}\beta_i n_i\right) &= \beta_1 n_1 + \beta_2\,\frac{1}{2}\,\beta_1\beta_2^{-1}\, n_1\,(\beta_2 n_1 - 2) + \beta_3\,\frac{1}{8}\,\beta_1\beta_2\beta_3^{-1}\, n_1^2\,(\beta_3 n_1 - 4) + ... \\ &+ \beta_{j-1}\left(2^{-\frac{(j-1)(j-2)}{2}}\right)\left(\prod_{i=1}^{j-2}\beta_i\right)\beta_{j-1}^{-1}\, n_1^{j-2}\,\left(\beta_{j-1} n_1 - 2^{j-2}\right) \\ &+ \beta_j\left(2^{-\frac{j(j-1)}{2}}\right)\left(\prod_{i=1}^{j-1}\beta_i\right)\beta_j^{-1}\, n_1^{j-1}\left(\beta_j n_1 - 2^{j-1}\right).\end{aligned}$$

Hence, $\forall j \in \{2, ..., l\}$:

$$\left(\sum_{i=1}^{j}\beta_i n_i\right) = 2^{-\frac{j(j-1)}{2}} \left(\prod_{i=1}^{j}\beta_i\right) n_1^j. \tag{16}$$

Let $\pi_j = \prod_{i=1}^{j}\beta_i$. Using (16) and for $j = l$, we rewrite (13) as:

$$\frac{2^{l-1}}{2^{-\frac{(l-1)(l-2)}{2}}\,\pi_{l-1}\, n_1^{l-1}} - \beta_l \frac{2^l n_{l+1}}{\left(2^{-\frac{l(l-1)}{2}}\,\pi_l\, n_1^l\right)^2} - \frac{\beta_l}{\beta_{l+1}}\,\frac{2^l}{2^{-\frac{l(l-1)}{2}}\,\pi_l\, n_1^l} = 0$$

$$\Rightarrow \quad n_{l+1} = 2^{-\frac{(l)(l+1)}{2}}\,\pi_l\,\beta_{l+1}^{-1}\, n_1^l\,(\beta_{l+1} n_1 - 2^l)$$

which proves (15). As for n_1, using (12) for $j = L-1$,

$$\frac{\partial \mathcal{C}}{\partial n_{L-1}} = 0 \quad \Rightarrow \quad n_1 = \left(\left(\frac{1}{\pi_{L-1}} \right) 2^{L(L-1)/2} \, n_L \right)^{1/L} \quad \square$$

We can now rewrite (8) as:

$$\min_{n_L} L \left(\frac{1}{\pi_{L-1}} 2^{L(L-1)/2} \, n_L \right)^{1/L} - \sum_{l=2}^{L} \left(\frac{2^{l-1}}{\beta_l} \right)$$
$$\text{s.t.} \quad \begin{cases} \nu_L \left(\sum_{i=1}^{L} \beta_i n_i \right)^{-1} \leq \mu \\ 0 < n_l \leq N_l. \end{cases}$$

Using (11), this can be written entirely in terms of n_L. We use a "generate-and-test" (brute-force search) strategy: First, the parameter n_L is varied from 1 to its upper bound N_L (with some quantization). Then, for each value of this parameter, we check the consistency of the candidate solution, that is, whether the first constraint (on expected false alarms) is satisfied and whether each n_l is bounded N_l. The value n_L minimizing the cost function is retained.

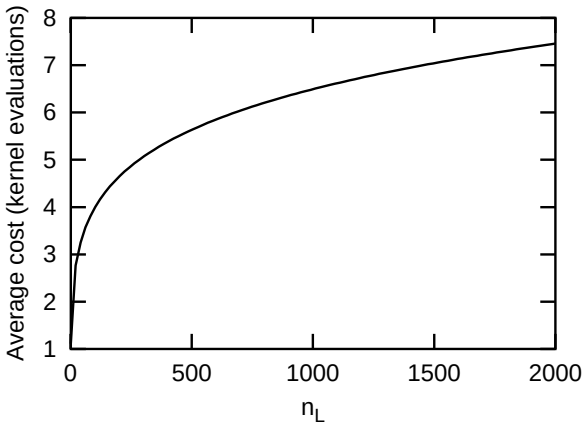


Figure 13: The average cost $\mathcal{C}(n_1, ..., n_L)$ is an increasing function of n_L.

For small values of n_L, the objective function in (8) (the average cost) typically takes small values (cf. Fig 13) and the upper bound constraints related to $\{n_l\}$ are generally satisfied, but the mean false alarm constraint might not be satisfied. For large values of n_L, the bounds on $\{n_l\}$ might not be satisfied and the average cost increases, although the mean false alarm constraint is typically satisfied.

Finally, we allow β to vary with Ψ according to (9). Notice that (8) might not have a solution for any Ψ; obviously we only consider values for which the constraints are satisfied for some n_L.

References

Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation.*, 11(7): 1691–1715, 1999.

Y. Amit, D. Geman, and X. Fan. A coarse-to-fine strategy for multi-class shape detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12):1606–1621, 2004.

S. Baker and S. Nayar. Pattern rejection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 544–549, 1996.

R. Battiti and C. Koch. Computing optical flow across multiple scales: a coarse-to-fine approach. *International Journal of Computer Vision*, 6(2):133–145, 1991.

G. Blanchard and D. Geman. Sequential testing designs for pattern recognition. *Annals of Statistics*, 33:1155–1202, 2005.

G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:849–865, 1988.

B. E. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Workshop on Computational Learning Theory.*, pages 144–152, 1992.

C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Proceedings of the Advances in Neural Information Processing Systems*, volume 9, pages 375–381. The MIT Press, 1997.

C.J.C. Burges. Simplified support vector decision rules. In *Proceedings of the International Conference on Machine Learning*, pages 71–77, 1996.

T. Cootes, K. Walker, and C. Taylor. View-based active appearance models. In *Proceedings of the IEEE International Conference on Face and Gesture Recognition.*, pages 227–232, 2000.

M. Elad, Y. Hel-Or, and R. Keshet. Pattern detection using a maximal rejection classifier. *Pattern Recognition Letters*, 23(12):1459–1471, 2002.

C.K. Eveland, D.A. Socolinsky, C.E. Priebe, and D.J. Marchette. A hierarchical methodology for class detection problems with skewed priors. *Journal of Classification*, 22:17–48, 2005.

T. Evgeniou, M. Pontil, C. Papageorgiou, and T. Poggio. Image representations for object detection using kernel classifiers. In *Proceedings of the Asian Conference on Computer Vision*, pages 687–692, 2000.

X. Fan. *Learning a hierarchy of classifiers for multi-class shape detection*. PhD thesis, Johns Hopkins University, Department of Electrical Engineering, 2006.

R. Féraud, O.J. Bernier, J.E. Viallet, and M. Collobert. A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):42–53, 2001.

F. Fleuret. *Détection hiérarchique de visages par apprentissage statistique*. PhD thesis, University of Paris VI, Jussieu., 1999.

F. Fleuret and D. Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(2):85–107, 2001.

S. Gangaputra and D. Geman. A design principle for coarse-to-fine classification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1877–1884, 2006a.

S. Gangaputra and D. Geman. The trace model for object detection and tracking. In *Proceedings of Workshop on Object Recognition, Taromina, Sicily, 2004, Lecture Notes in Computer Science*, 2006b.

C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004.

J. C. Gee and D. R. Haynor. Rapid coarse-to-fine matching using scale-specific priors. In *Proceedings of the SPIE Medical Imaging: Image Processing, M. H. Loew and K. M. Hanson, eds., Bellingham, WA:SPIE*, pages 416–427, 1996.

S. Geman, K. Manbeck, and D.E. McClure. Coarse-to-fine search and rank-sum statistics in object recognition. Technical report, Brown University, 1995.

B. Heisele, T. Serre, S. Mukherjee, and T. Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 18–24, 2001.

R.L. Hsu, M. Abdel-Mottaleb, and A. K. Jain. Face detection in color images. In *Proceedings of the IEEE International Conference on Image Processing*, pages 1046–1049, 2001.

D. P. Huttenlocher and W.J. Rucklidge. A multi-resolution technique for comparing images using the hausdorff distance. In *Proceedings of the Conference on Computer Vision and Pattern Recogntion*, pages 705–706, 1993.

F. Jung. *Reconnaissance d'objets par focalisation et détection de changements*. PhD thesis, Ecole Polytechnique, Paris, France, 2001.

T. Kanade. *Picture processing system by computer complex and recognition of human faces*. PhD thesis, Kyoto University, Department of Information Science, 1977.

D. Keren, M. Osadchy, and C. Gotsman. Antifaces: A novel, fast method for image detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):747–761, 2001.

W. Kienzle, G.H. Bakir, M. Franz, and B. Schölkopf. Face detection - efficient and rank deficient. In *Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems*, 2004.

T. Leung, M.C. Burl, and P Perona. Finding faces in cluttered scenes using random labelled graph matching. In *Proceedings of the International Conference on Computer Vision*, pages 637–644, 1995.

S.Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1112–1123, 2004.

J. Miao, B. Yin, K. Wang, L. Shen, and X. Chen. A hierarchical multiscale and multiangle system for human face detection in complex background using gravity center template. *Pattern Recognition*, 32(7):1237–1248, 1999.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Computationally efficient face detection. In *Proceedings of the International Conference on Computer Vision*, pages 695–700, 2001.

H. Rowley. *Neural network-based face detection*. PhD thesis, Carnegie Mellon University, 1999.

H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

H. Sahbi. *Coarse-to-fine support vector machines for hierarchical face detection*. PhD thesis, TU-798, Versailles University, 2003.

H. Sahbi and N. Boujemaa. Coarse-to-fine skin and face detection. In *Proceedings of the ACM International Conference on Multimedia.*, pages 432–434, 2000.

H. Sahbi, D. Geman, and N. Boujemaa. Face detection using coarse-to-fine support vector classifiers. In *Proceedings of the IEEE International Conference on Image Processing.*, pages 925–928, 2002.

H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 746–752, 2000.

B. Schölkopf, P. Knirsch, A. Smola, and C. Burges. Fast approximation of support vector kernel expansions and an interpretation of clustering as approximation in feature spaces. In *Proceedings of the Levi M. Schanz R.-J. Ahlers and F. May editors, Mustererkennung DAGM-Symposium Informatik aktuell*, pages 124–132, 1998.

J. Sobottka and I. Pittas. Segmentation and tracking of faces in color images. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition.*, pages 236–241, 1996.

D.A. Socolinsky, Neuheisel, C.E. Priebe, D.J. Marchette, and J. DeVinney. A boosted cccd classifier for fast face detection. *Computing Science and Statistics*, 35, 2003.

K-K. Sung. *Learning and example selection for object and pattern detection,*. PhD thesis, Massachusetts Institute of Technology, Electrical Engineering and Computer Science, 1996.

V.N. Vapnik. Statistical learning theory. *A Wiley-Interscience Publication*, 1998.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

J. Wu, M.D. Mullin, and J.M. Rehg. Linear asymmetric classifier for cascade detectors. In *Proceedings of the International Conference on Machine Learning*, pages 988–995, 2005.

M.H. Yang, D. Roth, and N. Ahuja. A snow-based face detector. In *Proceedings of Advances in Neural Information Processing Systems 12*, pages 855–861, 2000.

Journal of Machine Learning Research 7 (2006) 2125-2148 Submitted 12/05; Revised 5/06; Published10/06

Adaptive Prototype Learning Algorithms: Theoretical and Experimental Studies

Fu Chang FCHANG @IIS.SINICE.EDU.TW
Institute of Information Science
Academia Sinica
Taipei, Taiwan

Chin-Chin Lin ERIKSON@IIS.SINICA.EDU.TW
Department of Electrical Engineering
National Taipei University of Technology
Taipei, Taiwan

Chi-Jen Lu CJLU@IIS.SINICA.EDU.TW
Institute of Information Science
Academia Sinica
Taipei, Taiwan

Editor: Rocco Servedio

Abstract

In this paper, we propose a number of adaptive prototype learning (APL) algorithms. They employ the same algorithmic scheme to determine the number and location of prototypes, but differ in the use of samples or the weighted averages of samples as prototypes, and also in the assumption of distance measures. To understand these algorithms from a theoretical viewpoint, we address their convergence properties, as well as their consistency under certain conditions. We also present a soft version of APL, in which a non-zero training error is allowed in order to enhance the generalization power of the resultant classifier. Applying the proposed algorithms to twelve UCI benchmark data sets, we demonstrate that they outperform many instance-based learning algorithms, the *k*-nearest neighbor rule, and support vector machines in terms of average test accuracy.

Keywords: adaptive prototype learning, cluster-based prototypes, consistency, instance-based prototype, pattern classification

1 Introduction

We divide this section into two parts, with the first part addressing the background of all related methods and the second part discussing our contributions.

1.1 Background

In pattern cognition, one method for classifying objects, expressed as feature vectors, is to compute the distance between the vectors and certain labeled vectors, called prototypes. This approach selects the *k* nearest prototypes for each test object and classifies the object in terms of the labels of the prototypes and a voting mechanism. Prototypes are vectors that reside in the same vector space as feature vectors and can be derived from training samples in various ways. The simplest way is to use all training samples as prototypes (Fix and Hodges, 1951, 1952, 1991a, 1991b). Besides not incurring any training costs, this approach has two major advantages.

First, for a finite set of training samples S, the error rate using all samples as prototypes does not exceed twice the Bayes risk (Cover and Hart, 1967). Second, it ensures consistency, or asymptotic Bayes-risk efficiency (Stone, 1977; Devroye and Györfi, 1985; Zhao, 1987; Devroye et al., 1994).

However, recruiting all training samples as prototypes can incur a high computational cost during the test procedure, which is prohibitive in applications with large corpora. Consequently, certain editing rules have been proposed to reduce the number of prototypes. The condensed nearest neighbor (CNN) rule (Hart, 1968) was the first, and perhaps simplest, proposal among many subsequent ones, all of which try to extract a subset from a collection of samples. These algorithms execute a process iteratively to check the satisfaction of certain criteria for the current set of prototypes, and add or drop prototypes until a stop condition is met. Wilson and Martinez (2000) collected and compared many algorithms of this type (in particular, DROP1 to DROP5), and categorized them as instance-based learning (IBL) algorithms. More recently, an alternative IBL algorithm called the Iterative Case Filtering (ICF) algorithm (Brighton and Mellish, 2002) was proposed. ICF runs faster than most IBL algorithms, which drop rather than add samples (this point is discussed further in Section 7.2), yet it achieves comparable accuracy to the latter algorithms.

Another method for finding prototypes can be categorized as cluster-based learning (CBL) algorithms, in which prototypes are not samples per se, but can be derived as the weighted averages of samples. The *k*-means clustering algorithm (Lloyd, 1982; Max, 1960; Linde et al. 1980), the fuzzy *c*-means algorithm (Bezdek, 1981; Höppner et al., 1999), and the learning vector quantization algorithm (Kohonen, 1988, 1990) are examples of this method. Instead of representing prototypes as the weighted averages of samples, they can be represented as centroids of clusters (Devi and Murty, 2002), or as hyperrectangles (high-dimensional rectangles) (Salzberg, 1991). In the latter case, the distance between a sample and a hyperrectangle not containing the sample is defined as the Euclidean distance between the sample and the nearest face of the hyperrectangle.

In their guidelines for the design of prototype learning algorithms, Devroye et al. (1996, Chapter 19) propose some sufficient conditions for the consistency of this kind of algorithm. The conditions stipulate that: (a) the algorithm should minimize the empirical error, which is the error in classifying training samples; and (b) the number of prototypes should grow as a lower order of the number of training samples.

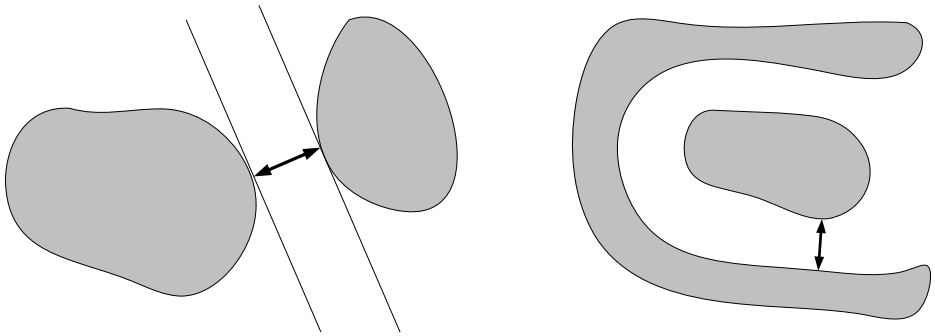
Support vector machines (SVM) can also be used for pattern classification. In this approach, objects are classified by maximizing the margins between samples with different labels, where the margin is defined as the gap between two parallel hyperplanes (Figure 1a). The consistency of SVM is assured if the samples are bounded and the margin between samples with different labels holds (Vapnik, 1995; Schölkopf et al. 1999; Cristianini and Shawe-Taylor, 2000).

1.2 Our Contributions

The requirement that data should be bounded is reasonable, since it is a common practice in applications to normalize feature values to a certain bounded interval (between 0 and 1, for example). The margin assumption, on the other hand, is unique to SVM. However, we can prove the consistency of CNN under a more relaxed assumption (Figure 1b). For convenience, we say that two labeled entities (that is, samples or prototypes) are homogeneous if they have the same label; otherwise, they are heterogeneous. We require a non-zero distance between heterogeneous samples.

Despite its consistency, CNN could be improved in two ways. First, its criterion for prototype satisfaction is rather weak and could be strengthened. Second, it is not difficult to develop an alternative process by using a cluster-based rule to construct prototypes. Experiments show that

the latter process often achieves better test accuracy than CNN. Another issue with this algorithm is its theoretical standing. The consistency of CNN derives from the fact that its prototypes are samples and thus always keep a certain distance from each other. The cluster centers, on the other hand, are not samples but the weighted averages of samples, so it is difficult to control the distances between them. To resolve this problem, we adopt a hybrid solution that combines cluster centers and certain selected samples to maintain a desirable separation between all the resultant prototypes.



(a) (b)

Figure 1. (a) A margin exists between two data sets. (b) A positive distance exists between two data sets.

Note that it is not always appropriate to minimize training errors for SVM. Sometimes, a higher number of training errors should be tolerated so that prediction errors can be reduced. Such flexibility, which is built into the "soft-margin" version of SVM (Cortes and Vapnik, 1995; Bartlett and Shaw-Taylor, 1999), yields better test accuracy than the "hard-margin" version. Fortunately, this flexibility also exists in adaptive prototype learning (APL) algorithms, and can be derived by a tradeoff between the number of prototypes and their predictive power. However, although APL reduces training errors by adding prototypes, it increases the risk of overfitting. A balance between these two factors is made possible by a cross-validation study, similar to that used for SVM. We discuss this point further in Section 6.

In summary, we propose two types of prototype learning algorithm. The first is an instance-based algorithm, which adds samples as prototypes according to an enhanced absorption criterion. The advantage of this approach (discussed in Section 7.2) is that it achieves substantially higher test accuracy at a relatively low training cost, compared to other instance-based algorithms, whose major merit is a lower ratio of prototypes to training samples. Although our algorithm achieves higher test accuracy at the expense of a somewhat higher ratio of prototypes to training samples, we believe this is acceptable, since it enables the proposed classifier to even outperform the *k*-nearest neighbor (*k*-NN) rule in terms of accuracy. The second approach is a hybrid method that constructs prototypes as either samples or the weighted averages of samples. Compared to SVM, the hybrid prototype learning method yields higher test accuracy, at the expense of a higher training cost (discussed in Section 7.3).

The remainder of the paper is organized as follows. In the next section, we present the Vapnik-Chervonenkis (VC) theory of multiclass classification. In Section 3, we provide proof of the consistency of CNN under certain conditions. In Section 4, the extension of CNN to APL is discussed, along with the convergence of APL and its consistency under the same conditions. In Sections 5 and 6, respectively, we describe a kernelized version and a soft version of APL. Section 7 contains experimental studies of APL and comparisons with some instance-based

learning algorithms, namely, *k*-NN, CNN and SVM. Finally, in Section 8, we present our conclusions.

2 Vapnik-Chervonenkis Theory of Multiclass Classification

In this section, we develop a basic theory of prototype-learning algorithms. In particular, we derive an asymptotic result for generalization errors of prototype learning algorithms. For the case of binary classification, in which an object is classified as one of two class types, the standard Vapnik-Chervonenkis (VC) theory provides such a bound. This theory, however, is not sufficient for our purpose, since we deal with multiclass classifications in which we want to classify an object into one of *m* classes, with $m \geq 2$. Here, we focus on extending the standard VC theory to such a case.

The standard VC theory is a probabilistic theory that has great breadth and depth. To present a complete version of the theory in a journal paper is impossible. In fact, it is also unnecessary, since a comprehensive treatment can be found in the book *A Probabilistic Theory of Pattern Recognition* (Devroye et al., 1996). For this reason, we follow its notations closely (with some minor changes to suit our purpose) and quote those theorems that are relevant to our task.

We assume there are *n* training samples $(\mathbf{x}_1, y_1)$, …, $(\mathbf{x}_n, y_n)$, and a test sample $(\mathbf{x}, y)$ drawn independently from the set $R^d \times \Lambda$ according to the same distribution, where $\Lambda = \{1, 2, \ldots, m\}$ is a set of labels or class types. Then, for a classifier $g: R^d \rightarrow \Lambda$, we define its training error $\hat{L}_n(g)$ and testing error $L(g)$ as follows.

Definition 1 *The training error of a classifier g is defined as the fraction of training samples misclassified by g, that is,* $\hat{L}_n(g) = (1/n)\sum_{i=1}^{n} I_{\{g(\mathbf{x}_i) \neq y_i\}}$, *where I is the indicator function such that* $I_{\{g(\mathbf{x}_i) \neq y_i\}} = 1$ *if and only if* $g(\mathbf{x}_i) \neq y_i$. *The testing error of a classifier g is defined as the probability that a test sample has been misclassified by g, that is, L(g) = Pr{g(**x**)≠y}.*

Typically, from the training samples, a learning algorithm tries to build a classifier *g* of a generic class *C*, with the objective that *g* can generalize well in the sense that it has a small testing error. The standard VC theory provides a bound for the testing error of *binary* classifiers. This bound can be expressed in terms of the following complexity measure of *C*.

Definition 2 *Let C be a collection of binary classifiers of the form* $g : R^d \rightarrow \{0, 1\}$. *For any n, the* n^{th} *shatter coefficient of C is defined as*

$$S(C, n) = \max_{T \subseteq R^d, |T|=n} |\{g_T : g \in C\}|,$$

where g_T *is the function obtained by restricting g to the domain T, and |X| for any set X is the number of elements of X.*

Intuitively, the n^{th} shatter coefficient of *C* is the maximum number of ways that an *n*-element subset can be partitioned by the classifiers in *C*. The following well-known result of Vapnik and Chervonenkis (1971, 1974a, 1974b) provides a bound for the testing error of classifiers in *C*, which we denote as the VC-bound. We adopt this result from Theorem 12.6 in Devroye et al. (1996).

Theorem 3 *Let C be a collection of binary classifiers. Then, for any n and any* $\varepsilon > 0$,

$$\Pr\left\{\sup_{g \in C} |\hat{L}_n(g) - L(g)| > \varepsilon\right\} \leq 8S(C, n)e^{-n\varepsilon^2/32}.$$

Next, we explain how to obtain an analogous result for multiclass classifiers. For a collection C of multiclass classifiers g: $R^d \to \Lambda$, let C^B be the class of binary classifiers g^i: $R^d \to \{0, 1\}$ such that $g^i(x) = 1$ if and only if $g(x) = i$, for $g \in C$ and $i = 1, 2, \ldots, m$.

Theorem 4 *Let C be a collection of multiclass classifiers of the form g: $R^d \to \Lambda$. Then, for any n and any $\varepsilon > 0$,*

$$\Pr\left\{\sup_{g\in C} |\hat{L}_n(g) - L(g)| > \varepsilon\right\} \le 8S(C^B, n)e^{-n\varepsilon^2/(8m^2)}.$$

Proof: First, consider any classifier g: $R^d \to \Lambda$. A sample $(\mathbf{x}, y) \in R^d \times \Lambda$ is misclassified by g if and only if it is misclassified by both g^y and $g^{g(\mathbf{x})}$. Thus, $2\hat{L}_n(g) = \sum_{i=1}^m \hat{L}_n(g^i)$ and $2L(g) = \sum_{i=1}^m L(g^i)$. Then, by triangle inequality, the condition $|L_n(g) - L(g)| > \varepsilon$ implies that

$$\sum_{i=1}^m |\hat{L}_n(g^i) - L(g^i)| \ge |\sum_{i=1}^m \hat{L}_n(g^i) - \sum_{i=1}^m L(g^i)| = |2\hat{L}_n(g) - 2L(g)| > 2\varepsilon,$$

and $|\hat{L}_n(g^i) - L(g^i)| > 2\varepsilon/m$ for some i. Therefore,

$$\Pr\left\{\sup_{g\in C} |\hat{L}_n(g) - L(g)| > \varepsilon\right\} \le \Pr\left\{\sup_{g^i \in C^B} |\hat{L}_n(g^i) - L(g^i)| > 2\varepsilon/m\right\}.$$

This theorem follows from the previous theorem with ε replaced by $2\varepsilon/m$. □

As stated in the above theorem, the VC-bound is the product of two terms. The first is just the shatter coefficient, whose magnitude depends on the collection of classifiers C. The second term decays exponentially to zero as $n \to \infty$. To obtain an asymptotic result from this product, we need to know how fast the shatter coefficient grows as $n \to \infty$. If its growth is slower than the decay of the second term, then the VC-bound approaches zero as $n \to \infty$.

Let us now define some terms. A prototype data pair $(\mathbf{p}, y)$ consists of a prototype $\mathbf{p} \in R^d$ and its label y. We say that a classifier uses the 1-NN rule based on prototype data pairs if g assigns to each $\mathbf{x} \in R^d$ the label of the nearest prototype to $\mathbf{x}$. The collection of all *multiclass* classifiers using the1-NN rule based on k prototype data pairs is denoted by $C_{(k)}$, while the collection of *binary* classifiers using the 1-NN rule based on k prototype data pairs is denoted by $B_{(k)}$. We want to derive a result for $C_{(k)}$ in terms of a known result of $B_{(k)}$. To do this, we adopt the following lemma from Devroye et al. (1996, p. 305), which provides a bound for $S(B_{(k)}, n)$.

Lemma 5 *$S(B_{(k)}, n) \le (ne/(d+1))^{(d+1)(k-1)}$.*

From Theorem 4 and Lemma 5, we derive the following result for $C_{(k)}$.

Theorem 6 *For any n and any $\varepsilon > 0$,*

$$\Pr\left\{\sup_{g\in C_{(k)}} |\hat{L}_n(g) - L(g)| > \varepsilon\right\} \le 8(ne/(d+1))^{(d+1)(k-1)} e^{-n\varepsilon^2/(8m^2)}.$$

Proof: In order to apply Theorem 4, we need to find a bound for $S(C^B_{(k)}, n)$, where $C^B_{(k)}$ derives from $C_{(k)}$ in the same way as C^B derives from C. Since $C^B_{(k)} \subseteq B_{(k)}$, we have $S(C^B_{(k)}, n) \le S(B_{(k)}, n)$, which follows easily from the definition of the shatter coefficient.

Therefore, by Lemma 5, we have $S(C^B_{(k)}, n) \le (ne/(d+1))^{(d+1)(k-1)}$. Combining this inequality and Theorem 4, we obtain the desired result. □

For a given sequence of training data $D_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, a classification rule is a sequence of classifiers $\{g_n\}$ such that g_n is built on D_n. Such a rule is said to be *consistent*, or asymptotically Bayes-risk efficient, if

$$\lim_{n\to\infty} \Pr\{L(g_n) - \inf_g L(g) > \varepsilon\} = 0,$$

for any $\varepsilon > 0$, where $\inf_g L(g)$ is the infimum (i.e., the greatest lower bound) of the testing errors of all classifiers of the form g: $R^d \to \Lambda$. A prototype classification rule, on the other hand, is a sequence $\{g_n\}$ such that g_n uses the 1-NN rule based on k_n prototypes for some k_n. The following corollary provides a sufficient condition for the consistency of a prototype classification rule. Note that, in stating the corollary, we use o($f(n)$) to denote a quantity whose ratio to $f(n)$ approaches zero as $n \to \infty$.

Corollary 7 *Suppose that $\{g_n\}$ is a prototype classification rule such that $\hat{L}_n(g_n) = 0$ and k_n = o($n\varepsilon^2/(m^2 d \log n)$) for all n. Then, for any $\varepsilon > 0$,*

$$\lim_{n\to\infty} \Pr\{L(g_n) - \inf_g L(g) > \varepsilon\} = \lim_{n\to\infty} \Pr\{L(g_n) > \varepsilon\} = 0.$$

Proof: Since $\hat{L}_n(g_n) = 0$, the condition that $L(g_n) > \varepsilon$ implies that $|\hat{L}_n(g_n) - L(g_n)| > \varepsilon$ and thus $\sup_{g\in C_{(k_n)}} |\hat{L}_n(g) - L(g)| > \varepsilon$ as well. Hence, $\Pr\{L(g_n) > \varepsilon\} \le \Pr\left\{ \sup_{g\in C_{(k_n)}} |\hat{L}_n(g) - L(g)| > \varepsilon \right\}$. Also, since k_n = o($n\varepsilon^2/(m^2 d \log n)$), by Theorem 6, we have $\Pr\{L(g_n) > \varepsilon\} \to 0$ as $n \to \infty$. Finally, $L(g_n) - \inf_g L(g) > \varepsilon$ implies that $L(g_n) > \varepsilon$, so the probability of the former inequality also approaches zero as $n \to \infty$. □

3 The Condensed Nearest Neighbor Rule

Following the notations defined in Section 2, we assume that a set of observed data, or samples, $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_n, y_n)$ is given. Our goal here is to extract a subset U_n from $X_n = \{\mathbf{x}_i\}_{i=1}^n$ in such a way that if $\mathbf{u}$ is the nearest member of U_n to $\mathbf{x}_i$, then $l(\mathbf{u}) = y_i$, where $l(\mathbf{u})$ is the label of $\mathbf{u}$. Members of U_n are called prototypes, and samples whose labels match those of their nearest prototypes are said to be *absorbed*.

The CNN rule (Hart, 1968) is a simple way of solving the above problem. Starting with $U_n = \{\mathbf{x}_0\}$, where $\mathbf{x}_0$ is randomly chosen from X_n, CNN scans all members of X_n. It then adds to U_n a member $\mathbf{x}$ of X_n whose nearest prototype's label does not match that of $\mathbf{x}$. The algorithm scans X_n as many times as necessary, until all members of X_n have been absorbed or, equivalently, no more prototypes can be added to U_n.

Let $\delta_n = \min\{\| \mathbf{x}_i - \mathbf{x}_j \| : \mathbf{x}_i, \mathbf{x}_j \in X_n, \text{and } l(\mathbf{x}_j) \ne l(\mathbf{x}_j)\}$, that is, δ_n is the minimal distance between heterogeneous samples. Since $\{\delta_n\}$ is a decreasing sequence, there exists a δ such that $\delta_n \to \delta$ as $n \to \infty$. The consistency of the CNN rule can be proved under the following two conditions. 1) *Boundedness*: all samples are included in a bounded set; that is, there exists a

region H of radius R such that $\Pr\{\mathbf{x} \in H\} = 1$. 2) *Non-zero separation*: the limit δ of $\{\delta_n\}$ is non-zero.

Lemma 8 *Under the conditions of boundedness and non-zero separation, the number of CNN prototypes cannot exceed* $(2R/\delta + 1)^d$*, where R is the radius of H and* δ *is the limit of* $\{\delta_n\}$.

Proof: We want to prove that all prototypes are δ-separated, that is, their distance is at least δ. This is true for any two prototypes with different labels, since all prototypes are samples and heterogeneous samples are δ-separated. Therefore, we only have to prove that all prototypes of the same label are also δ-separated.

We assume that **p** and **q** are prototypes of the same label. As CNN is a sequential process, its prototypes are constructed in linear order. Without loss of generality, we assume that **p** is constructed before **q**; hence, there must be a prototype **m** that is constructed before **q**, $l(\mathbf{m}) \neq l(\mathbf{q})$, and $\|\mathbf{q}-\mathbf{p}\| \geq \|\mathbf{q}-\mathbf{m}\|$. Now, since **q** and **m** have different labels, $\|\mathbf{q}-\mathbf{m}\| \geq \delta$. Combining these two facts, we obtain $\|\mathbf{q}-\mathbf{p}\| \geq \delta$.

We define a ball of radius r centered at **w** as $B(\mathbf{w}, r) = \{\mathbf{x} : \|\mathbf{x}-\mathbf{w}\| < r\}$. Let the prototypes be $\{\mathbf{p}_i\}_{i=1}^k$. Since they are δ-separated from each other, all the balls $B(\mathbf{p}_i, \delta/2)$ are non-overlapping. Moreover, the union of these balls is contained in a ball of radius $R + \delta/2$ (Figure 2). So, we must have $k(\delta/2)^d < (R+\delta/2)^d$, or $k < (2R/\delta+1)^d$. □

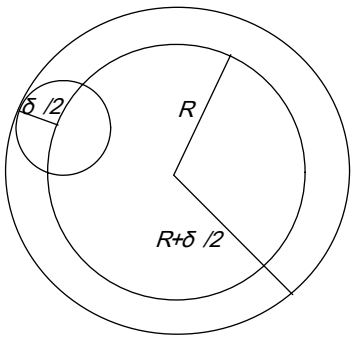


Figure 2. If all samples are contained in a ball of radius R, then all balls of radius $\delta/2$ centered at a sample are included in a ball of radius $R+\delta/2$.

From this lemma and the corollary to Theorem 6, we derive the following.

Theorem 9 *Let* $\{g_n\}$ *be a sequence of classifiers using the 1-NN rule based on CNN prototype data pairs. The boundedness and non-zero separation conditions ensure the consistency of* $\{g_n\}$.

4 Adaptive Prototype Learning Algorithms

An adaptive prototype learning algorithm is similar to CNN in that it adds as many prototypes as necessary until all samples have been absorbed. APL, however, differs from CNN in two respects: the absorption criterion and the nature of prototypes. In CNN, all prototypes are samples, whereas prototypes in APL can be samples or the weighted averages of samples. We denote prototypes that are samples as instance-based prototypes (IBPs) to differentiate them from cluster-based prototypes (CBPs), which are the weighted averages of samples. First, we develop a special type of APL algorithm for IBPs and prove its consistency under the conditions that ensure the consistency of CNN. We then propose a more complex type of APL that combines IBPs and CBPs, after which we address APL's convergence and consistency properties.

4.1 Generalized CNN

The instance-based APL includes CNN as a special case. For this reason, we denote it as a generalized CNN, or GCNN. The difference is that GCNN employs a strong absorption criterion, in contrast to the weak criterion employed by CNN. According to CNN, a sample **x** is absorbed if

$$\| \mathbf{x}-\mathbf{q} \| - \| \mathbf{x}-\mathbf{p} \| > 0, \tag{1}$$

where **p** and **q** are prototypes, **p** is the nearest homogeneous prototype to **x**, and **q** is the nearest heterogeneous prototype to **x**. For GCNN, however, we adopt the following criterion:

$$\| \mathbf{x}-\mathbf{q} \| - \| \mathbf{x}-\mathbf{p} \| > \rho\delta_n, \qquad \rho \in [0,1). \tag{2}$$

We say that a sample is *weakly* absorbed if it satisfies (1), and *strongly* absorbed if it satisfies (2). Note that (1) corresponds to the case where $\rho = 0$ in (2). Adopting (2) makes it possible to improve the classifier by optimizing ρ. The question of how to optimize ρ is addressed in Section 6.

We now describe the steps of GCNN.

G1 *Initiation*: For each label *y*, select a *y*-sample as an initial *y*-prototype.

G2 *Absorption Check*: Check whether each sample is strongly absorbed (absorbed, for short). If all samples are absorbed, terminate the process; otherwise, proceed to the next step.

G3 *Prototype Augmentation*: For each *y*, if any unabsorbed *y*-samples exist, select one as a new *y*-prototype; otherwise, no new prototype is added to label *y*. Return to G2 to proceed.

In G1, a *y*-sample is selected as follows. We let each *y*-sample cast a vote to its nearest *y*-sample, and select the one that receives the highest number of votes. In G3, an unabsorbed *y*-sample is selected as follows. Let $\Psi_y = \{\mathbf{x}_i: l(\mathbf{x}_i)=y \ \& \ \mathbf{x}_i \text{ is unabsorbed}\}$. We let each member of Ψ_y cast a vote for the nearest member in this set. The selected *y*-sample is the member of Ψ_y that receives the highest number of votes.

Lemma 10 *GCNN prototypes satisfy the following properties. (a) For each prototype* ***p****, no heterogeneous sample can be found in* $B(\mathbf{p},\delta_n)$. *(b) For any two heterogeneous prototypes* ***p*** *and* ***q****,* $\| \mathbf{p}-\mathbf{q} \| \geq \delta_n$. *(c) For any two homogeneous prototypes* ***m*** *and* ***n****,* $\| \mathbf{m}-\mathbf{n} \| > (1-\rho)\delta_n$.

Proof: Propositions (a) and (b) follow from the fact that GCNN prototypes are samples and the separation between any two heterogeneous samples is at least δ_n. To prove (c), let two homogeneous prototypes **m** and **n** be given, and let **m** be constructed before **n**. Since **n** is not absorbed by the time it is taken as a prototype, there exists a heterogeneous prototype **q** such that $\| \mathbf{n}-\mathbf{q} \| - \| \mathbf{n}-\mathbf{m} \| \leq \rho\delta_n$ or, equivalently,

$$\| \mathbf{n}-\mathbf{m} \| \geq \| \mathbf{n}-\mathbf{q} \| - \rho\delta_n. \tag{3}$$

Since **n** and **q** are heterogeneous, by (a), **n** cannot lie in $B(\mathbf{q},\delta_n)$. Thus,

$$\| \mathbf{n}-\mathbf{q} \| \geq \delta_n. \tag{4}$$

Combining (3) and (4), we obtain $\| \mathbf{n}-\mathbf{m} \| \geq \delta_n - \rho\delta_n = (1-\rho)\delta_n$. □

We define the number of iterations as the number of times G2 has been executed. The following lemma states that the number of iterations cannot exceed a certain magnitude. Let R_n be the radius of the smallest ball containing all samples in X_n.

Lemma 11 *The number of GCNN prototypes cannot exceed* $[(2R_n + \delta_n)/(1-\rho)\delta_n]^d$. *Moreover, GCNN converges within a finite number of iterations.*

Proof: Lemma 10 ensures that homogeneous and heterogeneous GCNN prototypes are separated by certain constants. Using this fact and a similar argument to that in the proof of Lemma 8, we conclude that the number of GCNN prototypes cannot exceed $[(2R + \delta_n)/(1-\rho)\delta_n]^d$. Since at least one prototype is created at each iteration, the number of iterations cannot exceed this number either. □

Now, under the conditions of boundedness and non-zero separation, we can also show that the number of GCNN prototypes is bounded from above by $[(2R + \delta_n)/(1-\rho)\delta_n]^d$, with R replacing R_n. Since $[(2R + \delta_n)/(1-\rho)\delta_n]^d \le [(2R + \delta)/(1-\rho)\delta]^d$, the number of GCNN prototypes is bounded from above by a constant independent of *n*. The consistency of GCNN follows from the same argument that demonstrates the consistency of CNN.

Theorem 12 *Under the conditions of boundedness and non-zero separation, GCNN is consistent.*

4.2 Linear Adaptive Prototype Learning

Having explained GCNN, we are ready to describe a more complex type of APL that can take a mixture of IBPs and CBPs as its prototypes. To differentiate it from GCNN, and from another version of APL to be described later, we denote this algorithm as linear APL (LAPL).

Recall that the consistency of GCNN derives from the separation of prototypes. We wish to obtain a similar separation between LAPL prototypes, but the addition of CBPs raises some problems.

The first problem is the separation required for heterogeneous prototypes. While a δ_n separation can be easily maintained by any two heterogeneous IBPs, it may not be maintained so easily by two heterogeneous CBPs. Therefore, we require the separation to be $f\delta_n$, where $f \in [0,1]$. How we determine the optimal value of *f* is discussed in Section 6.

The next problem is the absorption criterion. For LAPL, we adopt the following:

$$\| \mathbf{x} - \mathbf{q} \| - \| \mathbf{x} - \mathbf{p} \| > \rho f \delta_n, \quad \text{for } \rho \in [0,1). \tag{5}$$

How to optimize ρ is also addressed in Section 6.

The third problem is how to maintain a positive separation between *all* LAPL prototypes. To achieve this objective, we specify the following requirements.

(C1) For each prototype **p**, no heterogeneous sample exists in $B(\mathbf{p}, f\delta_n)$.

(C2) For any two heterogeneous prototypes **p** and **q**, $\| \mathbf{p} - \mathbf{q} \| \ge f\delta_n$.

(C3) For any two homogeneous prototypes **m** and **n**, $\| \mathbf{m} - \mathbf{n} \| > (1-\rho) f\delta_n$.

Thus, in the transition from GCNN to LAPL, we have systematically changed δ_n to $f\delta_n$. We now state the LAPL algorithm, and prove that the prototypes derived from it satisfy (C1), (C2), and (C3). We first describe the general scheme of the algorithm, and then provide the technical details. The steps of LAPL are:

H1 *Initiation*: For each label *y*, initiate a *y*-prototype as the average of all *y*-samples. If this prototype does not satisfy (C1), (C2), and (C3), we apply the prototype adjustment module (described later in this section).

H2 *Absorption Check*: Check whether each sample has been absorbed. If all samples have been absorbed, terminate the process; otherwise, proceed to the next step.

H3 *Prototype Refreshment*: For each un-satiated label y (i.e., some y-samples are unabsorbed), select an unabsorbed y-sample. We then apply a clustering algorithm to construct clusters, using the selected y-sample and all existing y-prototypes as seeds. The centers of the resultant clusters are new y-prototypes. If these prototypes do not satisfy (C1), (C2), and (C3), we apply the prototype adjustment module (described later in this section). Return to H2 to proceed.

We now provide the technical details.

Selection of Unabsorbed Samples in H1 and H3. The selection procedures in H1 and H3 are the same as those in G1 and G3.

Clustering Algorithms in H3. Any clustering algorithm can be used. For the experiment described in this paper, we use the k-means (KM) and the fuzzy c-means (FCM) clustering algorithms, both of which are applied to training samples of the same label. Thus, if there are m labels in the training data, we apply the algorithms m times. Details of the methods are as follows.

The KM method (Lloyd, 1982; Max, 1960; Linde et al. 1980) derives a locally optimal solution to the problem of finding a set of cluster centers $\{\mathbf{c}_i\}_{i=1}^{p}$ that minimizes the objective function

$$\sum_{j=1}^{n} \min_{i=1,\ldots,p} \| \mathbf{c}_i - \mathbf{x}_j \|^2 . \tag{6}$$

KM's iterative process is performed as follows. Setting seeds as the initial cluster centers, we add each sample to the cluster whose center is nearest to it. We then reset the center of each cluster as the average of all the samples that fall in that cluster. To ensure rapid convergence of KM, we require that the process stops when the number of iterations reaches 30, or the membership of the clusters remains unchanged after the previous iteration.

In FCM (Bezdek, 1981; Höppner et al., 1999), the objective function to be minimized is

$$\sum_{j=1}^{n} \sum_{i=1}^{p} u_{ij}^{m} \| \mathbf{c}_i - \mathbf{x}_j \|^2 , \quad \text{for } m \in (1, \infty) \tag{7}$$

under the constraint

$$\sum_{i=1}^{p} u_{ij} = 1 , \qquad \text{for } j = 1, 2, ..., n, \tag{8}$$

where u_{ij} is the membership grade of sample $\mathbf{x}_j$ to prototype $\mathbf{c}_i$. Using the Lagrangian method, we can derive the following equations:

$$u_{ij} = \frac{\left(1 / \| \mathbf{c}_i - \mathbf{x}_j \|\right)^{\frac{2}{m-1}}}{\sum_{k=1}^{p} \left(1 / \| \mathbf{c}_k - \mathbf{x}_j \|\right)^{\frac{2}{m-1}}}, \tag{9}$$

$$\mathbf{c}_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} \mathbf{x}_j}{\sum_{j=1}^{n} u_{ij}^{m}}, \tag{10}$$

for $i = 1, 2, \ldots, p$, and $j = 1, 2, \ldots, n$ respectively. FCM is a numerical method that finds a locally optimal solution for (9) and (10). Using a set of seeds as the initial solution for $\{\mathbf{c}_i\}_{i=1}^{p}$, the algorithm computes $\{u_{ij}\}_{i,j=1}^{p,n}$ and $\{\mathbf{c}_i\}_{i=1}^{p}$ iteratively. To ensure rapid convergence of FCM, we require that the process stops when the number of iterations reaches 30, or $\sum_{i=1}^{p} \| \mathbf{c}_i^{old} - \mathbf{c}_i^{new} \| = 0$.

The prototype adjustment module is used to adjust the location of prototypes if they do not satisfy the separation conditions (C1), (C2), and (C3).

Prototype Adjustment in H1. The purpose of this module is to replace prototypes that violate (C1) or (C2) with those that do not. Note that there is only one prototype per label in H1, so we do not need to worry about (C3). There are two steps in this stage.

Step 1: If we find a CBP **p** that violates (C1), which requires that no heterogeneous sample exists in $B(\mathbf{p}, f\delta_n)$, we replace **p** with a sample of the same label. The replacement sample is an IBP and is selected in exactly the same way as a seed is selected in G1.

Step 2: If we find a CBP **p** that violates (C2), which requires that $\|\mathbf{p}-\mathbf{q}\| \geq f\delta_n$ for any other prototype **q**, we replace **p** with an IBP of the same label. We perform this operation iteratively, until the desired separations hold between CBPs, and between CBPs and IBPs.

We now prove that after these two steps, all prototypes satisfy (C1) and (C2). We first prove that, after Step 1, all prototypes satisfy (C1). By assumption, all CBPs satisfy (C1) after this step. Also, all IBPs satisfy (C1), since all heterogeneous samples are δ_n-separated from them and $\delta_n \geq f\delta_n$. We now prove that, after Step 2, all prototypes satisfy both (C1) and (C2). It is clear that all prototypes satisfy (C1) at the end of this step; and each CBP maintains $f\delta_n$-separations from other prototypes by assumption. Also, since each IBP maintains δ_n-separations from other IBPs and $\delta_n \geq f\delta_n$, each IBP maintains $f\delta_n$-separations from other IBPs.

Prototype Adjustment in H3. This module adjusts prototypes in two steps.

Step I: A set of prototypes of the same label is called a *pack*. When a pack consists of CBPs that satisfy (C1) and they are $(1-\rho)f\delta_n$-separated from each other, as required by (C3), we preserve that pack. Otherwise, we replace it with the set of seeds from which the CBPs were derived.

Step II: Two packs are said to be $f\delta_n$-separated if any two prototypes drawn from them are $f\delta_n$-separated. When we find two packs that are not $f\delta_n$-separated, we replace one of them with the set of seeds from which its prototypes were derived. We perform this operation iteratively until the remaining packs are $f\delta_n$-separated.

We now show that, after Step I, all prototypes satisfy (C1) and (C3). For convenience, we call preserved packs *P-packs* and replacement packs *R-packs*. An R-pack consists of existing prototypes, called X-prototypes, and an unabsorbed sample, called a U-prototype. By induction, all X-prototypes meet (C1) and (C3). The U-prototype, denoted as **u**, also satisfies (C1), because heterogeneous samples are δ_n-separated from each other. It remains to show that **u** is $(1-\rho)f\delta_n$-separated from all X-prototypes. This fact follows from a similar argument to that for Lemma 10(c), so we omit the proof. We conclude that all the prototypes satisfy (C1) and (C3).

We now prove that, after Step II, all prototypes satisfy (C1), (C2), and (C3). It is clear that all prototypes satisfy (C1) and (C3) at the end of this step. It remains to prove that they also satisfy (C2), that is, heterogeneous prototypes are $f\delta_n$-separated. We want to show that all heterogeneous prototypes in the R-packs are $f\delta_n$-separated. As noted earlier, these prototypes consist of X-prototypes and U-prototypes. By induction, heterogeneous X-prototypes are $f\delta_n$-separated. Heterogeneous U-prototypes are also $f\delta_n$-separated, as noted before. U-prototypes are also $f\delta_n$-separated from heterogeneous X-prototypes, since all X-prototypes satisfy (C1). Thus, at the end of Step II, all prototypes satisfy (C2).

The $f\delta_n$-separation between heterogeneous prototypes and the $(1-\rho)f\delta_n$-separation between homogeneous prototypes imply the convergence of LAPL and also its consistency under the

conditions of boundedness and non-zero separation. Note that the above conclusions do not hold for the case where $f = 0$, which we deal with in Section 5.

Theorem 13 *The LAPL terminates within a finite number of iterations, provided that* $f \in (0,1]$, $\rho \in [0,1)$, *and* $m \in (1,\infty)$.

Theorem 14. *Let* $\{g_n\}$ *be a sequence of classifiers using the 1-NN rule based on LAPL prototype data pairs. The conditions of boundedness and non-zero separation ensure the consistency of* $\{g_n\}$, *provided that* $f \in (0,1]$, $\rho \in [0,1)$, *and* $m \in (1,\infty)$.

5 Kernelized Adaptive Prototype Learning Algorithms

Let $\Phi : R^d \to H$ be a function that maps from the d-dimensional Euclidean space to a Hilbert space, whose dimension dim(H) may be infinite. In a kernelized adaptive prototype learning algorithm, the goal is to build prototypes in H. To this end, we first transform the given observed data $\{\mathbf{x}_j\}_{j=1}^n$ into $\{\Phi(\mathbf{x}_j)\}_{j=1}^n$. When either KM or FCM is used to compute prototypes, each prototype in H is of the form

$$\mathbf{c}_i = \sum_{j=1}^n u_{ij}^m \Phi(\mathbf{x}_j) / \sum_{j=1}^n u_{ij}^m ,$$

where $\mathbf{c}_i$ and u_{ij} were introduced in (6), (7), and (8). When KM is used, $m \equiv 1$. Moreover, $u_{ij} = 1/n_i$ provided that the j^{th} sample falls in the i^{th} cluster, whose population size is n_i; otherwise, $u_{ij} = 0$. When FCM is used, we compute u_{ij} according to (9) in which the distance now becomes a kernel-based distance, to be defined below.

If $\dim(H) = \infty$, $\mathbf{c}_i$ cannot be expressed in vector form. Even when $\dim(H) < \infty$, it can be computationally expensive to find an explicit form of $\mathbf{c}_i$. Fortunately, we can compute the distance between $\Phi(\mathbf{x}_j)$ and $\mathbf{c}_i$ directly, provided there exists a kernel function (Mercer, 1909; Girosi, 1998)

$$K(\mathbf{x},\mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle, \quad \text{for } \mathbf{x},\mathbf{y} \in R^d .$$

When such a function exists, we obtain the kernel-based distance as

$$\begin{aligned} \| \mathbf{c}_i - \Phi(\mathbf{x}_j) \|_{Ker}^2 &= \langle \mathbf{c}_i - \Phi(\mathbf{x}_j), \mathbf{c}_i - \Phi(\mathbf{x}_j) \rangle \\ &= \langle \mathbf{c}_i, \mathbf{c}_i \rangle - 2\langle \mathbf{c}_i, \Phi(\mathbf{x}_j) \rangle + \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_j) \rangle . \end{aligned} \tag{11}$$

Moreover,

$$\langle \mathbf{c}_i, \mathbf{c}_i \rangle = \frac{\sum_{k=1}^n \sum_{l=1}^n u_{ik}^m u_{il}^m K(\mathbf{x}_k, \mathbf{x}_l)}{\left(\sum_{k=1}^n u_{ik}^m \right)^2}, \tag{12}$$

$$\langle \mathbf{c}_i, \Phi(\mathbf{x}_j) \rangle = \frac{\sum_{k=1}^n u_{ik}^m K(\mathbf{x}_k, \mathbf{x}_j)}{\sum_{k=1}^n u_{ik}^m}, \tag{13}$$

and

$$\left\langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_j) \right\rangle = K(\mathbf{x}_j, \mathbf{x}_j). \tag{14}$$

From $\{\| \mathbf{c}_i - \Phi(\mathbf{x}_j) \|^2_{Ker}\}^{p,n}_{i,j=1}$, we derive $\{u_{ij}\}^{p,n}_{i,j=1}$ according to the appropriate formula in the clustering algorithm being used. Since we do not want to express the prototypes explicitly, we use $\{u_{ij}\}^{p,n}_{i,j=1}$ to represent them instead. From the prototypes, we can compute $\{\| \mathbf{c}_i - \Phi(\mathbf{x}_j) \|^2_{Ker}\}^{p,n}_{i,j=1}$ again, using (11)-(14). This iterative process stops if the number of iterations reaches 30, or $\max_{i,j} | u^{old}_{ij} - u^{new}_{ij} | < 0.001$. These steps represent the kernelized versions of KM and FCM, depending on which definition of $\{u_{ij}\}^{p,n}_{i,j=1}$ is used. The kernelized version of FCM was proposed and studied by Wu, Xie and Yu (2003) and Kim et al. (2005).

There is also a kernelized version of GCNN, but we do not consider it in this paper, for the reason to be given in Section 7. We denote the kernelized version of LAPL simply as KAPL, which is derived from LAPL by replacing KM or FCM with an appropriate kernelized version. In addition, we make the following changes. First, the initial *y*-prototype in KAPL should be the average of all $\{\Phi(\mathbf{x}) : l(\mathbf{x}) = y\}$. Using (11)-(14), we compute the distance of each $\Phi(\mathbf{x})$ to this prototype. Second, we apply the prototype adjustment module in KAPL to separate prototypes. Prototype separation, however, does not imply the convergence of KAPL, since it may have to deal with data in a space of infinite dimensions.

To ensure the convergence of KAPL, we modify the prototype adjustment module as follows. As in LAPL, we adopt the necessary operations to create the desired prototype separation. However, prior to these operations we check if each prototype in a pack has a non-empty domain of attraction (DOA), where the DOA of a *y*-prototype **p** is the set of all *y*-samples $\Phi(\mathbf{x})$ for which **p** is the nearest prototype. Recall that we employ a clustering algorithm to create a pack of prototypes, using an unabsorbed sample $\Phi(\mathbf{u})$ and some other prototypes as seeds. If any prototype in a pack has an empty DOA, we replace that pack with the pack of prototypes constructed earlier. In this case, $\Phi(\mathbf{u})$ is called a *futile* sample. If a sample is declared *futile* in an iteration, it will not be taken as a sample in any later iteration.

Theorem 12 *The KAPL algorithm converges within a finite number of iterations.*

Proof: The number of futile samples is bounded from above, since it cannot exceed *n*, that is, the number of samples. We assume that the last futile sample is created at iteration *i*, with $i \le n$. If all samples are absorbed at the end of *i*, the proof is complete; otherwise, more prototypes will be created, all with non-empty DOAs. The number of unabsorbed samples must decrease to zero, or else the number of DOAs would eventually exceed the number of samples, which would be an absurd result. □

Note that if we treat futile samples in LAPL in the same way, we can prove the convergence of LAPL in the setting where there is no guarantee of prototype separation.

Theorem 13 *Adopting the prototype adjustment module used in KAPL, LAPL converges for* $f = 0$ *and* $m \in (1, \infty)$.

6 Soft Adaptive Prototype Learning Algorithms

The versions of APL proposed thus far are designed to continue constructing prototypes until all training samples are absorbed or, equivalently, the training error declines to zero. These could be called hard versions of APL. Insistence on a zero training error, however, runs the risk of overfitting. Another approach, called the *soft alternative*, maintains the error rate at a level that

enhances the generalization power of the resultant classifier. The optimal error rate can be determined in a cross validation task, which is also needed to find the optimal values of the parameters. All versions of APL involve some parameters; for example, they all involve f and ρ, which regulate prototype separation (cf. (2), (5), (C1), (C2), and (C3)). Moreover, if FCM is used to compute cluster centers, there is another parameter m (cf. (7), (9), and (10)) to consider. In addition, some parameters in KAPL are used to define the kernel-based distance. For example, when the RBF kernel

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \| \mathbf{x} - \mathbf{y} \|^2) \tag{15}$$

is used to define the distance, there is an additional parameter γ, whose range is assumed to be $(0, \infty)$.

To search for the optimal values of the parameters, we perform cross-validation. As all the parameters are assumed to be independent, we must evaluate all combinations of them and determine which one is the most suitable for the task. When a combination of parameter values Q is given, we build prototypes on K-1 folds of data, which serves as the training data, and measure the test accuracy on the remaining fold of data, which serves as the validation data. We determine the optimal training error rate associated with Q as follows.

Given a set of training data and a set of validation data and assuming that the latter is the k^{th} fold of the data, k = 1, 2, …, K, we construct prototypes and record the following information. First, for a given level of e, we record the lowest number of iterations $n_k(e,Q)$ at which the training error rate falls below e. We also compute the validation accuracy rate $v_k(e,Q)$ for all the prototypes obtained at the end of iteration $n_k(e,Q)$. Let $v(e,Q) = \sum_{k=1}^{K} v_k(e,Q) / K$. The optimal training error rate is then

$$e_{opt}(Q) = \arg\max_{e} v(e,Q).$$

Note that once we have constructed prototypes to achieve a training error e_1, we do not need to start from the scratch to obtain a lower training error e_2. Instead, we continue to construct more prototypes until e_2 is reached. At the end of this process, we obtain $v(e,Q)$ for all e and thus $v(e_{opt}(Q),Q)$. When we have done this for all Q, we obtain the optimal Q as

$$Q_{opt} = \arg\max_{Q} v(e_{opt}(Q),Q) .$$

One additional parameter that needs be optimized is the number of k nearest prototypes, which we use in a voting mechanism to determine the label of a test sample. If a tie occurs, we classify the sample according to the nearest prototype. The optimal value of k should be evaluated in the cross-validation applied to the other parameters.

7 Experimental Results

To evaluate the APL algorithms and compare their performance with that of alternative methods, we use 12 benchmark data sets retrieved from the UCI databases (Newman et al., 1998). The results are described in three subsections. The first describes the four types of APL. In the second subsection, we compare the performance of GCNN with six instance-based prototype algorithms proposed in the literature. Then, in the third subsection, we compare the performance of the four APLs with SVM and k-NN. Note that many of the methods, including ours, require that the data must be bounded. One way to meet this requirement is to normalize all the feature values to [0,255], which can be done by the following linear transformation:

$$x \mapsto \frac{(x-v)\times 255}{V-v},$$

where x is a given feature value, V is the maximum value of the feature, and v is the minimum value. All experimental results reported in this section were obtained using an Intel Pentium 4 CPU 3.4GHz with a 1GB RAM.

7.1 Evaluation of APLs

The four types of APL are listed in Table 1. The first one is GCNN. The other three types of APL are: fuzzy linear APL (*f*-LAPL), crisp kernelized APL (*c*-KAPL), and fuzzy kernelized APL (*f*-KAPL). We use "*f*-" to indicate that the clustering algorithm employed is FCM, and "*c*-" to indicate that the technique is KM. In the experiments, the soft versions of the four APLs are used. Although we can consider the kernelized version of GCNN using RBF as the kernel function, this version of GCNN gains only slightly higher testing accuracy, at the expense of a much higher number of prototypes, than GCNN. So we choose not to discuss it. We do not discuss *c*-LAPL either, since it usually has a lower performance than *f*-LAPL.

In Table 2, we show the parameters used in the four types of APL and also the values of the parameters whose combinations are considered in our experiments. The values result from a trade-off between the demand for accuracy and the need to reduce the computation time. When a combination, Q, of parameter values is given, we have to record $v(e,Q)$ for certain values of e. In our experiments, the values of e, at which we record $v(e,Q)$, are percentages that start from 0% and increase by some increments until they reach 30%. All the percentages are listed in Table 3. The 12 benchmark data sets retrieved from the UCI databases are listed in Table 4, which also shows the number of labels, the number of samples, the number of features per sample, and the number of folds into which we divide the samples during cross validation.

	Assumed Distance
GCNN	Euclidean
f-LAPL	Euclidean
c-KAPL	RBF
f-KAPL	RBF

Table 1. The four types of APL studied in our experiments.

	Values	GCNN	*f*-LAPL	*c*-KAPL	*f*-KAPL
f	0., .1, .25, .5, .75, 1.		√	√	√
ρ	0., .1, .25, .5, .75, .99	√	√	√	√
m	1.05, 1.1, 1.2, 1.3, 1.4		√		√
γ	$a\times 10^{-b}$; a = 1, 2, ..., 9; b = 4, 5, ..., 7			√	√

Table 2. Parameters: their value range, and the types of APL that involve them; "√" indicates that the parameter is used in that type of APL. The parameters f and ρ appear in (2), (5), (C1), (C2), and (C3); m appears in (7); and γ appears in (15).

Values of *e*
0%, 1%, 2%, 3%, 4%, 5%, 7.5%, 10%, 20%

Table 3. The values of e at which we record $v(e,Q)$.

	Number of Labels	Number of Samples	Number of Features	Number of Folds
Iris	3	150	4	5
Wine	3	178	13	5
Glass	6	214	9	5
Ionosphere	2	351	34	10
Cancer	2	683	9	10
Zoo	7	101	16	5
Heart	2	270	13	5
TAE	3	151	5	5
BUPA Liver Disorders (BLD)	2	345	6	5
New Thyroid	3	215	5	5
SPECTF	2	267	44	5
Ecoli	8	336	7	5

Table 4. Information contained in the 12 data sets.

In Table 5, we show three performance measures of the four APLs, namely, the accuracy rate, the training time, and the condensation ratio. Given that K-fold cross-validation is conducted, the accuracy rate (AR) is the average accuracy over all validation data sets, each of which is one of the K folds; the training time (TT) is the sum of the training times of all training data sets, each of which consists of K-1 folds; and the condensation ratio (CR) is the average prototype-to-sample ratios obtained from all training data. Note that for most types of APL, we drop the decimal parts of their training times, since they are relatively insignificant to the integer parts. At the bottom of Table 5, we also show the average of the three measures over the 12 data sets. The boldface figures indicate that the performance of the corresponding method is the best of all the methods applied to the given data set.

The averaged figures in Table 5 show that, in terms of training time, the four APLs are ranked in the following order: GCNN, *f*-LAPL, *c*-KAPL, and *f*-KAPL. The number of all possible combinations of parameter values is the major factor that affects the amount of training time. If we divide the total training time by the above number, then the temporal differences among the four algorithms are reduced drastically, as shown in Table 6. Since APL training under different combinations of parameter values is conducted independently, some fashion of parallel computing, such as cluster computing or grid computing, would help reduce the training time.

GCNN requires the least amount of training time because it picks samples as prototypes, thereby avoiding the rather costly computation of clustering. The *c*-KAPL and *f*-KAPL algorithms, on the other hand, employ kernelized versions of KM and FCM respectively, which are relatively slow. In terms of accuracy, the order of the four APLs is exactly the opposite of that for the training time.

DATA SET		GCNN	*f*-LAPL	*c*-KAPL	*f*-KAPL
Iris	AR	96.62	97.95	98.63	98.40
	TT	0.8	30	16,225	81,334
	CR	9.6	10.33	54.00	5.83
Wine	AR	98.06	99.02	99.02	99.56
	TT	1	144	24,378	175,647
	CR	21.7	18.40	20.22	92.98
Glass	AR	69.39	71.26	72.23	72.73
	TT	1.37	314	18,906	108,891
	CR	48.5	35.98	44.98	22.90
Ionosphere	AR	89.07	91.46	95.88	95.87
	TT	9.45	8,010	399,693	3,078,420
	CR	17.5	5.63	4.56	6.05
Cancer	AR	97.5	97.79	97.50	97.79
	TT	3.34	2,301	496,817	5,265,013
	CR	17.9	4.44	12.74	19.70
Zoo	AR	97.66	97.66	97.66	97.66
	TT	0.83	11	21,666	135,346
	CR	23.2	18.32	22.77	24.50
Heart Rate	AR	85.57	86.90	85.83	86.43
	TT	1.56	1,134	72,925	607,436
	CR	42.6	21.67	35.83	23.98
TAE	AR	63.21	62.47	65.22	65.61
	TT	0.95	229	18,682	133,157
	CR	43.2	51.82	45.86	46.85
BLD	AR	65.93	67.34	67.72	70.52
	TT	2.4	3,379	232,211	1,378,124
	CR	47.9	35.87	74.13	23.33
New Thyroid	AR	97.31	97.76	98.57	99.05
	TT	0.92	135	19,289	134,671
	CR	8.7	12.79	3.72	10.00
SPECTF	AR	83.55	85.63	86.13	87.04
	TT	9.1	8,820	167,428	1,363,339
	CR	28.3	30.15	50.37	28.37
Ecoli	AR	86.44	86.81	86.18	87.06
	TT	1.32	920	35,151	216,235
	CR	24.6	31.85	48.51	27.98
AVERAGE	AR	85.86	86.84	87.55	88.14
	TT	2.3	2,119	126,948	1,056,468
	CR	27.5	23.10	34.81	27.71

Table 5. The performance of the four APLs, where AR = Accuracy Rate (%), TT = Training Time (sec), and CR = Condensation Ratio (%).

	Number of Combinations	Total Training Time	Training Time per Combination
GCNN	6	2.3	0.38
f-LAPL	155	2,119	13.67
c-KAPL	1,116	126,948	113.75
f-KAPL	5,580	1,056,468	189.33

Table 6. The number of all possible combinations of parameter values, the total training time, and the training time per combination for the four types of APL.

	Time to Compute δ_n (sec)	GCNN Run Time (sec)	Ratio (%)
Average	0.012	0.38	3.2

Table 7. The average amount of time to compute δ_n, the average run time of GCNN, and their ratio.

These findings suggest that the high accuracy rates of APLs are derived at the expense of a rather high computational cost. Hence, there is a tradeoff between accuracy rates and training costs, which allows users to choose the most suitable APL based on the size of their problems, their computing resources, and the degree of accuracy they require. There are two reasons for this tradeoff. First, the cluster-based approach has higher generalization power than the instance-based approach, since it picks the weighted averages of samples as prototypes and they are relatively immune to noise. Second, the RBF-based approach has higher generalization power than the Euclidean-based approach. To understand why this is so, we note that for very small γ, the RBF distance between $\mathbf{x}$ and $\mathbf{y}$ is approximately $2\gamma \| \mathbf{x} - \mathbf{y} \|^2$. This means that the RBF distance covers the Euclidean distance as a special case, and using the RBF distance may allow us to find a better-performing classifier than the one we obtain by using the Euclidean distance.

Recall that when applying any APL algorithm we must first compute δ_n, the minimum distance between heterogeneous samples. One may be curious about the ratio of the computing time for δ_n to the run time of APL. In fact, the ratio is 3.2% for GCNN (Table 7) and much less for the other types of APL.

The reason for such a small ratio is as follows. If the number of training samples is *n*, then the time complexity of computing δ_n is in the order of n^2, while the time complexity of conducting APL training is in the order of n^3. To confirm the latter fact, we note that APL training takes no more than *n* iterations. Within each iteration, checking the absorption criterion takes no more than n^2 steps, and clustering takes no more than $30 \times n^2$ steps (if cluster-based prototypes are required), where 30 is the maximum number of iterations allowed in a clustering algorithm. Furthermore, the space complexity of APL training is in the order of n^2 at most.

LAPL and KAPL are associated with parameters *f* and ρ, which appear in the absorption criterion (5) and requirements (C1), (C2), and (C3) (cf. Section 4). We were curious to know how the parameters' values affect the prototypes built in the training process, so we studied the training of *f*-LAPL on the 12 data sets. We assume that all parameters, except *f,* are fixed at certain values. The absorption criterion requires that a training sample should be closer to its nearest homogeneous prototype than to its heterogeneous prototype by at least $f\rho\delta_n$. If we raise the value of *f*, we increase the likelihood of a sample becoming unabsorbed so that more prototypes would have to be built. This fact is reflected in Table 8, which shows that the average

condensation ratio increases as the value of *f* increases. What happens when we fix the values of all parameters except ρ? By raising the value of ρ, we also make the absorption criterion more difficult to satisfy and therefore increase the number of prototypes that need to be built. This fact is reflected in Table 9.

f	0.00	0.10	0.25	0.50	0.75	1.00
Average Condensation Ratio (%)	22.18	23.26	24.56	27.83	30.70	34.92

Table 8. Average condensation ratio of *f*-LAPL over the 12 data sets for various values of *f* when $m = 1.1$, $\rho = 0.5$, and $e = 0$.

ρ	0.00	0.10	0.25	0.50	0.75	0.99
Averaged Condensation Ratio (%)	22.25	23.22	24.53	27.83	30.81	34.31

Table 9. Average condensation ratio of *f*-LAPL over the 12 data sets for various values of ρ when $m = 1.1$, $f = 0.5$, and $e = 0$.

7.2 Comparison of GCNN with Some Instance-Based Learning Algorithms

As noted earlier, GCNN differs from LAPL and KAPL in that it adopts samples as prototypes. It is thus one of the methods, called instance-based learning algorithms, which reduce an entire set of training samples to a subset, while maintaining as much generalization power as possible. For this reason, we compare GCNN with some of the methods that have been proposed in the literature.

Two approaches can be adopted in IBL algorithms. The first is incremental, so it starts with a null set and gradually adds samples as prototypes. Both CNN and GCNN are incremental algorithms. For comparison purposes, we also include a primitive version of GCNN, called pGCNN. It is similar to GCNN, except that the value of parameter *f* is fixed at 0. Note that pGCNN is not the same as CNN. In pGCNN, we select unabsorbed samples through a voting procedure (cf. Section 4) and the training error rate *e* is determined by cross-validation (cf. Section 6). In CNN, however, unabsorbed samples are selected randomly and *e* is fixed at 0.

The second approach is decremental, so it starts with the entire set of samples and gradually removes samples that are considered properly "protected" by the retained ones. For algorithms of this type, we include DROP1 to DROP5 (Wilson and Martinez, 2005) and ICF (Brighton and Mellish, 2002) for comparison. They differ from each other in the way samples are ordered for removal, and in the criterion for removing samples. For further details, readers should refer to the cited references. We used the code provided by Wilson and Martinez (2005) for DROP1 to DROP5, and implemented our own codes for ICF.

For all the methods, we apply cross-validation, similar to that used for the APLs, whereby the 12 data sets are divided into the same number of folds (cf. Table 4). Moreover, in measuring the test accuracy, we use the top-*k* nearest prototypes with *k* being determined in the cross-validation (cf. Section 6). Table 10 shows the performance of all the instance-based methods, with the averaged results shown at the bottom of the table. From the latter results, we observe that GCNN achieves the best accuracy among all the compared methods. In general, the incremental methods have lower training costs than the decremental methods. The only exception is GCNN, which is little slower than ICF. On the other hand, the incremental methods build more prototypes than the decremental methods. Among the incremental methods, GCNN achieves a higher accuracy rate than the other two methods, at the expense of building more prototypes and a higher training cost. Meanwhile, pGCNN constructs fewer prototypes and has a lower training cost than GCNN,

DATA SET		Incremental Methods			Decremental Methods					
		CNN	pGCNN	GCNN	DROP1	DROP2	DROP3	DROP4	DROP5	ICF
Iris	AR	93.07	96.62	96.62	92.65	95.71	95.23	95.23	95.53	95.04
	TT	0.08	0.11	0.8	0.11	0.19	0.16	0.16	0.14	0.11
	CR	13.67	10.1	9.6	6.6	9.2	10.5	10.5	8.9	22
Wine	AR	96.02	96.31	98.06	92.97	93.15	93.42	93.42	98.06	92.81
	TT	0.09	0.12	1	0.41	0.41	0.36	0.36	0.56	0.13
	CR	15.31	10.3	21.7	7.5	12.6	12.1	12.1	8.1	11.1
Glass	AR	65.20	67.54	69.39	59.02	65.83	66.23	67.12	64.19	64.09
	TT	0.11	0.1	1.37	0.38	0.33	0.41	0.39	0.53	0.17
	CR	50.12	36.5	48.5	20.3	27.1	18.8	24.3	23.5	22.3
Ionosphere	AR	87.76	87.16	89.07	77.13	88.16	86.70	88.06	88.06	81.20
	TT	0.16	0.30	9.45	5.4	6.6	8.18	7.8	13.98	0.48
	CR	23.01	15	17.5	5.9	10.1	5.3	8.2	9	3.7
Cancer	AR	96.47	97.21	97.5	96.32	96.47	96.03	96.47	96.32	96.47
	TT	0.14	0.30	3.34	21.1	34.2	34.8	30.2	21.9	0.67
	CR	10.05	7.5	17.9	2.1	5	3	3.7	3.9	2.5
Zoo	AR	97.19	96.32	97.66	94.97	93.46	92.53	93.82	90.43	90.59
	TT	0.09	0.14	0.83	0.33	0.28	0.25	0.25	0.28	0.16
	CR	15.25	11.6	23.2	14.8	17	18.4	18.8	15	44.3
Heart	AR	83.88	81.95	85.57	77.36	82.10	79.73	80.31	81.09	76.41
	TT	0.11	0.17	1.56	0.89	0.97	1.09	1	1.2	0.14
	CR	41.11	30	42.6	11	16.1	11.1	12.8	13.5	14.3
TAE	AR	58.16	61.69	63.21	51.08	51.15	51.36	53.63	55.64	52.12
	TT	0.1	0.13	0.95	0.09	0.09	0.11	0.11	0.07	0.11
	CR	61.26	42.7	43.2	25.6	27.1	23	24.3	28.8	26.6
BLD	AR	65.95	65.41	65.93	57.38	60.34	59.98	62.75	63.92	60.22
	TT	0.15	0.23	2.4	0.56	0.5	0.63	0.55	0.7	0.12
	CR	58.84	42	47.9	23	29.9	19.3	25.2	23.4	18
New Thyroid	AR	94.89	96.28	97.31	90.83	93.85	95.27	94.42	93.61	93.55
	TT	0.09	0.13	0.92	0.27	0.36	0.33	0.33	0.45	0.14
	CR	13.14	9.8	8.7	6.2	11.3	7.2	8.4	7.6	8
SPECTF	AR	79.60	83.55	83.55	77.27	74.99	79.98	74.29	76.13	76.26
	TT	0.17	0.38	9.1	2.59	3.17	3.28	2.9	3.86	0.25
	CR	46.35	20.9	28.3	10	16.5	9.1	11.7	11.7	10.1
Ecoli	AR	83.97	83.58	86.44	81.22	85.94	83.53	86.64	84.36	83.17
	TT	0.14	0.17	1.32	0.98	0.84	1.44	1.28	1.36	0.25
	CR	36.01	17.5	24.6	9.5	14.6	12.00	12.7	12.2	11.3
AVERAGE	AR	83.51	84.47	85.86	79.02	81.76	81.67	82.18	82.28	80.16
	TT	0.12	0.19	2.3	2.76	4	4.25	3.78	3.75	0.23
	CR	32.01	21.2	27.5	11.9	16.4	12.5	14.4	13.8	16.2

Table 10. The performance of three incremental methods and six decremental methods.

and yields higher accuracy and generates fewer prototypes than CNN. Both GCNN and pGCNN generate fewer prototypes than CNN, because their training error rate *e* can be non-zero, while it is fixed at zero for CNN.

Since pGCNN is a special case of GCNN with $\rho = 0$, comparison of their accuracy rates offers us an opportunity to examine the sensitivity of GCNN to the parameter values. The difference between the average accuracy rates is 1.39%, but for the Ecoli and Heart data sets, the

differences increase to 2.86% and 3.62% respectively, showing that the search for the optimal parameter values can be very useful. A similar situation is found with other types of APL.

7.3 Comparison of LAPL and KAPL with *k*-NN and SVM

To further evaluate the performance of the four APLs, we run two other alternative learning methods: *k*-NN, and SVM. Once again, for both methods, we apply cross-validation, similar to that used for APLs. For SVM, we employ the soft-margin version with the RBF kernel. Recall that the RBF function involves a parameter γ. In SVM, the value range of γ is taken as $\{a\times10^{-b}$: $a = 1, 2, \ldots, 9$ and $b = 3, 4, \ldots, 6\}$, which differs from that of KAPL by a factor of 10. Also, since the soft-margin version of SVM is used, there is an additional parameter C, which serves as a penalty factor for SVM training errors whose value range is taken as $\{10^c$: $c = -1, 0, \ldots, 5\}$. We use the LIBSVM toolkit (Hsu and Lin, 2002) to train SVM. For *k*-NN, the optimal value of k is determined during cross-validation, in much the same way that we optimize the k nearest prototypes for use in the voting procedure to determine the label of a test sample (cf. Section 6).

One crucial difference between SVM and APL is the way of dealing with multiclass data sets, that is, data sets comprised of more than two class types. Since SVM only deals with one binary classification at a time, we need to use a decomposition scheme when applying it to multiclass data sets. We employ one-against-others (Bottou et al., 1994) in our experiment. In other words, if there are m class types in total, we train m SVM classifiers, each of which classifies a sample as A or not A, where A is one of the m class types. One-against-one (Knerr et al., 1990; Platt et al., 2000) is an alternative decomposition scheme that allows us to train $m(m\text{-}1)/2$ classifiers. In our experience, the one-against-others scheme usually yields comparable or better accuracy rates than the one-against-one approach; however, the training cost is higher. For APLs, on the other hand, we construct prototypes for all class types simultaneously. Thus, in our experiments, there is *no* decomposition scheme for APLs.

The accuracy rates and training times of all the methods are given in Table 11. The boldface numbers have the same meaning as before, while the underlined numbers are the accuracy rates that are lower than the corresponding SVM results. As usual, we list the averaged results over all the 12 data sets at the bottom of the table. From the last results, we observe that all the APLs outperform *k*-NN in terms of accuracy; and GCNN is faster in training than SVM, but it is less accurate. The other three APLs incur higher training costs than SVM, but yield higher accuracy rates.

8 Conclusion

We have proposed a number of adaptive prototype learning algorithms that construct prototypes out of training samples. They differ in the use of samples or the weighted averages of samples as prototypes, and in the use of the Euclidean distance or a kernel-based distance. The algorithms can be further strenghened by allowing a non-zero training error rate, which improves the test accuracy. Our experiments, in which four types of APL were applied to 12 benchmark data sets, confirm the algorithms' efficacy in terms of test accuracy compared to many instance-based learning algorithms, the *k*-NN rule, and SVM.

DATA SET		APL Methods				Alternative Methods	
		GCNN	*f*-LAPL	*c*-KAPL	*f*-KAPL	*k*-NN	SVM
Iris	AR	96.62	97.95	98.63	98.40	97.03	96.47
	TT	0.8	30	16,225	81,334		70.52
Wine	AR	98.06	99.02	99.02	99.56	97.64	98.97
	TT	1	144	24,378	175,647		90.88
Glass	AR	69.39	71.26	72.23	72.73	70.40	69.43
	TT	1.37	314	18,906	108,891		299.68
Ionosphere	AR	89.07	91.46	95.88	95.87	86.72	95.08
	TT	9.45	8,010	399,693	3,078,420		362.20
Cancer	AR	97.5	97.79	97.50	97.79	96.91	97.06
	TT	3.34	2,301	496,817	5,265,013		321.92
Zoo	AR	97.66	97.66	97.66	97.66	96.55	95.86
	TT	0.83	11	21,666	135,346		153.64
Heart	AR	85.57	86.90	85.83	86.43	83.77	84.83
	TT	1.56	1,134	72,925	607,436		130.00
TAE	AR	63.21	62.47	65.22	65.61	57.78	64.23
	TT	0.95	229	18,682	133,157		605.24
BLD	AR	65.93	67.34	67.72	70.52	63.90	71.19
	TT	2.4	3,379	232,211	1,378,124		1181.68
New Thyroid	AR	97.31	97.76	98.57	99.05	96.31	97.78
	TT	0.92	135	19,289	134,671		78.80
SPECTF	AR	83.55	85.63	86.13	87.04	79.90	81.48
	TT	9.1	8,820	167,428	1,363,339		143.88
Ecoli	AR	86.44	86.81	86.18	87.06	87.33	88.13
	TT	1.32	920	35,151	216,235		421.88
AVERAGE	AR	85.86	86.84	87.55	88.14	84.52	86.71
	TT	2.3	2,119	126,948	1,056,468		321.72

Table 11. The performance of the four APLs, *k*-NN and SVM.

Acknowledgements

We wish to thank Po-Han Kuo and Chien-Hsing Chou for providing some of the experimental results.

References

P. Bartlett and J. Shaw-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1999.

J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, New York, 1981.

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: A case study in handwriting digit recognition. In *Proc. Int. Conf. Pattern Recognition*, pages 77–87, 1994.

H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. In *Data Mining and Knowledge Discovery*, 6:153–172, 2002.

C. Cortes and V. Vapnik. Support vector machines. In *Machine Learning*, 20:1-25, 1995.

T. Cover and P. Hart. Nearest neighbor pattern classification. In *IEEE Trans. Information Theory*, 13:21-27, 1967.

N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

F. Devi and M. Murty. An incremental prototype set building technique. In *Pattern Recognition*, 35:505–513, 2002.

L. Devroye and L. Györfi. *Nonparametric Density Estimation: The L1 View*. John Wiley, New York, 1985.

L. Devroye, L. Györfi, A. Krzyżak, and G. Lugosi. On the strong consistency of nearest neighbor regression function estimates. In *Annals of Statistics*, 22:1371-1385, 1994.

L. Devroye, L Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

E. Fix and J. L. Hodges. Discriminatory analysis. Nonparametric discrimination: Consistency properties. Technical Report 4, Project Number 21-49-004, USAF School of Aviation Medicine. Randolph Field, Texas, 1951.

E. Fix and J. L. Hodges. Discriminatory analysis: small sample performance. Technical Report 11, Project Number 21-49-004, USAF School of Aviation Medicine. Randolph Field, Texas, 1952.

E. Fix and J. L. Hodges. Discriminatory analysis. Nonparametric discrimination: Consistency properties. In B. Dasarathy, editor, *Nearest Neighbor Pattern Classification Techniques*, pages 32-39, IEEE Computer Society Press, Los Alamitos, 1991.

E. Fix and J. L. Hodges. Discriminatory analysis: small sample performance. In B. Dasarathy, editor, *Nearest Neighbor Pattern Classification Techniques*, pages 40-56, IEEE Computer Society Press, Los Alamitos, 1991.

F. Girosi. An equivalence between sparse approximation and support vector machines. In *Neural Computation*, 10(6):1455-1481, 1998.

P. Hart. The condensed nearest neighbor rule. In *IEEE Trans. Information Theory*, 14:515-516, 1968.

F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. In *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. John Wiley & Sons, New York, 1999.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. In *IEEE Transactions on Neural Networks*, 13(2):415-425, 2002.

D.-W. Kim, K.Y. Lee, D. Lee, K. H. Lee. Evaluation of the performance of clustering algorithms in kernel-induced feature space. In *Pattern Recognition*, 38:607-611, 2005.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*, Springer-Verlag, New York, 1990.

T. Kohonen. *Self-Organization and Associated Memory*. Springer-Verlag, Berlin, 1988.

T. Kohonen. Statistical pattern recognition revisited. In R. Eckmiller, editor, *Advanced Neural Computers*, North-Holland, Amsterdam, pages 137-144, 1990.

Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. In *Pattern Recognition*, 19:84-95, 1980.

S. Lloyd. Least squares quantization in PCM. In *IEEE Trans. Information Theory*, 28:129-137, 1982.

J. Max. Quantizing for minimum distortion. In *IEEE Trans. Information Theory*, 6:7-12, 1960.

T. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. In *Philosophical Transactions of the Royal Society of London*, Series A, 209:415-446, 1909.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI Repository of machine learning databases, Irvine, CA: University of California, Department of Information and Computer Science, 1998. [http://www.ics.uci.edu/~mlearn/ MLRepository.html].

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAG's for multiclass classification. In *Advances in Neural Information Processing Systems*, 12:547-553, MIT Press, Cambridge, 2000.

S. Salzberg. A nearest hyperrectangle learning method. In *Machine Learning*, 6:251-276, 1991.

B. Schölkopf, C. J. C. Burges, and A. J. Smola. In *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1999.

C. Stone. Consistent nonparametric regression. In *Annals of Statistics*, 5:595-645, 1977.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Theory of Probability and Its Applications*, 16:264-280, 1971.

V. Vapnik and A. Chervonenkis. Ordered risk minimization I. In *Automation and Remote Control*, 35:1226-1235, 1974.

V. Vapnik and A. Chervonenkis. Ordered risk minimization II. In *Automation and Remote Control*, 35:1403-1412, 1974.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

D. R. Wilson and T. R. Martinez. Reduction techniques with instance-based learning algorithms. In *Machine Learning*, 38:257–286, 2000.

Z.-D. Wu, W.-X. Xie and J.-P. Yu. Fuzzy c-means clustering algorithm based on kernel method. In *Fifth Intern. Conf. Computational Intelligence and Multimedia Applications*, pages 1–6, 2003.

L. Zhao. Exponential bounds of mean error for the nearest neighbor estimates of regression functions. In *Journal of Multivariate Analysis*, 21:168-178, 1987.

Journal of Machine Learning Research 7 (2006) 2149-2187 Submitted 3/06; Revised 7/06; Published 10/06

A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests

Luis M. de Campos LCI@DECSAI.UGR.ES
Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S.I. Informática y de Telecomunicaciones, Universidad de Granada
18071-Granada, Spain

Editor: Nir Friedman

Abstract

We propose a new scoring function for learning Bayesian networks from data using score+search algorithms. This is based on the concept of mutual information and exploits some well-known properties of this measure in a novel way. Essentially, a statistical independence test based on the chi-square distribution, associated with the mutual information measure, together with a property of additive decomposition of this measure, are combined in order to measure the degree of interaction between each variable and its parent variables in the network. The result is a non-Bayesian scoring function called MIT (mutual information tests) which belongs to the family of scores based on information theory. The MIT score also represents a penalization of the Kullback-Leibler divergence between the joint probability distributions associated with a candidate network and with the available data set. Detailed results of a complete experimental evaluation of the proposed scoring function and its comparison with the well-known K2, BDeu and BIC/MDL scores are also presented.

Keywords: Bayesian networks, scoring functions, learning, mutual information, conditional independence tests

1. Introduction

Nowadays, Bayesian networks (Jensen, 1996; Pearl, 1988) constitute a widely accepted formalism for representing knowledge with uncertainty and efficient reasoning. A Bayesian network comprises a qualitative and a quantitative component. While the qualitative part represents structural information about a problem domain, in the form of causality, relevance or (in)dependence relationships between variables, the quantitative part (which allows us to introduce uncertainty into the model) represents probability distributions that quantify these relationships. Once a complete Bayesian network has been built, it is an efficient tool for performing inferences. However, there still remains the previous problem of building such a network, that is, to provide the graph structure and the numerical parameters necessary for characterizing it. As it may be difficult and time-consuming to build Bayesian networks using the method of eliciting opinions from domain experts, and given the increasing availability of data in many domains, directly learning Bayesian networks from data is an interesting alternative.

There are many learning algorithms for automatically building Bayesian networks from data. Although some of these are based on testing conditional independences, in this paper we are more interested in those algorithms based on the so-called *score+search* paradigm. These see the learning task as a combinatorial optimization problem, where a search method operates on a search space

associated with Bayesian networks, the search being guided by a scoring function that evaluates the degree of fitness between each element in this space and the available data.

The aim of this work is to define and study a new scoring function to be used by this class of Bayesian network learning algorithms as a competitive alternative to existing scoring functions (Bouckaert, 1993, 1995; Buntine, 1991; Chow and Liu, 1968; Cooper and Herskovits, 1992; Friedman and Goldszmidt, 1996; Heckerman et al., 1995; Herskovits and Cooper, 1990; Lam and Bacchus, 1994; Suzuki, 1993). We also want to empirically evaluate the merits of the new score by means of a comparative experimental study.

The proposed scoring function is based on the concept of mutual information. This measure has several interesting properties, the most important for our purposes being the possibility of building a statistical test of independence based on the chi-square distribution. Mutual information has already been used either directly or indirectly within Bayesian network learning algorithms based on score and search (Bouckaert, 1993; Chow and Liu, 1968; Lam and Bacchus, 1994). The associated statistical test has also been used by several learning algorithms based on conditional independence tests (Acid and de Campos, 2001; Cheng et al., 2002; de Campos and Huete, 2000; Spirtes et al., 1993). However, what is new is the simultaneous quantification of the results of a set of independence tests based on mutual information. Basically, we use mutual information in order to measure the degree of interaction between each variable and its parent variables in the network, but penalizing this value using a term related to the chi-square distribution. This penalization term takes into account not only the network complexity but also its reliability. The result will undoubtedly be a scoring function, but any score+search-based algorithm using it will have some similarities with the learning methods based on independence tests (although we believe that our scoring function makes better use of the information provided by the tests than these methods). To a certain extent what we are proposing is a hybrid algorithm (either an algorithm based on *scoring independences and search* or an algorithm based on *quantitative conditional independence tests*).

Sections 2 and 3 of this paper provide some background about learning Bayesian networks and types of scoring functions, respectively. Section 4 covers the development of the new scoring function, which we shall call MIT (mutual information tests). Section 5 carries out an empirical comparative study of MIT against several state-of-the-art scoring functions (K2, BDeu and BIC/MDL). We first define the performance measures to be used and we then describe the corresponding experimental designs and the obtained results. Section 6 contains our conclusions and some proposals for future research. Finally, Appendix A includes proof of all the theorems set out in the paper.

2. Learning Bayesian Networks

Let us consider a finite set $\mathbf{U_n} = \{X_1, X_2, \ldots, X_n\}$ of discrete random variables.[1] A generic variable of the set $\mathbf{U_n}$ will be denoted as either X_i or X. The domain of each variable X_i is a finite set $V_i = \{x_{i1}, \ldots, x_{ir_i}\}$. A generic element of V_i will be denoted as x_i. In general, we shall use uppercase letters to denote variables, lowercase letters to denote states of the variables, and bold-faced letters (either uppercase or lowercase) to denote sets (of either variables or states of the variables, respectively).

A Bayesian network (BN) is a graphical representation of a joint probability distribution (Pearl, 1988) that includes two components:

1. Although there are also Bayesian networks with continuous variables, here we are only interested in the case where all the variables are discrete.

- First, a *directed acyclic graph* (DAG) $G = (\mathbf{U_n}, E_G)$, where $\mathbf{U_n}$, the set of nodes, represents the system variables,[2] and E_G, the set of arcs, represents direct dependency relationships between variables; the absence of arcs linking pairs of variables in turn represents the existence of conditional independence relationships between these variables. A conditional independence relationship between two variables X_i and X_j, given a subset of variables $\mathbf{Z}$, denoted as $I(X_i, X_j|\mathbf{Z})$, means that given the values of the variables in $\mathbf{Z}$, our degree of belief about the possible values of X_i is not modified once we know the value of variable X_j: $p(x_i|x_j, \mathbf{z}) = p(x_i|\mathbf{z})$. Each variable $X_i \in \mathbf{U_n}$ has an associated *parent set* in the graph G, $Pa_G(X_i) = \{X_j \in \mathbf{U_n} \mid X_j \rightarrow X_i \in E_G\}$. If X_i has no parent (it is a root node), then $Pa_G(X_i) = \emptyset$.

- The second component is a set of numerical parameters, which usually represent conditional probability distributions: for each variable X_i in $\mathbf{U_n}$, we store a family of conditional distributions $p(X_i|pa_G(X_i))$, one for each possible *configuration*,[3] $pa_G(X_i)$, of the parent set of X_i in the graph. If X_i has no parent, then $p(X_i|pa_G(X_i))$ equals $p(X_i)$. From these conditional distributions, we can obtain the joint distribution over $\mathbf{U_n}$ using:

$$p(x_1, x_2, \ldots, x_n) = \prod_{X_i \in \mathbf{U_n}} p(x_i|pa_G(X_i))$$

The problem of learning Bayesian networks from data consists in finding the BN that (according to certain criterion) best fits the available data. This problem has been studied in depth over the last ten years and consequently, there are currently a considerable number of learning algorithms. As Bayesian networks have two different components (the graphical and the numerical model), the algorithms for learning BNs must deal with two different but highly related tasks: learning the structure (the DAG) and learning the parameters (the conditional probabilities). These two tasks cannot be carried out completely independently: on the one hand, in order to estimate the conditional probabilities, we must know the graphical structure; on the other, in order to determine whether the graph we are trying to find contains certain arcs, we need to estimate certain statistics from the data which, depending on the kind of learning algorithm being used, will be employed either to carry out some conditional independence tests or to measure the intensity of the relationships between the nodes involved in these arcs.

In this paper, we are only interested in algorithms for learning the structure of Bayesian networks. As we mentioned previously, most of these algorithms can be grouped into two different categories: methods based on *conditional independence tests* (also called *constraint-based* methods) and methods based on *scoring functions and search*, although there are also algorithms that use a combination of independence-based and scoring-based methods with different hybridization strategies (Acid and de Campos, 2000, 2001; Dash and Druzdzel, 1999; de Campos et al., 2003; Singh and Valtorta, 1995; Spirtes and Meek, 1995).

The algorithms based on independence tests (Cheng et al., 2002; de Campos, 1998; de Campos and Huete, 2000; Meek, 1995; Pearl and Verma, 1991; Spirtes et al., 1993; Verma and Pearl, 1990; Wermuth and Lauritzen, 1983) perform a qualitative study of the dependence and independence relationships between the variables in the domain (obtained from the data by means of conditional independence tests), and attempt to find a network that represents these relationships as far as possible. Two fundamental issues for these algorithms are the number and the complexity of

2. In the same way, we shall represent a variable and its associated node in the graph.
3. A configuration of a set of variables $\mathbf{Z}$ is an assignment of values to each of the variables in $\mathbf{Z}$.

the independence tests, and this can also cause unreliable results. Nevertheless, constraint-based algorithms generally come with rigorous theoretical founding and have developed a body of work that details sound and complete methods to make use of independence relations in the data while correctly accounting for structure.

The algorithms based on a scoring function attempt to find a graph that maximizes the selected score, which is usually defined as a measure of fitness between the graph and the data. All of them use the scoring function in combination with a search method in order to measure the goodness of each explored structure from the space of feasible solutions. Different learning algorithms are obtained depending on the search procedure used, as well as on the definitions of the scoring function and the search space.

The scoring functions are based on different principles, such as entropy and information (Chow and Liu, 1968; Herskovits and Cooper, 1990), the minimum description length (Bouckaert, 1993, 1995; Friedman and Goldszmidt, 1996; Lam and Bacchus, 1994; Suzuki, 1993), or Bayesian approaches (Buntine, 1991; Cooper and Herskovits, 1992; Heckerman et al., 1995; Kayaalp and Cooper, 2002). The most usual scoring functions will be described later in more detail.

As far as the search is concerned, although the most frequently used are local search methods (Buntine, 1991; Chickering et al., 1995; Cooper and Herskovits, 1992; de Campos et al., 2003; Heckerman et al., 1995) due to the exponentially large size of the search space, there is a growing interest in other heuristic search methods such as simulated annealing (Chickering et al., 1995), tabu search (Acid and de Campos, 2003; Bouckaert, 1995), branch and bound (Tian, 2000), genetic algorithms and evolutionary programming (Larrañaga et al., 1996; Myers et al., 1999; Wong et al., 1999), Markov chain Monte Carlo (Kocka and Castelo, 2001; Myers et al., 1999), variable neighborhood search (de Campos and Puerta, 2001a), ant colony optimization (de Campos et al., 2002), greedy randomized adaptive search procedures (GRASP) (de Campos et al., 2002), and estimation of distribution algorithms (Blanco et al., 2003).

Most learning algorithms employ different search methods but the same search space: the DAG space. Possible alternatives are the space of the orderings of the variables (de Campos et al., 2002; de Campos and Huete, 2002; de Campos and Puerta, 2001b; Friedman and Koller, 2003; Larrañaga et al., 1996), with a secondary search in the DAG space compatible with a given ordering; the space of *essential graphs* (Pearl and Verma, 1990) (also called *patterns* or *completed PDAGs*), which are partially directed acyclic graphs[4] or PDAGs that canonically represent equivalence classes of DAGs (Andersson et al., 1997; Chickering, 2002; Dash and Druzdzel, 1999; Madigan et al., 1996; Spirtes and Meek, 1995); and the space of *RPDAGs* (restricted PDAGs), which also represent equivalence classes of DAGs (Acid and de Campos, 2003; Acid et al., 2005).

3. Scoring Functions for Learning Bayesian Networks

Focusing on the methods for learning Bayesian networks based on the score+search paradigm, the problem can be formally expressed as follows: given a *complete*[5] training data set $D = \{\mathbf{u}^1, \ldots, \mathbf{u}^N\}$ of instances of $\mathbf{U_n}$, find a DAG G^* such that

$$G^* = \arg\max_{G \in \mathcal{G}_n} g(G : D),$$

4. Containing both directed (arcs) and undirected (links) edges.
5. We consider neither missing values nor latent variables.

where $g(G:D)$ is the scoring function measuring the degree of fitness of any candidate DAG G to the data set, and $\mathcal{G}_n$ is the family of all the DAGs defined on $\mathbf{U_n}$.

The learning algorithms that search in the DAG space with local search-based methods can be more efficient if the scoring function being used has the property of *decomposability*: a scoring function g is *decomposable* if the value assigned to each structure can be expressed as a sum (in the logarithmic space) of local values that depend only on each node and its parents:

$$g(G:D) = \sum_{X_i \in \mathbf{U_n}} g(X_i, Pa_G(X_i) : D)$$

$$g(X_i, Pa_G(X_i) : D) = g(X_i, Pa_G(X_i) : N^D_{X_i, Pa_G(X_i)}),$$

where $N^D_{X_i, Pa_G(X_i)}$ are the sufficient statistics of the set of variables $\{X_i\} \cup Pa_G(X_i)$ in D, that is, the number of instances in D corresponding to each possible configuration of $\{X_i\} \cup Pa_G(X_i)$.

For example, a search procedure that only changes one arc at each move can efficiently evaluate the improvement obtained by this change. It can reuse most of the previous computations and only the statistics for the variables whose parent sets have been modified must be recomputed. In this way, the insertion or deletion of an arc $X_j \to X_i$ in a DAG G can be evaluated by computing only one new local score, $g(X_i, Pa_G(X_i) \cup \{X_j\} : D)$ or $g(X_i, Pa_G(X_i) \setminus \{X_j\} : D)$, respectively; the reversal of an arc $X_j \to X_i$ requires the evaluation of two new local scores, $g(X_i, Pa_G(X_i) \setminus \{X_j\} : D)$ and $g(X_j, Pa_G(X_j) \cup \{X_i\} : D)$.

Another property which is particularly interesting if the learning algorithm searches in a space of equivalence classes of DAGs is called the *score equivalence*: a scoring function g is *score-equivalent* if it assigns the same value to all DAGs that are represented by the same essential graph. In this way, the result of evaluating an equivalence class will be the same regardless of which DAG from this class is selected.

There are different ways to measure the degree of fitness of a DAG with respect to a data set. Most can be grouped into two categories: Bayesian and information measures. We shall use the following notation: the number of states of the variable X_i is r_i; the number of possible configurations of the parent set $Pa_G(X_i)$ of X_i is q_i; obviously, $q_i = \prod_{X_j \in Pa_G(X_i)} r_j$; w_{ij}, $j = 1, \ldots q_i$, represents a configuration of $Pa_G(X_i)$; N_{ijk} is the number of instances in the data set D where the variable X_i takes the value x_{ik} and the set of variables $Pa_G(X_i)$ take the value w_{ij}; N_{ij} is the number of instances in the data set where the variables in $Pa_G(X_i)$ take their j-th configuration w_{ij}; obviously $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$; similarly, N_{ik} is the number of instances in D where the variable X_i takes its k-th value x_{ik}, and therefore $N_{ik} = \sum_{j=1}^{q_i} N_{ijk}$; the total number of instances in D is N.

3.1 Bayesian Scoring Functions

Starting from a prior probability distribution on the possible networks, the general idea is to compute the posterior probability distribution conditioned to the available data D, $p(G|D)$. The best network is the one that maximizes the posterior probability. It is not in fact necessary to compute $p(G|D)$ and for comparative purposes, computing $p(G,D)$ is sufficient since the term $p(D)$ is the same for all the possible networks. As it is easier to work in the logarithmic space, in practice, the scoring functions use the value $\log(p(G,D))$ instead of $p(G,D)$.

One of the first Bayesian scoring functions, called K2, was proposed by Cooper and Herskovits (1992). It relies on several assumptions (multinomiality, lack of missing values, parameter independence, parameter modularity, uniformity of the prior distribution of the parameters given the

network structure), and can be expressed as follows:

$$g_{K2}(G:D) = \log(p(G)) + \sum_{i=1}^{n}\left[\sum_{j=1}^{q_i}\left[\log\left(\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\right) + \sum_{k=1}^{r_i}\log\left(N_{ijk}!\right)\right]\right], \quad (1)$$

where $p(G)$ represents the prior probability of the DAG G. Afterwards, the so-called BD (Bayesian Dirichlet) score was proposed by Heckerman et al. (1995) as a generalization of K2:

$$g_{BD}(G:D) = \log(p(G)) + \sum_{i=1}^{n}\left[\sum_{j=1}^{q_i}\left[\log\left(\frac{\Gamma(\eta_{ij})}{\Gamma(N_{ij}+\eta_{ij})}\right) + \sum_{k=1}^{r_i}\log\left(\frac{\Gamma(N_{ijk}+\eta_{ijk})}{\Gamma(\eta_{ijk})}\right)\right]\right], \quad (2)$$

where the values η_{ijk} are the hyperparameters for the Dirichlet prior distributions of the parameters given the network structure, and $\eta_{ij} = \sum_{k=1}^{r_i}\eta_{ijk}$. $\Gamma(.)$ is the function *Gamma*, $\Gamma(c) = \int_0^\infty e^{-u}u^{c-1}du$. It should be noted that if c is an integer, $\Gamma(c) = (c-1)!$. If the values of all the hyperparameters are $\eta_{ijk} = 1$, we obtain the K2 score as a particular case of BD.

In practical terms, the specification of the hyperparameters η_{ijk} is quite difficult (except if we use non-informative assignments, as the ones employed by K2). However, by considering the additional assumption of likelihood equivalence (Heckerman et al., 1995), it is possible to specify the hyperparameters relatively easily. While the result is a scoring function called BDe (and its expression is identical to the BD one in Equation 2), the hyperparameters can now be computed in the following way:

$$\eta_{ijk} = \eta \times p(x_{ik}, w_{ij}|G_0),$$

where $p(.|G_0)$ represents a probability distribution associated with a *prior Bayesian network* G_0 and η is a parameter representing the equivalent sample size.

A particular case of BDe which is especially interesting appears when $p(x_{ik}, w_{ij}|G_0) = \frac{1}{r_i q_i}$, that is, the prior network assigns a uniform probability to each configuration of $\{X_i\} \cup Pa_G(X_i)$. The resulting score is called BDeu, which was originally proposed by Buntine (1991). This score only depends on one parameter, the equivalent sample size η, and is expressed as follows:

$$g_{BDeu}(G:D) = \log(p(G)) + \sum_{i=1}^{n}\left[\sum_{j=1}^{q_i}\left[\log\left(\frac{\Gamma(\frac{\eta}{q_i})}{\Gamma(N_{ij}+\frac{\eta}{q_i})}\right) + \sum_{k=1}^{r_i}\log\left(\frac{\Gamma(N_{ijk}+\frac{\eta}{r_i q_i})}{\Gamma(\frac{\eta}{r_i q_i})}\right)\right]\right]. \quad (3)$$

Regarding the term $\log(p(G))$ which appears in all the previous expressions, it is quite common to assume a uniform distribution (except if we really have information about the greater desirability of certain structures) so that it becomes a constant and can be removed.

3.2 Scoring Functions based on Information Theory

These scoring functions represent another option for measuring the degree of fitness of a DAG to a data set and are based on codification and information theory concepts. Coding attempts to reduce as much as possible the number of elements which are necessary to represent a message (depending on its probability). Frequent messages will therefore have shorter codes whereas larger codes will be assigned to the less frequent messages. The minimum description length principle (MDL) selects the coding that requires minimum length to represent the messages. Another more general formulation of the same idea establishes that in order to represent a data set with one model from a specific type, the best model is the one that minimizes the sum of the description length

of the model and the description length of the data given the model. Complex models usually require greater description lengths but reduce the description length of the data given the model (they are more accurate). On the other hand, simple models require shorter description lengths but the description length of the data given the model increases. The minimum description length principle establishes an appropriate trade-off between complexity and precision.

In our case, the data set to be represented is D and the selected class of models are Bayesian networks. Therefore, the description length includes the length required to represent the network plus the length necessary to represent the data given the network (Bouckaert, 1993, 1995; Friedman and Goldszmidt, 1996; Lam and Bacchus, 1994; Suzuki, 1993). In order to represent the network, we must store its probability values, and this requires a length which is proportional to the number of free parameters of the factorized joint probability distribution.[6] This number, called network complexity and denoted as $C(G)$, is:

$$C(G) = \sum_{i=1}^{n} (r_i - 1) q_i.$$

The usual proportionality factor is $\frac{1}{2}\log(N)$ (Rissanen, 1986). Therefore, the description length of the network is:

$$\frac{1}{2} C(G) \log(N).$$

Regarding the description of the data given the model, by using Huffmann codes its length turns out to be the negative of the log-likelihood, that is, the logarithm of the likelihood function of the data with respect to the network. This value is minimum for a fixed network structure when the network parameters are estimated from the data set itself by using maximum likelihood. The log-likelihood can be expressed in the following way (Bouckaert, 1995):

$$LL_D(G) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \left(\frac{N_{ijk}}{N_{ij}} \right). \tag{4}$$

Therefore, the MDL scoring function (by changing the signs to deal with a maximization problem) is:

$$g_{MDL}(G:D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \left(\frac{N_{ijk}}{N_{ij}} \right) - \frac{1}{2} C(G) \log(N). \tag{5}$$

Another way of measuring the quality of a Bayesian network is to use measures based on information theory and some of these are closely related with the previous one. The basic idea is to select the network structure that best fits the data, penalized by the number of parameters which are necessary to specify the joint distribution. This leads to a generalization of the scoring function in Equation 5:

$$g(G:D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \left(\frac{N_{ijk}}{N_{ij}} \right) - C(G) f(N), \tag{6}$$

where $f(N)$ is a non-negative penalization function. If $f(N) = 1$, the score is based on the Akaike information criterion (AIC) (Akaike, 1974). If $f(N) = \frac{1}{2}\log(N)$, then the score, called BIC, is

6. There are other versions (Lam and Bacchus, 1994) that also include the description length of the graph itself, which is proportional to the sum of the number of parents for each node, $\sum_{i=1}^{n} |Pa_G(X_i)|$. However, the most usual formulation does not consider it.

based on the Schwarz information criterion (Schwarz, 1978), which coincides with the MDL score. If $f(N) = 0$, we have the maximum likelihood score, although this is not very useful as the best network using this criterion is always a complete network which includes all the possible arcs.

It is interesting to note that another way of expressing the log-likelihood in Equation 4 is:

$$LL_D(G) = -N \sum_{i=1}^{n} H_D(X_i|Pa_G(X_i)), \tag{7}$$

where $H_D(X_i|Pa_G(X_i))$ represents the conditional entropy of the variable X_i given its parent set $Pa_G(X_i)$, for the probability distribution p_D:

$$H_D(X_i|Pa_G(X_i)) = \sum_{j=1}^{q_i} p_D(w_{ij}) \left(- \sum_{k=1}^{r_i} p_D(x_{ik}|w_{ij}) \log(p_D(x_{ik}|w_{ij})) \right),$$

and p_D is the joint probability distribution associated with the data set D, obtained from the data by maximum likelihood. The log-likelihood $LL_D(G)$ can also be expressed as follows (Bouckaert, 1995):

$$LL_D(G) = -NH_D(G),$$

where $H_D(G)$ represents the entropy of the joint probability distribution associated with the graph G when the network parameters are estimated from D by maximum likelihood:

$$H_D(G) = - \sum_{x_1,\ldots,x_n} \left(\left(\prod_{i=1}^{n} p_D(x_i|pa_G(X_i)) \right) \log \left(\prod_{i=1}^{n} p_D(x_i|pa_G(X_i)) \right) \right).$$

Therefore, another interpretation of the scoring functions based on information is that they attempt to minimize the conditional entropy of each variable given its parents, and so they search for the parent set of each variable that gives as much information as possible about this variable (or which most restricts the distribution). It is necessary to add a penalization term since the minimum conditional entropy is always obtained after adding all the possible variables to the parent set.

An alternative way to avoid this overfitting without using a penalization function was proposed by Herskovits and Cooper (1990) who used the maximum likelihood score, but the process of inserting arcs into the network was stopped by means of a statistical test, which determined whether the difference in entropy between the current network and the one obtained by including an additional arc was statistically significant.

With respect to the characteristics of the different scoring functions, all are decomposable and with the exception of K2 and BD, they are also score-equivalent (Chickering, 1995).

4. A New Scoring Function based on Mutual Information and Independence Tests

In order to explain the ideas behind the proposed scoring function more clearly, we shall first introduce several preliminary considerations. These will lead to a first version of the scoring function, which will be later refined in order to obtain the final version.

4.1 Preliminary Considerations

Our goal is to design a scoring function in such a way that the value $g(G:D)$ represents a measure of the distance between the joint probability distribution associated with the DAG G, p_G, and the

joint probability distribution associated with the data, p_D. We should mention that p_G must be understood to be the joint probability distribution that factorizes according to G and whose local conditional probability distributions are estimated from D by means of maximum likelihood, that is,

$$p_G(x_1,\ldots,x_n) = \prod_{i=1}^{n} p_D(x_i|pa_G(X_i)).$$

A reasonable choice for the distance measure is the *Kullback-Leibler divergence* (Kullback, 1968):

$$KL(p_D,p_G) = \sum_{x_1,\ldots,x_n} p_D(x_1,\ldots,x_n)\log\left(\frac{p_D(x_1,\ldots,x_n)}{p_G(x_1,\ldots,x_n)}\right).$$

This distance can also be expressed in another more convenient way:

$$\begin{aligned} KL(p_D,p_G) &= -H_D(\{X_1,\ldots,X_n\}) + \sum_{\substack{i=1 \\ Pa_G(X_i)=\emptyset}}^{n} H_D(X_i) \\ &\quad + \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} \Big(H_D(\{X_i\}\cup Pa_G(X_i)) - H_D(Pa_G(X_i))\Big), \end{aligned} \tag{8}$$

where $H_D(\mathbf{X})$ represents the entropy of the set of variables $\mathbf{X}$ with respect to the distribution p_D.

We shall now consider the concept of *mutual information*. Given a probability distribution p defined over two sets of variables $\mathbf{X}$ and $\mathbf{Y}$, the mutual information between $\mathbf{X}$ and $\mathbf{Y}$ is:

$$MI(\mathbf{X},\mathbf{Y}) = \sum_{\mathbf{x},\mathbf{y}} p(\mathbf{x},\mathbf{y})\log\left(\frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})}\right),$$

which can also be expressed in terms of entropy as:

$$MI(\mathbf{X},\mathbf{Y}) = H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{X}\cup\mathbf{Y}). \tag{9}$$

Mutual information (which is simply the Kullback-Leibler divergence between the joint distribution for $\mathbf{X}$ and $\mathbf{Y}$ and the product of the corresponding marginals) can be considered as a way of measuring the dependence degree between the sets of variables $\mathbf{X}$ and $\mathbf{Y}$, which is null when the two sets of variables are independent and maximum when they are functionally dependent. By using Equation 9, we can rewrite Equation 8 as follows (Lam and Bacchus, 1994):

$$KL(p_D,p_G) = -H_D(\{X_1,\ldots,X_n\}) + \sum_{i=1}^{n} H_D(X_i) - \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} MI_D(X_i,Pa_G(X_i)). \tag{10}$$

As the two first terms in Equation 10 do not depend on the DAG G being considered, we obtain:

$$\arg\min_{G\in\mathcal{G}_n} KL(p_D,p_G) = \arg\max_{G\in\mathcal{G}_n} \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} MI_D(X_i,Pa_G(X_i)), \tag{11}$$

and therefore minimizing the Kullback-Leibler divergence is equivalent to maximizing the sum of the measures of mutual information between each variable and its parent variables in the graph.

We have still not achieved anything useful, however, since mutual information has the property that $MI(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}) \geq MI(\mathbf{X}, \mathbf{Y})$, in other words, mutual information always increases by including additional variables. Therefore, the complete network will always have minimum Kullback-Leibler divergence with respect to the data. In fact, by taking into account Equation 7 and the relation between mutual information and conditional entropy, namely $MI(\mathbf{X}, \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y})$, we can write:

$$\sum_{\substack{i=1 \\ Pa_G(X_i) \neq \emptyset}}^{n} MI_D(X_i, Pa_G(X_i)) = \frac{LL_D(G)}{N} + \sum_{i=1}^{n} H_D(X_i). \tag{12}$$

Therefore, minimizing the Kullback-Leibler divergence is also equivalent to maximizing log-likelihood. The following expression is equivalent to the previous one:

$$\sum_{\substack{i=1 \\ Pa_G(X_i) \neq \emptyset}}^{n} MI_D(X_i, Pa_G(X_i)) = \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \left(\frac{N N_{ijk}}{N_{ik} N_{ij}} \right).$$

However, there are certain advantages to using mutual information instead of log-likelihood as we shall see later. First, let us consider the concept of *conditional mutual information* between $\mathbf{X}$ and $\mathbf{Y}$ given a set of variables $\mathbf{Z}$, defined as:

$$MI(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = \sum_{\mathbf{z}} \left(p(\mathbf{z}) \sum_{\mathbf{x},\mathbf{y}} p(\mathbf{x}, \mathbf{y}|\mathbf{z}) \log \left(\frac{p(\mathbf{x}, \mathbf{y}|\mathbf{z})}{p(\mathbf{x}|\mathbf{z}) p(\mathbf{y}|\mathbf{z})} \right) \right),$$

which can be expressed by $MI(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = H(\mathbf{X}|\mathbf{Z}) - H(\mathbf{X}|\mathbf{Y} \cup \mathbf{Z})$, and also by:

$$MI(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = H(\mathbf{X} \cup \mathbf{Z}) + H(\mathbf{Y} \cup \mathbf{Z}) - H(\mathbf{Z}) - H(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}).$$

The following property[7] of conditional mutual information is important for our purposes:

$$MI(\mathbf{X}, \mathbf{Y} \cup \mathbf{W}|\mathbf{Z}) = MI(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) + MI(\mathbf{X}, \mathbf{W}|\mathbf{Z} \cup \mathbf{Y}). \tag{13}$$

Another fundamental property of mutual information is:

Theorem 1 (Kullback, 1968) *Given a data set D with N elements, if the hypothesis that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$ is true, then the statistics $2N\,MI_D(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ approximates to a distribution $\chi^2(l)$ (Chi-square) with $l = (r_{\mathbf{X}} - 1)(r_{\mathbf{Y}} - 1)r_{\mathbf{Z}}$ degrees of freedom, where $r_{\mathbf{X}}$, $r_{\mathbf{Y}}$ and $r_{\mathbf{Z}}$ represent the number of configurations for the sets of variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$, respectively. If $\mathbf{Z} = \emptyset$, the statistics $2N\,MI_D(\mathbf{X}, \mathbf{Y})$ approximates to a distribution $\chi^2(l)$ with $l = (r_{\mathbf{X}} - 1)(r_{\mathbf{Y}} - 1)$ degrees of freedom.*

4.2 Developing a New Scoring Function

The basic idea underlying the new scoring function that we shall propose is very simple: to use the mutual information $MI_D(X_i, Pa_G(X_i))$ in order to measure the degree of interaction between each variable X_i and its parents $Pa_G(X_i)$, as in Equation 11, but penalizing this value using a term related

7. It should be noted that this property is a numeric version of the properties of *decomposition*, *weak union* and *contraction* of the probabilistic independence relationships and other dependence models (Pearl, 1988). These three properties, together with *symmetry*, characterize the dependence models called *semi-graphoids*.

to the χ^2 distribution. This term attempts to re-scale the mutual information values in order to prevent these values from systematically increasing as the number of variables in $Pa_G(X_i)$ does.

In our opinion, one problem with the scoring functions based on information (Equation 6) is that they penalize log-likelihood globally, with a combination of the network complexity and a function that depends only on the number of instances. Since we believe that as the log-likelihood can be decomposed as a sum of components (each being associated with a variable and its parents), then each of these components should be penalized differently, depending not only on its complexity but also on its *reliability*. For example, a DAG where a variable X_i has many parents is always penalized in the same way, without taking into account to what extent this topology is actually necessary to adequately and reliably represent the distribution for X_i. The scoring function that we shall propose naturally incorporates this kind of penalization, and is based on solid statistical grounds.

Given a DAG G, let us consider the mutual information between a variable X_i and its parents, $MI_D(X_i, Pa_G(X_i))$. Let s_i be the number of parent variables[8] of X_i, $s_i = |Pa_G(X_i)|$. Let us assume that $Pa_G(X_i) = \{X_{i1}, \ldots, X_{is_i}\}$. By iteratively applying Equation 13, we can express $MI_D(X_i, Pa_G(X_i))$ as:

$$\begin{aligned} MI_D(X_i, Pa_G(X_i)) &= MI_D(X_i, \{X_{i1}, \ldots, X_{is_i}\}) \\ &= MI_D(X_i, \{X_{i1}, \ldots, X_{i(s_i-1)}\}) + MI_D(X_i, X_{is_i} | \{X_{i1}, \ldots, X_{i(s_i-1)}\}) \\ &= MI_D(X_i, \{X_{i1}, \ldots, X_{i(s_i-2)}\}) + MI_D(X_i, X_{i(s_i-1)} | \{X_{i1}, \ldots, X_{i(s_i-2)}\}) + \\ &\quad MI_D(X_i, X_{is_i} | \{X_{i1}, \ldots, X_{i(s_i-1)}\}) = \ldots\ldots \\ &= MI_D(X_i, X_{i1}) + \sum_{j=2}^{s_i} MI_D(X_i, X_{ij} | \{X_{i1}, \ldots, X_{i(j-1)}\}). \end{aligned} \quad (14)$$

The elements in this decomposition of the mutual information will be interpreted as follows: starting with an empty set of parents of X_i, we have first included the arc $X_{i1} \rightarrow X_i$, and the degree of dependence between these variables is $MI_D(X_i, X_{i1})$. We then insert the arc $X_{i2} \rightarrow X_i$ and as X_{i1} is already a parent of X_i, the dependence degree between X_{i2} and X_i is $MI_D(X_i, X_{i2}|X_{i1})$. We continue inserting arcs in this way until the last one $X_{is_i} \rightarrow X_i$ (with a dependence degree between X_{is_i} and X_i equal to $MI_D(X_i, X_{is_i}|\{X_{i1}, \ldots, X_{i(s_i-1)}\})$) has been included. If we do not insert any additional arcs, this is because each remaining variable X_h does not contribute any additional information[9] with respect to X_i, this information being measured as $MI_D(X_i, X_h|\{X_{i1}, \ldots, X_{is_i}\})$. The key question is how to determine whether the values of mutual information represent an appreciable (i.e., statistically significant) amount of information. At this point, we can use the result in Theorem 1.

We know that $2NMI_D(X_i, X_{ij}|\{X_{i1}, \ldots, X_{i(j-1)}\})$ approximates to a distribution $\chi^2(l_{ij})$, with the appropriate degrees of freedom l_{ij}. Let us fix a *confidence level* α and determine the value $\chi_{\alpha,l_{ij}}$ such that $p(\chi^2(l_{ij}) \leq \chi_{\alpha,l_{ij}}) = \alpha$. This does in fact represent a statistical test of conditional independence: if $2NMI_D(X_i, X_{ij}|\{X_{i1}, \ldots, X_{i(j-1)}\}) \leq \chi_{\alpha,l_{ij}}$, then we accept the hypothesis of independence between X_i and X_{ij} given $\{X_{i1}, \ldots, X_{i(j-1)}\}$ (with probability α); otherwise we reject it.

The use of this kind of independence test within BN learning algorithms is quite frequent (Acid and de Campos, 2001; de Campos and Huete, 2000; Spirtes et al., 1993). It has also been used by algorithms based on score+search to stop the search process (Acid and de Campos, 2000; Herskovits and Cooper, 1990). The problem with an independence test is that it only asserts whether the

8. s_i should not be confused with q_i, which represents the number of configurations of these variables.

9. There may obviously be some variables that cannot be included as parents of X_i since they would create directed cycles in the graph.

variables are independent or not, rather than quantifying the extent to which they are. For example, if an algorithm is trying to decide which of the two variables X_j and X_k to exclude from the parent set of another variable X_i, if both variables turn out to be dependent on X_i (given its current parent set), the test is not able to discriminate between them, although it may be possible for one variable to be more closely dependent on X_i than the other.

Our proposal is to quantify the result of the independence test to build the scoring function. The difference $2NMI_D(X_i, X_{ij}|\{X_{i1},\ldots,X_{i(j-1)}\}) - \chi_{\alpha,l_{ij}}$ gives us a measure of the degree of interest for adding the variable X_{ij} to the current parent set of X_i: if the difference is negative (the test would say that X_i and X_{ij} are independent), the score will decrease, and the more clearly independent the variables are, the more it will decrease; when the difference is positive (the test would assert that these two variables are dependent), the score will increase, and the more dependent X_i and X_{ij} are, the more it will increase.

Therefore, a measure of the global quality of the set $Pa_G(X_i)$ as the parent set of variable X_i is:

$$\begin{aligned} g(X_i, Pa_G(X_i) : D) &= \sum_{j=2}^{s_i} \left(2N\,MI_D(X_i, X_{ij}|\{X_{i1},\ldots,X_{i(j-1)}\}) - \chi_{\alpha,l_{ij}}\right) \\ &\quad +2N\,MI_D(X_i, X_{i1}) - \chi_{\alpha,l_{i1}}, \end{aligned} \tag{15}$$

where $\chi_{\alpha,l_{ij}}$ is the value such that $p(\chi^2(l_{ij}) \leq \chi_{\alpha,l_{ij}}) = \alpha$, and the number of degrees of freedom is:

$$l_{ij} = \begin{cases} (r_i - 1)(r_{ij} - 1)\prod_{k=1}^{j-1} r_{ik} & j = 2,\ldots,s_i \\ (r_i - 1)(r_{i1} - 1) & j = 1\,. \end{cases} \tag{16}$$

The expression in Equation 15 is then a global quantification of a series of s_i simultaneous conditional independence tests, and by virtue of the decomposition of mutual information in Equation 14, it is equivalent to:

$$g(X_i, Pa_G(X_i) : D) = 2N\,MI_D(X_i, Pa_G(X_i)) - \sum_{j=1}^{s_i} \chi_{\alpha,l_{ij}}. \tag{17}$$

The scoring function would therefore be defined according to Equation 11 as:

$$g(G : D) = \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} \left(2N\,MI_D(X_i, Pa_G(X_i)) - \sum_{j=1}^{s_i} \chi_{\alpha,l_{ij}}\right). \tag{18}$$

It should be noted that although the value of mutual information will increase after new variables are added to the parent set, the penalization component (which contains one term for each parent variable) will also increase. In this way, we are able to appropriately re-scale the mutual information measure.

The value of α, which represents the confidence level associated with the statistical test, is a free parameter that may be fixed to any standard value (for example 0.90, 0.95 or 0.99). However, since we are in fact performing several simultaneous tests (as many as the number of variables in $Pa_G(X_i)$), and also taking into account the Bonferroni inequality,[10] in order for the *global* confidence level to be acceptable (that is to say, a reasonably high value of $p(\cap_{j=1}^{s_i}(\chi^2(l_{ij}) \leq \chi_{\alpha,l_{ij}}))$), it will be necessary for α to be greater than the standard values used when performing a single test.

10. $p(\cap_{i=1}^{n} A_i) \geq 1 - \sum_{i=1}^{n}\left(1 - p(A_i)\right)$, where A_i represent any events.

In order to accurately compute the values $\chi_{\alpha,l}$, we can use a standard method which is based on the algorithm proposed by Hill and Pike (1965, 1985) to compute the chi-squared integral (i.e., the probability $p(\chi^2(l) > x)$) in combination with a simple bisection search. Alternatively, if speed is more important than great accuracy, as the $\chi^2(l)$ distribution can be approximated by several transformations of the standardized normal distribution $N(0,1)$ for large degrees of freedom (Evans et al., 1993), we can use tabulated exact values for $l \leq 100$ and the Wilson-Hilferty approximation (which is quite accurate) for $l > 100$:

$$\chi^2(l) \approx l\left[1 - \frac{2}{9l} + \sqrt{\frac{2}{9l}}N(0,1)\right]^3.$$

4.3 The MIT Score

Throughout the previous discussion, we have omitted one very important detail: the decomposition of mutual information that we have used (Equation 14) is not unique and we can decompose $MI_D(X_i, Pa_G(X_i))$ in many other ways - as many as the number of possible orderings of the variables in $Pa_G(X_i)$, that is, $s_i!$. Each corresponds to a different way of including the variables in the parent set of X_i one at a time. The ordering does not affect the value $MI_D(X_i, Pa_G(X_i))$, but it can affect the penalization component (this will be the case whenever the number of states r_{ik} of all the variables is not the same). By way of example, let us assume that $Pa_G(X_i) = \{X_1, X_2, X_3\}$. The six possible decompositions of $MI_D(X_i, \{X_1, X_2, X_3\})$ are:

$$\begin{aligned}
&MI_D(X_i, X_1) + MI_D(X_i, X_2|X_1) + MI_D(X_i, X_3|\{X_1, X_2\})\\
&MI_D(X_i, X_1) + MI_D(X_i, X_3|X_1) + MI_D(X_i, X_2|\{X_1, X_3\})\\
&MI_D(X_i, X_2) + MI_D(X_i, X_1|X_2) + MI_D(X_i, X_3|\{X_1, X_2\})\\
&MI_D(X_i, X_2) + MI_D(X_i, X_3|X_2) + MI_D(X_i, X_1|\{X_2, X_3\})\\
&MI_D(X_i, X_3) + MI_D(X_i, X_1|X_3) + MI_D(X_i, X_2|\{X_1, X_3\})\\
&MI_D(X_i, X_3) + MI_D(X_i, X_2|X_3) + MI_D(X_i, X_1|\{X_2, X_3\}).
\end{aligned}$$

Let us suppose that the number of states of the variables X_i, X_1, X_2 and X_3 is $r_i = 3$, $r_1 = 2$, $r_2 = 3$ and $r_3 = 4$. The penalization component in Equation 17 for each of the six previous decompositions is therefore:

$$\begin{aligned}
\chi_{\alpha,2} + \chi_{\alpha,8} + \chi_{\alpha,36} &= 107.93\\
\chi_{\alpha,2} + \chi_{\alpha,12} + \chi_{\alpha,32} &= 109.21\\
\chi_{\alpha,4} + \chi_{\alpha,6} + \chi_{\alpha,36} &= 108.91\\
\chi_{\alpha,4} + \chi_{\alpha,18} + \chi_{\alpha,24} &= 111.96\\
\chi_{\alpha,6} + \chi_{\alpha,8} + \chi_{\alpha,32} &= 111.07\\
\chi_{\alpha,6} + \chi_{\alpha,16} + \chi_{\alpha,24} &= 112.89.
\end{aligned}$$

The numerical values in these expressions are computed for the parameter $\alpha = 0.999$. It should be noted that the total number $\sum_{j=1}^{s_i} l_{ij}$ of degrees of freedom is always the same, 46 in this case, which would correspond to the degrees of freedom of a marginal independence test between X_i and $Pa_G(X_i)$; such a test would use $(r_i - 1)(\prod_{j=1}^{s_i} r_{ij} - 1)$ degrees of freedom[11] (the value of $\chi_{\alpha,46}$

11. Observe that $\sum_{j=1}^{s_i} l_{ij} = \sum_{j=1}^{s_i}\left((r_i - 1)(r_{ij} - 1)\prod_{k=1}^{j-1} r_{ik}\right) = (r_i - 1)(\prod_{j=1}^{s_i} r_{ij} - 1)$.

in the example is 81.40). In any case, the values are different since the chi-square distribution is not additive with respect to the number of degrees of freedom.[12] Therefore, depending on the selected ordering, the score in Equation 17 will be different. This is undesirable since the same DAG (depending on the path that the search process follows to reach it) would be evaluated differently. In order to solve this problem, we believe that the best we can do is to use the most conservative option, that is, to use the greatest of all these values so as to evaluate each parent set in the worst possible way.

In order to formalize this idea, let $\sigma_i = (\sigma_i(1), \ldots, \sigma_i(s_i))$ denote any permutation of the index set $(1, \ldots, s_i)$ of the variables in $Pa_G(X_i) = \{X_{i1}, \ldots, X_{is_i}\}$, and let us define:

$$l_{i\sigma_i(j)} = \begin{cases} (r_i - 1)(r_{i\sigma_i(j)} - 1) \prod_{k=1}^{j-1} r_{i\sigma_i(k)} & j = 2 \ldots, s_i \\ (r_i - 1)(r_{i\sigma_i(1)} - 1) & j = 1\,. \end{cases} \tag{19}$$

Then, instead of using Equation 17, the global quality measure of the set $Pa_G(X_i)$ that we propose is:

$$g(X_i, Pa_G(X_i) : D) = 2N\,MI_D(X_i, Pa_G(X_i)) - \max_{\sigma_i} \sum_{j=1}^{s_i} \chi_{\alpha, l_{i\sigma_i(j)}}.$$

The final expression of the proposed scoring function, which we shall call MIT (from mutual information tests), is:

$$g_{MIT}(G : D) = \sum_{\substack{i=1 \\ Pa_G(X_i) \neq \emptyset}}^{n} \left(2N\,MI_D(X_i, Pa_G(X_i)) - \max_{\sigma_i} \sum_{j=1}^{s_i} \chi_{\alpha, l_{i\sigma_i(j)}} \right). \tag{20}$$

Computing each penalization component $\max_{\sigma_i} \sum_{j=1}^{s_i} \chi_{\alpha, l_{i\sigma_i(j)}}$ in the previous expression might seem to be a very time-consuming task since it would be necessary to evaluate all the $s_i!$ possible permutations of the variables in the set $Pa_G(X_i)$ in order to calculate the maximum. Fortunately, this will not be necessary as this maximum can be obtained in a much simpler way:

Theorem 2 *For the values $l_{i\sigma_i(j)}$ defined in Equation 19,*

$$\max_{\sigma_i} \sum_{j=1}^{s_i} \chi_{\alpha, l_{i\sigma_i(j)}} = \sum_{j=1}^{s_i} \chi_{\alpha, l_{i\sigma_i^*(j)}},$$

where σ_i^* is any permutation of $Pa_G(X_i)$ satisfying $r_{i\sigma_i^*(1)} \geq r_{i\sigma_i^*(2)} \geq \ldots \geq r_{i\sigma_i^*(s_i)}$, whenever the function $f_{i,\alpha} : \mathcal{N}^{s_i} \longrightarrow \mathcal{R}$, defined as $f_{i,\alpha}(l_1, \ldots, l_{s_i}) = \sum_{j=1}^{s_i} \chi_{\alpha, l_j}$, is a Shur-concave function.

This result says that the permutation that produces the maximum penalization value is the one where the first variable has the greatest number of states, the second variable has the second largest number of states, and so on. In the previously considered example, this permutation is $\{X_3, X_2, X_1\}$, and this reaches a maximum value equal to 112.89.

Conjecture 3 *The function $f_{i,\alpha}$ defined in Theorem 2 is Shur-concave, whenever $\alpha \geq 0.59$.*

12. With the exception of a sum of *independent* chi-square distributions, which obviously is not the case.

The combination of theoretical and empirical arguments that support this conjecture is included in the Appendix. The restriction concerning α does not represent any practical problem since we shall always use values of α which are much greater than 0.59.

Another way of measuring the quality of a set of variables $\mathbf{Z}$ as the parent set of X_i, which as it turns out is equivalent to the previous one, is as follows: we can consider that $\mathbf{Z}$ will be a good parent set if it continues to be a good parent set when one of its variables is removed, $\mathbf{Z} \setminus \{Y\}$, and also the variable Y that we have removed should not have been removed, that is, Y is not independent of X_i given $\mathbf{Z} \setminus \{Y\}$. As we can do this for each variable in $\mathbf{Z}$, the final value should be the smallest one (we are again using a conservative or pessimistic view). This leads to a recursive definition of $g(X_i, Pa_G(X_i) : D)$. The way of measuring the degree of undesirability of removing the variable Y from $\mathbf{Z}$ is to use the difference between the mutual information statistic $2N\,MI_D(X_i, Y|\mathbf{Z} \setminus \{Y\})$ and the chi-square value $\chi_{\alpha,l}$ with the appropriate degrees of freedom. In this way, if Y is truly independent on X_i given $\mathbf{Z} \setminus \{Y\}$, then this difference will be negative and in this case we would prefer to use $\mathbf{Z} \setminus \{Y\}$ instead of $\mathbf{Z}$ as the parent set of X_i. If, on the contrary, the difference is positive, the set $\mathbf{Z}$ will be preferable to $\mathbf{Z} \setminus \{Y\}$.

We can therefore recursively define the score $g_r(X_i, Pa_G(X_i) : D)$ in the following way:

$$g_r(X_i, Pa_G(X_i) : D) = \min_{X_{ij} \in Pa_G(X_i)} \Big\{ g_r(X_i, Pa_G(X_i) \setminus \{X_{ij}\} : D) + 2N\,MI_D(X_i, X_{ij} | Pa_G(X_i) \setminus \{X_{ij}\}) - \chi_{\alpha, l^r_{ij}} \Big\}, \quad (21)$$

where $\chi_{\alpha,l^r_{ij}}$ is the value such that $p(\chi^2(l^r_{ij}) \leq \chi_{\alpha,l^r_{ij}}) = \alpha$ and the number of degrees of freedom is $l^r_{ij} = (r_i - 1)(r_{ij} - 1)\prod_{\substack{k=1 \\ k \neq j}}^{s_i} r_{ik}$. The starting point of this recursive definition is obviously $g_r(X_i, \emptyset : D) = 0$. We can prove the following result:

Theorem 4 *The MIT scoring function defined in Equation 20 can also be expressed as:*

$$g_{MIT}(G : D) = \sum_{\substack{i=1 \\ Pa_G(X_i) \neq \emptyset}}^{n} g_r(X_i, Pa_G(X_i) : D),$$

where $g_r(X_i, Pa_G(X_i) : D)$ are the local scores defined in Equation 21.

Let us study some of the properties of the MIT score.

Theorem 5 *The MIT scoring function defined in Equation 20 is decomposable.*

Unfortunately, MIT is not score-equivalent. Let us consider the following example: for the two DAGs G_1 and G_2 in Figure 1 and which are equivalent, let us suppose that the number of states of each variable is: $r_1 = 5$, $r_2 = 4$, $r_3 = 3$, $r_4 = 2$. Therefore:

$$\begin{aligned} g(G_1 : D) &= 2N(MI_D(X_1, \{X_2, X_3\}) + MI_D(X_2, X_3) + MI_D(X_3, X_4)) \\ &\quad -(\chi_{\alpha,12} + \chi_{\alpha,32} + \chi_{\alpha,6} + \chi_{\alpha,2}) \\ g(G_2 : D) &= 2N(MI_D(X_2, \{X_1, X_3\}) + MI_D(X_3, X_1) + MI_D(X_4, X_3)) \\ &\quad -(\chi_{\alpha,12} + \chi_{\alpha,30} + \chi_{\alpha,8} + \chi_{\alpha,2}). \end{aligned}$$

Although it seems that the part corresponding to mutual information is different in both cases, it is in fact not. It is sufficient to take into account Equation 12 and remember that the maximum

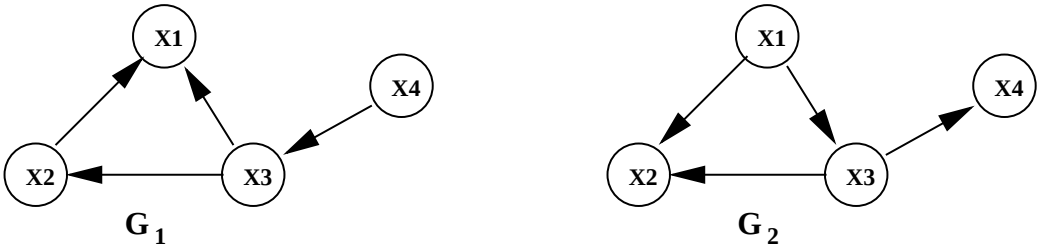


Figure 1: Two equivalent DAGs with different values of the MIT score

likelihood score is score-equivalent. The problem appears with the penalization by means of the sum of chi-square values: if the variables have a different number of states (as in this case), the results are different. More specifically, the penalization component is 131.67 for G_1 but 132.55 for G_2 (assuming that $\alpha = 0.999$).

The MIT score, however, satisfies a less demanding property than score-equivalence, and this concerns another type of space of equivalent DAGs, namely RPDAGs (Acid and de Campos, 2003). They are PDAGs which represent sets of equivalent DAGs, although they are not a canonical representation of equivalence classes of DAGs (two different RPDAGs may correspond to the same equivalence class). Let us introduce some additional notation and then the concept of RPDAG. The *skeleton* of a DAG is the undirected graph that results from ignoring the directionality of every arc. A *h-h pattern* (*head-to-head pattern*) in a DAG G is an ordered triplet of nodes, (X_i, X_k, X_j), such that G contains the arcs $X_i \rightarrow X_k$ and $X_j \rightarrow X_k$. Given a PDAG $G = (\mathbf{U_n}, E_G)$, for each node $X_i \in \mathbf{U_n}$, $Sib_G(X_i) = \{X_j \in \mathbf{U_n} \mid X_i — X_j \in E_G\}$ is the set of *siblings* or *neighbors* of X_i. A PDAG G is an RPDAG if and only if it satisfies the following conditions:

1. $\forall X_i \in \mathbf{U_n}$, if $Pa_G(X_i) \neq \emptyset$ then $Sib_G(X_i) = \emptyset$.
2. G contains neither directed nor completely undirected cycles.
3. $\forall X_i, X_j \in \mathbf{U_n}$, if $X_j \in Pa_G(X_i)$ then either $|Pa_G(X_i)| \geq 2$ or $Pa_G(X_j) \neq \emptyset$.

The difference between essential graphs and RPDAGs appears when there are triangular structures: essential graphs may have completely undirected cycles, but these cycles must be *chordal* (Andersson et al., 1997). In other words, undirected cycles are forbidden in RPDAGs, whereas in essential graphs only undirected non-chordal cycles are forbidden. It can be seen that all the DAGs which are represented by a given RPDAG are equivalent and have the same skeleton and the same h-h patterns, whereas the DAGs associated with an essential graph have the same skeleton and the same v-structures (h-h patterns where the extreme nodes are not adjacent) (Pearl and Verma, 1990). Therefore, the role played by the v-structures in essential graphs is the same as that played by the h-h patterns in RPDAGs. The objective of RPDAGs is to trade the uniqueness of the representation of equivalence classes of DAGs for a more manageable one, because testing whether a given PDAG G is an RPDAG is easier than testing whether G is an essential graph.

Theorem 6 *The MIT scoring function assigns the same value to all DAGs that are represented by the same RPDAG.*

Although the MIT score should not be used to search in the space of essential graphs, we can therefore use it without any problem to search in both the DAG and the RPDAG space.

To conclude our study of the new score, we have observed an interesting relation between MIT and the scoring functions based on Equation 6. First, it should be noted that the log-likelihood of the simplest possible network, namely the empty network $G_\emptyset$, is, according to Equation 4 (and taking into account that in this case $q_i = 1$ and $N_{ijk} = N_{ik}$):

$$LL_D(G_\emptyset) = \sum_{i=1}^{n} \sum_{k=1}^{r_i} N_{ik} \log\left(\frac{N_{ik}}{N}\right) = -N \sum_{i=1}^{n} H_D(X_i).$$

Then, considering Equation 12, we can express the sum of mutual information measures between each variable and its set of parents in G as follows:

$$\sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} MI_D(X_i, Pa_G(X_i)) = \frac{LL_D(G) - LL(G_\emptyset)}{N}.$$

Therefore, the sum of mutual information measures coincides with the difference between the log-likelihood of G and the one of $G_\emptyset$ or, equivalently, with the difference between the description length of the data given $G_\emptyset$ and given G. Now, let us consider the difference between G and $G_\emptyset$ in terms of complexity, which is:

$$C(G) - C(G_\emptyset) = \sum_{i=1}^{n} (r_i - 1) q_i - \sum_{i=1}^{n} (r_i - 1) = \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} (r_i - 1)(q_i - 1) = \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} \sum_{j=1}^{s_i} l_{ij},$$

with l_{ij} defined as in Equation 16. Therefore, for the information-based scoring function defined in Equation 6, using $f(N) = 1/2$, the difference between the scores of G and $G_\emptyset$ is:

$$\begin{aligned} g(G:D) - g(G_\emptyset:D) &= \Big(LL(G) - C(G) f(N)\Big) - \Big(LL(G_\emptyset) - C(G_\emptyset) f(N)\Big) \\ &= N \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} MI_D(X_i, Pa_G(X_i)) - \frac{1}{2} \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} \sum_{j=1}^{s_i} l_{ij} \\ &= \frac{1}{2} \sum_{\substack{i=1 \\ Pa_G(X_i)\neq\emptyset}}^{n} \Big(2N\, MI_D(X_i, Pa_G(X_i)) - \sum_{j=1}^{s_i} l_{ij}\Big). \end{aligned} \tag{22}$$

The similarity of this expression with those in Equations 18 and 20 is apparent. Therefore, the MIT score of a network G could be interpreted in terms of the difference between the information-based scores of G and $G_\emptyset$, and also as the decrease in description length achieved by using G instead of $G_\emptyset$. By considering that the mean value of a χ^2 distribution with l degrees of freedom is just l, we can see that the MIT score appears when we replace in Equation 22 the mean values of the $\chi^2(l_{ij})$ distributions by the corresponding α-quantiles.

5. Experimental Evaluation

In order to determine the possible merit of the proposed scoring function in practical terms, in this section we shall carry out an experimental evaluation of the MIT score, comparing it with other well-known scoring functions. The selected scoring functions are the most frequently used: K2 (Equation 1), BDeu (Equation 3) and BIC/MDL (Equation 5). For BDeu, we shall use a uniform

prior distribution over possible structures and as this score is quite sensitive with respect to the value of the equivalent sample size, we shall use five values of this parameter, more precisely $\eta = 1, 2, 4, 8, 16$. For the single parameter of the MIT score (i.e., the confidence level), we shall use three values: $\alpha = 0.99, 0.999, 0.9999$.

The software necessary to carry out the experiments has been developed on the *Elvira* system (Elvira, 2002), a Java tool for building and using Bayesian networks and influence diagrams.

First, we define the performance criteria that we shall use to compare the different scoring functions.

5.1 Performance Criteria

One way of measuring the quality of a scoring function is to study its ability to *reconstruct* (in combination with a learning algorithm based on score+search) the Bayesian network which generated the data. In other words, we begin with a Bayesian network G_0 which is completely specified in terms of structure and parameters, and we obtain a data set of a given size by sampling from G_0. Then, using the scoring function together with a search method, we obtain a learned network G, which must be compared with the original network G_0. This capacity for reconstruction can be understood in two different but complementary ways: reconstructing the graphical structure and reconstructing the associated joint probability distribution. In terms of the first of these, the usual evaluation consists in measuring the structural differences between the original and the learned networks. More precisely, the number of added arcs ($A(G)$), deleted arcs ($D(G)$), and inverted arcs ($I(G)$) in the learned network with respect to the original one is computed. In order to eliminate fictitious differences or similarities between the two networks regarding the number of inverted arcs (caused by different but equivalent subDAG structures), before the two networks are compared they will be converted into their corresponding essential graph representation using the algorithm proposed by Chickering (1995). If G' and G'_0 represent the essential graphs associated with G and G_0, respectively, then the three measures of structural difference can be calculated using the following expressions:

$$A(G) = \frac{1}{2}\sum_{i=1}^{n} |Ad_{G'}(X_i) \setminus Ad_{G'_0}(X_i)|$$

$$D(G) = \frac{1}{2}\sum_{i=1}^{n} |Ad_{G'_0}(X_i) \setminus Ad_{G'}(X_i)|$$

$$I(G) = \sum_{i=1}^{n} \Big(|Pa_{G'_0}(X_i) \cap Sib_{G'}(X_i)| + |Pa_{G'}(X_i) \cap Sib_{G'_0}(X_i)| + |Pa_{G'_0}(X_i) \cap Ch_{G'}(X_i)| \Big).$$

where $Ch_H(X_i) = \{X_j \in \mathbf{U_n} \mid X_i \rightarrow X_j \in E_H\}$ and $Ad_H(X_i) = Pa_H(X_i) \cup Ch_H(X_i) \cup Sib_H(X_i)$ are the sets of children and adjacent nodes of X_i in a PDAG H. As a way of summarizing these three measures, the *Hamming distance*, which is simply the sum of all the structural differences, $H(G) = A(G) + D(G) + I(G)$, is also usually considered.

In terms of the ability to reconstruct the joint probability distribution, we can evaluate this by means of a distance measure between the distributions associated with the original and the learned networks, p_{G_0} and p_G, respectively. We shall use the Kullback-Leibler divergence:

$$KL(G) = KL(p_{G_0}, p_G) = \sum_{x_1,\ldots,x_n} p_{G_0}(x_1,\ldots,x_n) \log\left(\frac{p_{G_0}(x_1,\ldots,x_n)}{p_G(x_1,\ldots,x_n)}\right).$$

The conditional probability distributions that constitute the factorization of p_G will be calculated from the data set using the Laplace estimation (Good, 1965), which avoids the problem of obtaining an infinite value of the Kullback-Leibler divergence, caused by zero probability values in p_G.

The calculus of this distance measure for joint distributions with many variables is computationally very expensive. However, by taking advantage of the factorization of the distributions, the complexity may be considerably reduced and the value $KL(G)$ can be expressed as follows:

$$\begin{aligned} KL(G) &= \sum_{i=1}^{n}\sum_{k=1}^{r_i}\sum_{j=1}^{q_i^{G_0}} p_{G_0}(x_{ik}, w_{ij}^{G_0}) \log(p_{G_0}(x_{ik}|w_{ij}^{G_0})) \\ &\quad - \sum_{i=1}^{n}\sum_{k=1}^{r_i}\sum_{j=1}^{q_i^{G}} p_{G_0}(x_{ik}, w_{ij}^{G}) \log(p_{G}(x_{ik}|w_{ij}^{G})), \end{aligned}$$

where $w_{ij}^{G_0}$ and w_{ij}^{G} represent the j-th configuration of the parent sets of X_i in G_0 and G, respectively (each having a total number of possible configurations equal to $q_i^{G_0}$ and q_i^{G}, respectively). In this way, the only probability values that must be computed are $p_{G_0}(x_{ik}, w_{ij}^{G_0})$ and $p_{G_0}(x_{ik}, w_{ij}^{G})$, and this can be done relatively efficiently by using a propagation algorithm in the network G_0. We have used an exact algorithm based on variable elimination.

One alternative way of measuring the quality of a scoring function which does not require an initial Bayesian network to be used as a starting point is to use the network learned with such a scoring function for a specific task and then to evaluate the level of success achieved. As Bayesian networks have been used in different ways to build classifiers, we can evaluate the quality of a scoring function (at least in comparative terms) by building a classifier using an algorithm for learning Bayesian networks which is specific for classification and equipped with the scoring function, and then measuring its classification capacity.

5.2 Experiments for Reconstructing Bayesian Networks

In order to make our comparative study more representative, we shall use different problems or rather different original networks. We shall also use different database sizes. Although this parameter clearly affects the quality of the networks learned with any scoring function (greater sizes lead to better estimations), we want to check which of the scoring functions may be more or less sensitive in the sense that their behavior deteriorates more quickly when smaller sample sizes are used.

In the following sections, we shall first give details of the experimental design before presenting the obtained results.

5.2.1 EXPERIMENTAL DESIGN

We have selected four Bayesian networks corresponding to different problems: Alarm (Figure 2), Boblo (Figure 3), Insurance (Figure 4) and Hailfinder (Figure 5).

The Alarm network displays the relevant variables and relationships for the Alarm Monitoring System (Beinlich et al., 1989), a diagnostic application for patient monitoring. This network contains 37 variables and 46 arcs. Boblo (Rasmussen, 1995) is part of a system for determining the blood group of Jersey cattle. The Boblo network contains 23 variables and 24 arcs. Hailfinder (Abramson et al., 1996) is a normative system that forecasts severe summer hail in northeastern Colorado. The Hailfinder network contains 56 variables and 66 arcs. Insurance (Binder et al., 1997)

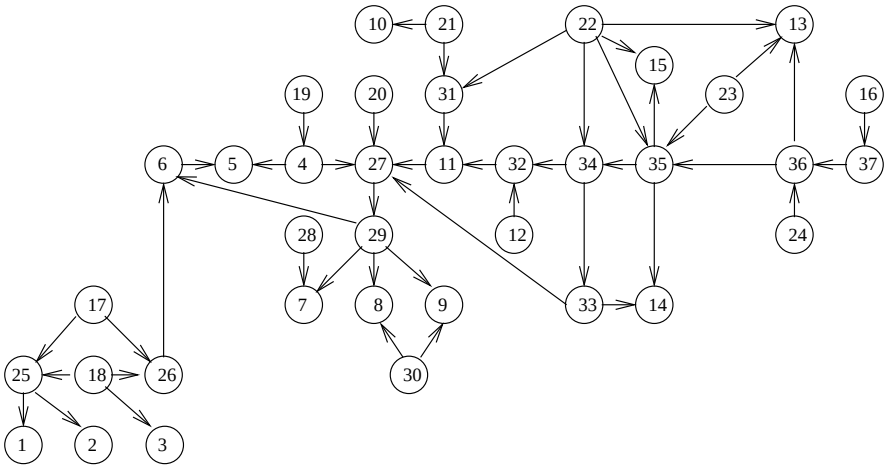


Figure 2: The Alarm network

is a network for evaluating car insurance risks. The Insurance network contains 27 variables and 52 arcs. All these networks have been widely used in specialist literature for comparative purposes.

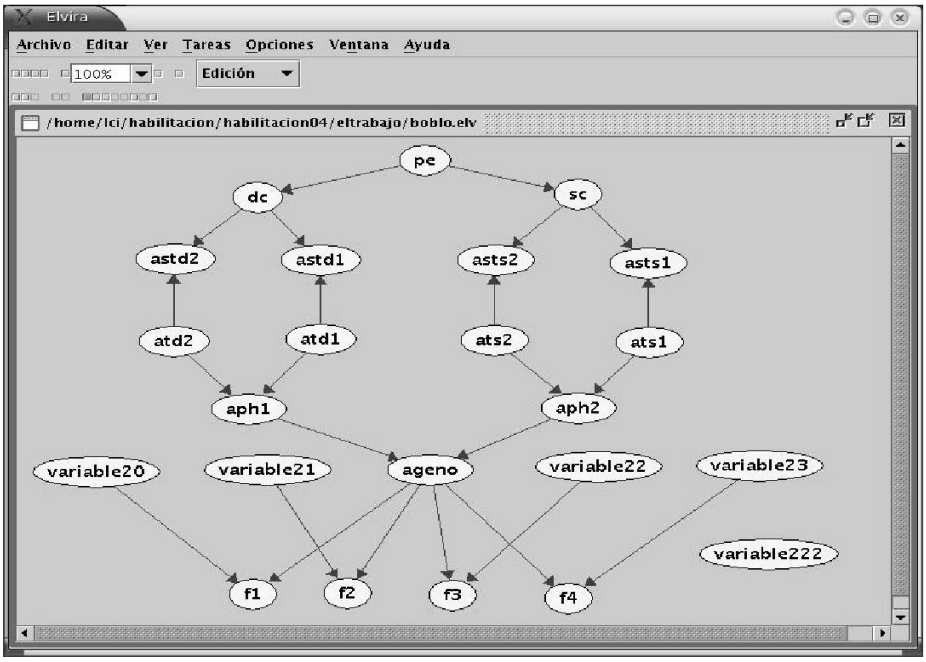


Figure 3: The Boblo network

Each network has been used to generate several databases, each of which contains 10000 instances; more precisely, we have generated five data sets for each problem. The results that we will show are the averages across the five data sets. The sample sizes considered are $N = 10000$, 5000 and 1000 (using the complete data sets and the first 5000 and 1000 instances of each one, respectively).

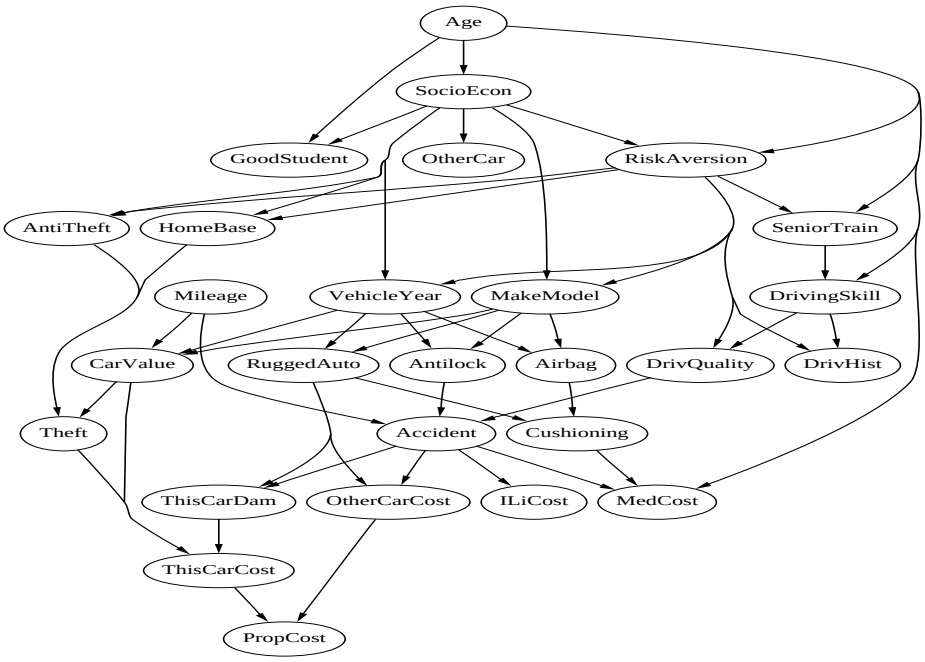


Figure 4: The Insurance network

The search method that we shall use is a local search in the DAG space with the classical operators of arc addition, arc deletion and arc reversal. The starting point of the search is always the empty graph. Although our main objective is to compare the proposed score with others, given that MIT has some similarities with constraint-based methods, it is also interesting to include one of these methods in the comparison. We have selected the well-known PC algorithm (Spirtes et al., 1993). This algorithm also depends on one parameter α representing the confidence level of the independence tests. We shall use three values: $\alpha = 0.90, 0.95, 0.99$.

We therefore have a design $13 \times 4 \times 3$ (10 scoring functions plus 3 versions of a constraint-based algorithm, 4 problems and 3 sample sizes), and for each of these 156 configurations we use 5 different databases, which gives us a total of 780 experiments.

5.2.2 RECONSTRUCTION RESULTS

Tables 1, 2, 3 and 4 display the results obtained for the Alarm, Boblo, Hailfinder and Insurance networks, respectively. For each sample size and each method, each table shows the average values of the previously mentioned performance measures (A, D, I, H and KL). The best value for each performance measure is written in bold and the second best in italics. In the last two rows of each table, we also show the KL values for the original network (with parameters re-trained from the corresponding database) and the empty network, which may serve as a kind of scale. Table 5 displays an illustrative summary of the results: it shows the number of times (from the 12 configurations being considered for each method) that each method has obtained the best result (and either the best or the second best result) for each of the five performance measures.

The first thing that can be observed is that these results seem to confirm our intuition about the need to use MIT with a greater confidence level α than those typically used for independence tests,

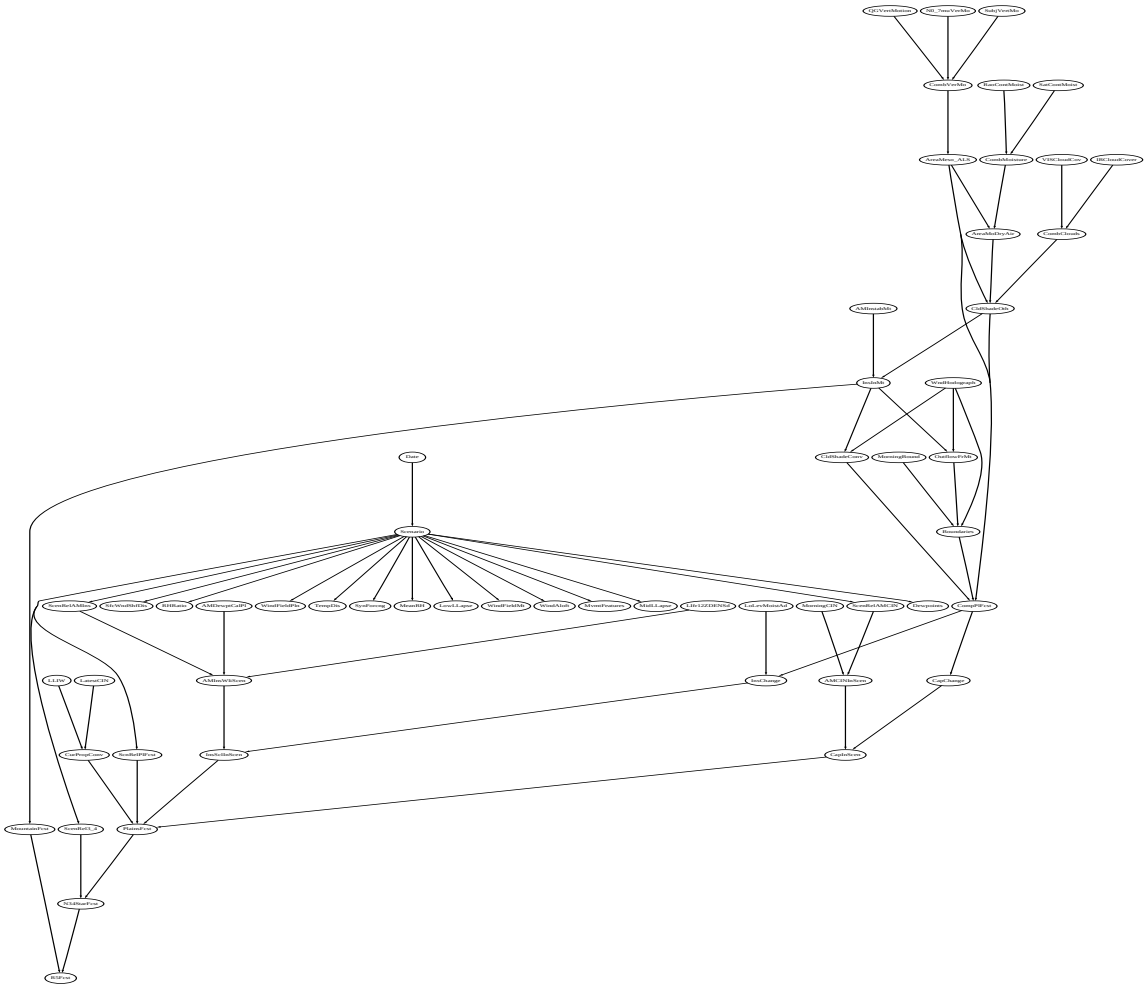


Figure 5: The Hailfinder network

since MIT with the values $\alpha = 0.999, 0.9999$ offers better results than with $\alpha = 0.99$. It is also possible to observe how MIT generally behaves better than the other scores, with respect to all the performance measures, and more specifically, in terms of BIC/MDL (which is the closest scoring function in spirit to the new score), MIT systematically obtains much better results. Although BIC behaves acceptably in terms of the number of added arcs, it does however have a marked propensity to remove a large number of arcs. This suggests that the penalization component used by BIC is not well calibrated. On the other hand, the different versions of BDeu behave rather poorly (except in terms of the number of deleted arcs). K2 only offers good results for the KL divergence. The PC algorithm behaves very good for the number of added and inverted arcs. However, its results in terms of the number of deleted arcs and KL divergence are extremely poor.

Focusing on the two main performance measures (the Hamming distance and the KL divergence), for each pair of methods, Tables 6 and 7 contain the number of times that each method obtains better results than the other. Table 6 refers to the KL divergence and Table 7 to the Hamming distance. In both cases, the MIT versions using high confidence levels (0.9999 and 0.999)

ALARM															
N	1000					5000					10000				
Score	A	D	I	H	KL	A	D	I	H	KL	A	D	I	H	KL
M9999	4.2	4.6	9.6	*18.4*	0.32752	4.6	*2.4*	**4.6**	**11.6**	**0.06384**	7.6	*2.6*	9.2	19.4	**0.04372**
M999	4.2	4.0	9.4	**17.6**	0.31571	4.2	3.0	**4.6**	*11.8*	*0.06448*	9.8	*2.6*	10.0	22.4	0.04563
M99	7.8	4.0	9.4	21.2	*0.31270*	8.4	**2.0**	*4.8*	15.2	0.06925	12.6	**2.4**	10.0	25.0	0.04743
BIC	*7.2*	7.4	20.0	34.6	0.49799	7.4	4.6	14.0	26.0	0.18683	9.6	3.4	18.2	31.2	0.09983
K2	10.0	4.2	16.0	30.2	**0.27079**	8.4	3.2	14.2	25.8	0.07222	8.8	3.0	14.6	26.4	*0.04375*
BD1	11.0	4.0	17.4	32.4	0.32570	9.6	3.2	13.4	26.2	0.08782	8.2	3.0	14.2	25.4	0.04855
BD2	14.6	4.2	20.6	39.4	0.33198	11.0	2.8	15.0	28.8	0.09294	7.4	*2.6*	16.0	26.0	0.04387
BD4	18.0	**3.4**	15.4	36.8	0.32044	11.6	*2.4*	17.6	31.6	0.06652	14.0	3.2	19.4	36.6	0.04797
BD8	27.8	3.8	17.8	49.4	0.34363	16.8	2.6	16.0	35.4	0.07469	13.4	**2.4**	15.0	30.8	0.04491
BD16	48.8	*3.6*	19.4	71.8	0.42465	31.8	3.0	15.2	50.0	0.09508	24.4	2.8	14.2	41.4	0.04582
PC90	2.8	17.0	**8.4**	28.2	2.63819	0.6	9.0	5.4	15.0	1.21272	*0.4*	8.0	**4.6**	**13.0**	1.06377
PC95	*2.2*	17.6	**8.4**	28.2	2.69645	*0.4*	9.2	5.4	15.0	1.29207	**0.2**	7.6	*5.8*	*13.6*	0.95810
PC99	**1.8**	18.8	*8.8*	29.4	2.82810	**0.2**	10.6	6.0	16.8	1.63841	*0.4*	7.8	6.2	14.4	1.00228
true					0.21351					0.04759					0.02421
empty					10.2445					10.0677					10.0631

Table 1: Results for the Alarm network

BOBLO															
N	1000					5000					10000				
Score	A	D	I	H	KL	A	D	I	H	KL	A	D	I	H	KL
M9999	*0.4*	5.0	*0.8*	*6.2*	0.15105	**0.0**	2.2	**0.0**	2.2	0.03359	0.8	*0.2*	**1.6**	**2.6**	*0.01396*
M999	*0.4*	4.4	**0.4**	**5.2**	*0.14458*	*0.2*	1.8	**0.0**	*2.0*	0.03266	0.8	*0.2*	**1.6**	**2.6**	*0.01396*
M99	1.0	4.0	1.2	*6.2*	0.14812	*0.2*	*1.6*	**0.0**	**1.8**	**0.03208**	1.2	**0.0**	**1.6**	*2.8*	**0.01353**
BIC	2.0	6.4	4.6	13.0	0.16222	3.0	3.8	*4.6*	11.4	0.03651	2.8	2.4	*3.0*	8.2	0.01993
K2	10.6	4.0	8.8	23.4	**0.13805**	11.0	2.6	7.6	21.2	0.03563	7.8	1.2	6.8	15.8	0.01748
BD1	28.6	*3.2*	2.8	34.6	0.15329	13.4	*1.6*	*4.6*	19.6	*0.03211*	7.2	2.0	4.4	13.6	0.01481
BD2	30.8	**2.6**	4.0	37.4	0.15452	21.2	2.2	7.2	30.6	0.03928	16.8	1.6	7.4	25.8	0.01705
BD4	37.4	**2.6**	2.8	42.8	0.16213	28.0	1.8	4.8	34.6	0.03983	26.2	1.4	6.4	34.0	0.02065
BD8	50.8	3.6	3.4	57.8	0.17616	41.2	**1.4**	5.2	47.8	0.04539	38.2	1.0	9.2	48.4	0.02317
BD16	64.2	**2.6**	6.6	73.4	0.18015	54.0	2.0	6.0	62.0	0.05415	49.6	1.4	3.2	54.2	0.02830
PC90	**0.0**	13.0	5.4	18.4	2.02929	0.8	10.0	6.2	17.0	1.44017	1.4	10.2	6.2	17.8	1.43512
PC95	**0.0**	14.4	5.0	19.4	2.22612	*0.2*	10.0	6.0	16.2	1.43634	*0.2*	9.6	6.4	16.2	1.42543
PC99	**0.0**	15.0	4.6	19.6	2.33032	**0.0**	10.8	5.6	16.4	1.50436	**0.0**	9.8	6.2	16.0	1.42574
true					0.13107					0.02712					0.01355
empty					7.44795					7.42898					7.42653

Table 2: Results for the Boblo network

compare favorably with the other scores. They systematically produce networks with much fewer structural differences with respect to the original networks and, at the same time, they almost always estimate the true joint probability distributions more closely. In terms of the Hamming distance, BIC is somewhat better than K2 and much better than BDeu, which systematically obtains worse results as the equivalent sample size increases. However, regarding the Kullback-Leibler divergence, K2 is much better than BIC and most of the versions of BDeu. The constraint-based algorithm is not able to find a good approximation of the joint probability distribution, probably because of the high number of deleted arcs together with the low number of added arcs.[13] In terms of the Hamming distance, PC performs better than all the Bayesian scores, although MIT and, to a lesser extent, BIC, outperform it.

13. Extra arcs could be useful to compensate for the missing arcs.

HAILFINDER															
N	1000					5000					10000				
Score	A	D	I	H	KL	A	D	I	H	KL	A	D	I	H	KL
M9999	*7.2*	12.2	*8.2*	**27.6**	**1.08438**	**8.0**	*5.8*	**4.2**	**18.0**	**0.26576**	**6.2**	*5.6*	**1.2**	**13.0**	**0.14678**
M999	8.6	*11.0*	8.6	*28.2*	1.13183	9.6	**5.6**	*4.6*	*19.8*	0.29131	*7.6*	**5.4**	*1.6*	*14.6*	0.16634
M99	19.6	**10.0**	**6.8**	36.4	1.45014	21.2	*5.8*	8.8	35.8	0.47866	18.2	5.8	9.8	33.8	0.28220
BIC	**6.4**	16.2	15.0	37.6	1.36774	9.6	13.8	14.4	37.8	0.38606	10.0	10.2	17.2	37.4	0.21192
K2	10.4	13.2	18.2	41.8	*1.09179*	*9.0*	8.6	22.0	39.6	*0.27891*	10.2	7.6	22.2	40.0	*0.15910*
BD1	16.0	18.4	16.2	50.6	1.43422	17.0	13.0	21.4	51.4	0.40585	19.2	10.8	26.4	56.4	0.23520
BD2	16.2	17.0	20.4	53.6	1.35804	19.2	12.6	20.6	52.4	0.35806	16.2	9.8	18.8	44.8	0.19763
BD4	16.6	17.2	13.8	47.6	1.30878	18.4	13.2	18.0	49.6	0.36146	19.0	8.8	17.0	44.8	0.18702
BD8	15.8	15.8	16.8	48.4	1.25347	20.2	12.0	20.4	52.6	0.33352	21.4	9.2	25.6	56.2	0.18622
BD16	23.0	15.0	15.2	53.2	1.30559	22.8	10.4	15.0	48.2	0.33260	23.0	8.2	15.2	46.4	0.19391
PC90	10.2	36.6	8.8	55.6	9.19075	14.8	33.4	7.0	55.2	8.38057	16.6	33.2	8.4	58.2	8.25173
PC95	10.2	36.6	9.0	55.8	9.19961	13.8	33.2	6.8	53.8	8.38573	15.6	32.8	8.0	56.4	8.23382
PC99	11.6	36.8	9.4	57.8	9.15348	13.8	33.4	6.6	53.8	8.32864	14.8	32.4	7.2	54.4	8.21041
true					1.18225					0.28146					0.14798
empty					20.6712					20.6048					20.5969

Table 3: Results for the Hailfinder network

INSURANCE															
N	1000					5000					10000				
Score	A	D	I	H	KL	A	D	I	H	KL	A	D	I	H	KL
M9999	3.4	14.8	13.4	31.6	0.50383	4.8	10.2	12.8	27.8	0.14468	3.8	7.2	6.4	**17.4**	**0.06440**
M999	3.6	*14.0*	13.0	*30.6*	0.50499	5.0	*9.4*	12.2	**26.6**	**0.14226**	4.2	*6.6*	9.0	*19.8*	*0.06653*
M99	3.8	**12.2**	13.4	**29.4**	**0.45608**	6.8	**8.8**	11.8	*27.4*	0.14513	4.6	**6.4**	14.0	25.0	0.06952
BIC	4.0	23.0	12.0	39.0	0.97628	*4.4*	14.8	15.8	35.0	0.25910	5.2	11.0	12.4	28.6	0.13403
K2	9.2	17.0	19.4	45.6	0.52187	10.6	12.8	23.2	46.6	0.16905	10.4	11.8	21.4	43.6	0.10118
BD1	6.2	17.2	13.8	37.2	0.57087	6.2	12.0	14.8	33.0	0.18197	7.2	10.6	19.0	36.8	0.12997
BD2	5.6	14.8	14.2	34.6	*0.48989*	7.2	12.6	21.0	40.8	0.16623	8.8	11.0	18.6	38.4	0.13644
BD4	9.4	15.0	19.0	43.4	0.50435	8.6	10.8	14.4	33.8	0.15113	6.0	8.4	16.4	30.8	0.08331
BD8	16.2	16.4	17.8	50.4	0.53299	14.6	11.6	21.6	47.8	0.15281	10.2	9.2	13.2	32.6	0.09064
BD16	22.2	14.6	19.6	56.4	0.58103	20.4	10.0	24.4	54.8	*0.14247*	18.8	7.6	19.8	46.2	0.08384
PC90	2.0	30.6	**8.8**	41.4	2.31070	**0.2**	22.2	**8.4**	30.8	0.96871	*0.2*	19.4	**4.8**	24.4	0.58962
PC95	*1.8*	30.6	*9.0*	41.4	2.31837	**0.2**	22.4	*9.6*	32.2	1.03911	*0.2*	19.6	*5.0*	24.8	0.57544
PC99	**1.4**	31.2	**8.8**	41.4	2.42852	**0.2**	23.2	10.8	34.2	1.05543	**0.0**	20.0	5.4	25.4	0.62231
true					0.55527					0.12023					0.06205
empty					8.46596					8.44041					8.43720

Table 4: Results for the Insurance network

We believe that these results support the conclusion that the MIT score can compete favorably with state-of-the-art scoring functions and constraint-based algorithms for the task of learning general purpose Bayesian networks. Moreover, in the case that we wish to select a non-Bayesian scoring function based on information theory, we would recommend BIC/MDL be discarded and MIT used instead.

It is also interesting to remark that the two scoring functions that behave best (MIT and K2) are not score equivalent, whereas the two that obtain comparatively poor results (BIC and BDeu), are. Therefore, score equivalence does not seem to be an important property for learning Bayesian networks by searching in the DAG space. This confirms the previous results stated by Yang and Chang (2002).

While it is clear from the previous experiments that the new score, in combination with the particular search procedure being used, has an excellent performance, we would also like to test whether the different scores differentiate structures that are more accurate or generalize better, inde-

	times best/times best or second best				
Score	A	D	I	H	KL
M9999	*3* / *5*	0 / 5	**5** / **7**	**6** / *8*	**6** / **7**
M999	0 / 3	*2* / *8*	*4* / *6*	*4* / **11**	1 / 5
M99	0 / 1	**7** / **9**	3 / 4	2 / 5	*3* / 4
BIC	1 / 2	0 / 0	0 / 2	0 / 0	0 / 0
K2	0 / 1	0 / 0	0 / 0	0 / 0	2 / *6*
BD1	0 / 0	0 / 2	0 / 1	0 / 0	0 / 1
BD2	0 / 0	1 / 2	0 / 0	0 / 0	0 / 1
BD4	0 / 0	*2* / 3	0 / 0	0 / 0	0 / 0
BD8	0 / 0	*2* / 2	0 / 0	0 / 0	0 / 0
BD16	0 / 0	1 / 2	0 / 0	0 / 0	0 / 1
PC90	2 / 4	0 / 0	**5** / 5	1 / 1	0 / 0
PC95	*3* / **9**	0 / 0	1 / 5	0 / 1	0 / 0
PC99	**8** / **9**	0 / 0	1 / 2	0 / 0	0 / 0

Table 5: Number of times that each method obtained the best/the best or second best result in terms of each performance measure

	Kullback-Leibler												
	M9999	M999	M99	K2	BIC	BD1	BD2	BD4	BD8	BD16	PC90	PC95	PC99
M9999	–	7	7	10	12	10	11	11	12	11	12	12	12
M999	4	–	8	6	12	11	10	11	11	12	12	12	12
M99	5	4	–	6	9	9	8	8	8	7	12	12	12
K2	2	6	6	–	12	10	9	8	10	10	12	12	12
BIC	0	0	3	0	–	3	2	2	3	3	12	12	12
BD1	2	1	3	2	9	–	6	3	4	6	12	12	12
BD2	1	2	4	3	10	6	–	6	6	7	12	12	12
BD4	1	1	4	4	10	9	6	–	8	8	12	12	12
BD8	0	1	4	2	9	8	6	4	–	9	12	12	12
BD16	1	0	5	2	9	6	5	4	3	–	12	12	12
PC90	0	0	0	0	0	0	0	0	0	0	–	7	7
PC95	0	0	0	0	0	0	0	0	0	0	5	–	9
PC99	0	0	0	0	0	0	0	0	0	0	5	3	–

Table 6: Number of times that the methods in rows are better than the ones in columns in terms of the Kullback-Leibler divergence

pendently of the search issues. One way to do this is to generate an ensemble of networks that were found by the search procedures using the different scores and see how each of the scores rank the networks in this ensemble. So, for each of the sixty databases used in the previous experiments we have considered the ten networks obtained by the different scoring functions, computing the ranking of these networks according to each score. We have also computed the ranking of these networks according to each of the two main performance measures, the KL divergence and the Hamming distance.

Hamming													
	M9999	M999	M99	K2	BIC	BD1	BD2	BD4	BD8	BD16	PC90	PC95	PC99
M9999	–	5	8	12	12	12	12	12	12	12	11	11	11
M999	6	–	9	12	12	12	12	12	12	12	11	11	11
M99	3	3	–	12	12	12	12	12	12	12	9	9	11
K2	0	0	0	–	4	6	8	9	11	12	4	4	4
BIC	0	0	0	8	–	8	10	11	11	12	7	7	7
BD1	0	0	0	6	4	–	10	8	9	10	5	4	5
BD2	0	0	0	4	2	2	–	6	10	10	4	4	4
BD4	0	0	0	3	1	4	5	–	11	11	3	3	3
BD8	0	0	0	1	1	3	2	1	–	10	3	3	2
BD16	0	0	0	0	0	2	2	1	2	–	3	3	3
PC90	1	1	3	8	5	7	8	9	9	9	–	5	7
PC95	1	1	3	8	5	7	8	9	9	9	4	–	8
PC99	1	1	1	8	5	7	8	9	10	9	4	2	–

Table 7: Number of times that the methods in rows are better than the ones in columns in terms of the Hamming distance

To measure the degree of association between the rankings generated by each scoring function and each measure of performance, we have used the nonparametric Spearman correlation coefficient[14] for ordinal data (Hogg and Craig, 1994), which varies between -1 (perfect negative correlation) and $+1$ (perfect positive correlation).

Tables 8 and 9 display the average values of the Spearman coefficient with respect to Hamming distance and KL divergence, respectively, grouped by problem and database size.

Average Spearman correlation w.r.t. Hamming distance								
	Problem				Database size			All
	Alarm	Boblo	Hailfinder	Insurance	1000	5000	10000	
M9999	**0.69**	*0.97*	**0.74**	**0.69**	**0.83**	**0.72**	**0.77**	**0.77**
M999	0.62	**0.98**	*0.71*	*0.68*	*0.81*	*0.70*	*0.73*	*0.75*
M99	0.53	0.96	0.66	0.65	0.77	0.65	0.68	0.70
K2	0.55	0.63	-0.02	0.21	0.32	0.27	0.44	0.34
BIC	*0.67*	0.93	0.60	0.61	0.75	0.64	0.72	0.70
BD1	0.44	0.50	-0.40	0.40	0.12	0.18	0.40	0.23
BD2	0.41	0.29	-0.39	0.42	0.06	0.12	0.35	0.18
BD4	0.32	-0.12	-0.42	0.38	-0.13	-0.03	0.28	0.04
BD8	0.20	-0.59	-0.48	0.35	-0.27	-0.17	0.06	-0.13
BD16	-0.02	-0.77	-0.53	0.21	-0.50	-0.28	-0.05	-0.28

Table 8: Average values of the Spearman correlation coefficient between the rankings generated by each scoring function and the Hamming distance

These results confirm that, in terms of the KL divergence, MIT and K2 are the best scores (with K2 being in this case slightly better than MIT), whereas MIT and BIC are the best scores in terms of

14. $\rho = 1 - \frac{6\sum_{i=1}^{N} d_i^2}{N(N^2-1)}$, where $\{d_i\}$ are the differences between the ranks of each observation on the two variables.

Average Spearman correlation w.r.t. KL divergence								
	Problem				Database size			All
	Alarm	Boblo	Hailfinder	Insurance	1000	5000	10000	
M9999	0.80	0.72	*0.51*	0.77	0.66	0.68	*0.76*	0.70
M999	0.83	*0.74*	0.47	0.80	*0.71*	*0.69*	0.74	*0.71*
M99	*0.85*	*0.74*	0.34	0.82	0.70	0.66	0.71	0.69
K2	**0.92**	**0.81**	**0.55**	0.70	**0.76**	**0.70**	**0.77**	**0.74**
BIC	0.48	0.65	0.33	0.30	0.34	0.44	0.55	0.44
BD1	0.84	0.51	-0.23	0.73	0.29	0.51	0.59	0.46
BD2	0.84	0.38	-0.17	0.79	0.28	0.52	0.58	0.46
BD4	0.84	0.05	-0.08	*0.83*	0.20	0.47	0.55	0.41
BD8	0.79	-0.37	-0.01	**0.85**	0.17	0.39	0.38	0.31
BD16	0.61	-0.51	-0.01	*0.83*	0.01	0.34	0.35	0.23

Table 9: Average values of the Spearman correlation coefficient between the rankings generated by each scoring function and the KL divergence

the Hamming distance (with MIT being better than BIC). In our opinion, the fact that MIT behaves very good in terms of both structural and distributional quality support the conclusion that it is a very competitive scoring function.

5.3 Experiments in Automatic Classification

As we commented previously, another approach to evaluating the quality of a scoring function is to use it to learn a Bayesian network classifier, and then to measure the performance of the classifier, for example in terms of predictive accuracy. In this section, we shall apply this method in order to compare MIT with the other scores.

Since the objective of a classifier is not to obtain a good representation of a joint probability distribution for the class and the attributes but rather one for the posterior probability distribution of the class given the attributes, several specialized algorithms that carry out the search into different types of restricted DAG topologies have been developed (Acid et al., 2005; Cheng and Greiner, 1999; Ezawa et al., 1996; Friedman, Geiger and Goldszmidt, 1997; Sahami, 1996), most of these being extensions (using augmenting arcs) or modifications of the well-known Naive Bayes basic topology. This approach generally obtains more satisfactory results than the algorithms for learning unrestricted types of Bayesian networks in terms of classification accuracy.

The BN learning algorithm that we shall use carries out a local search in a space of PDAGs called class-focused RPDAGs (C-RPDAGs), which are RPDAGs representing sets of DAGs which are equivalent in terms of classification (in the sense that they produce the same posterior probabilities for the class variable). Using the BDeu score, this algorithm has proved more effective than other Bayesian network classifiers (Acid et al., 2005).

As in the previous section, we shall first give details of the experimental design before going on to present the obtained results.

5.3.1 EXPERIMENTAL DESIGN

We have selected 29 data sets which were all obtained from the *UCI repository of machine learning databases* (Blake and Merz, 1998), with the exception of 'mofn-3-7-10' and 'corral', which were designed by Kohavi and John (1997). All these data sets have been widely used in specialist literature for comparative purposes in classification.

Table 10 briefly describes the characteristics of each database, including the number of instances, attributes and states for the class variable. Some of these data sets have been preprocessed in the following way: the continuous variables have been discretized using the procedure proposed by Fayyay and Irani (1993), and the instances with undefined/missing values were eliminated. For this preprocessing stage, we have used the MLC++ System (Kohavi et al., 1994).

#	Database	N. cases	N. attributes	N. classes
1	adult	45222	14	2
2	australian	690	14	2
3	breast	682	10	2
4	car	1728	6	4
5	chess	3196	36	2
6	cleve	296	13	2
7	corral	128	6	2
8	crx	653	15	2
9	diabetes	768	8	2
10	flare	1066	10	2
11	german	1000	20	2
12	glass	214	9	7
13	glass2	163	9	2
14	heart	270	13	2
15	hepatitis	80	19	2
16	iris	150	4	3
17	letter	20000	16	26
18	lymphography	148	18	4
19	mofn-3-7-10	1324	10	2
20	mushroom	8124	22	2
21	nursery	12960	8	5
22	pima	768	8	2
23	satimage	6435	36	6
24	segment	2310	19	7
25	shuttle-small	5800	9	7
26	soybean-large	562	35	19
27	vehicle	846	18	4
28	vote	435	16	2
29	waveform-21	5000	21	3

Table 10: Description of the data sets used in the classification experiments

For each database and each scoring function, we have built a classifier using the algorithm based on C-RPDAGs. As in our previous experiments, the probability distributions associated with the obtained network structures have been computed from the data sets using the Laplace estimation.

The selected performance measure is predictive accuracy, that is, the percentage of successful predictions on a test set which is different from the training set. This accuracy has been measured as the average of three runs, the accuracy of each run being estimated using 10-fold cross-validation. Within each run, the cross-validation folds were the same for all the classifiers on each data set.[15] We used repeated runs and 10-fold cross-validation according to the recommendations by Kohavi (1995) in order to obtain a good balance between bias and variance of the estimation.

As these experiments are much more computationally expensive than those in the previous section, instead of using all the different versions of MIT and BDeu, we have selected only one. From the results in Tables 6 and 7, we believe that the best candidate scores are M9999 and BD4. We therefore have a 29×4 design (29 problems and 4 scoring functions), and for each of these 116 configurations, we carry out 3 iterations of 10-fold cross-validation, with a total of 3480 runs of the C-RPDAG learning algorithm.

5.3.2 CLASSIFICATION RESULTS

Table 11 displays the results of these experiments. The best results obtained for each problem are highlighted in bold. We can observe that there are no great differences between the different scoring functions (with the exception perhaps of BIC which seems to behave worst).

In order to determine whether the observed differences are statistically significant, we have also used a non-parametric statistical test: the Wilcoxon paired signed rank test, with a significance level equal to 0.01. We have used this test on each of the three cross-validation iterations. We shall then say that there is a significant difference if the Wilcoxon test detects a difference in at least one of the three iterations, and that there is a *very* significant difference if the test detects differences in all the three iterations. Table 11 also indicates whether the results obtained for K2, BIC and BDeu are significantly worse (–), very significantly worse (– –), significantly better (+) or very significantly better (++) than those of MIT for each data set.

In Table 12, we compare each classifier with the others according to these criteria. The entry in row i column j represents the number of times that classifier i is significantly better or very significantly better than classifier j. These results confirm that K2, BDeu and MIT behave in a similar way, with MIT being slightly better, and that BIC is clearly the worst score.

6. Concluding Remarks

In this paper, we have defined a new scoring function for learning Bayesian networks through score+search algorithms. This is based on the well-known properties of the mutual information measure and which are used in a novel way. We begin with the idea of minimizing the Kullback-Leibler divergence between the joint probability distribution associated with a data set and the one associated with a Bayesian network, which is equivalent to maximizing the sum of the mutual information measures between each variable and its set of parents in the network. We then use a decomposition property of mutual information in order to express each of these measures as a sum of the conditional mutual information measures between the variable and each of its parents, given the subset of the remaining parent variables which antecede the current parent in a given order.

Using another mutual information property that allows us to build an independence test relying on the chi-square distribution, it is possible to interpret mutual information between a variable and

15. The cross-validation folds are in fact the same as those considered by Acid et al. (2005).

#	Database	K2	BIC	BD4	M9999
1	adult	**85.71**	85.42 (–)	85.50	85.66
2	australian	85.65	**86.28**	85.27	85.22
3	breast	**97.56**	**97.56**	97.41	97.36
4	car	93.73	85.63 (– –)	93.83	**94.17**
5	chess	96.50	95.81	**96.71** (+)	96.17
6	cleve	80.54	**82.46**	81.56	82.13
7	corral	**100.00**	**100.00**	**100.00**	**100.00**
8	crx	85.13	**86.61**	86.00	86.00
9	diabetes	**78.65**	78.56	78.60	78.60
10	flare	83.18	82.77	**83.37**	83.21
11	german	74.63	74.40	**74.87**	74.23
12	glass	71.57	70.12	71.56	**71.85**
13	glass2	**85.45**	84.83	85.22	85.44
14	heart	82.47	82.59	**83.21**	82.59
15	hepatitis	90.83	87.50	**92.50**	90.00
16	iris	93.33	94.22	**94.44**	94.22
17	letter	**85.99** (+)	76.73 (– –)	85.55	85.45
18	lymphography	82.83	81.78	**83.49**	81.25
19	mofn-3-7-10	97.36 (–)	93.56 (– –)	99.09	**100.00**
20	mushroom	**100.00**	**100.00**	**100.00**	**100.00**
21	nursery	94.71 (– –)	91.30 (– –)	93.38 (– –)	**95.45**
22	pima	**78.86**	78.51	78.21	78.43
23	satimage	87.84 (–)	84.57 (– –)	88.32	**88.51**
24	segment	94.92	92.16 (– –)	94.55	**95.11**
25	shuttle-small	99.67	**99.79**	99.60	99.65
26	soybean-large	**93.30**	88.85 (–)	92.64	91.81
27	vehicle	**72.46**	71.75	72.10	72.26
28	vote	**94.79**	92.95	93.72	94.03
29	waveform-21	82.47	82.47	**83.06**	82.21
	Average	87.94	86.52	**88.06**	87.97

Table 11: Predictive accuracy of the different scoring functions

	K2	BIC	BD4	M9999
K2	——	9 / **5**	2 / **1**	1 / **0**
BIC	0 / **0**	——	1 / **0**	0 / **0**
BD4	3 / **1**	8 / **6**	——	1 / **0**
M9999	3 / **1**	8 / **6**	1 / **1**	——

Table 12: Number of times that the classifiers in rows are significantly better / **very significantly better** than the ones in columns

its parents as a sum of the statistics associated with a set of simultaneous conditional independence tests. Each of these tests indicates whether it is worth adding a new parent, taking into account those parents which have already been included. The value of each statistic is compared with a

reference value, and the sum of the differences between statistics and reference values is used to quantify the global quality of the parent set. The result is a scoring function (called MIT) which is similar to those based on maximizing a penalized version of the log-likelihood, such as BIC/MDL. In our case, however, the penalization component is specific rather than global for each variable and its parents, and takes into account not only the complexity of the structure but also its reliability. Although MIT is a scoring function, the result of using it within an algorithm based on score and search has many similarities with learning algorithms based on independence tests. However, in our case, the tests are not only used to decide whether the variables are independent or not, but they also quantify the extent to which they are.

We have also carried out a complete experimental evaluation of the proposed score, comparing it with state-of-the-art scoring functions (such as K2, BDeu and BIC/MDL) and with a constraint-based algorithm using different evaluation criteria: structural differences between the original and the learned networks, distance between the probability distributions associated with these networks, and predictive accuracy of the classifiers constructed using the different scores. The results of these experiments show that MIT can compete with the Bayesian scores and that it should be the score of reference within those based on information theory.

The MIT scoring function is decomposable and is not score equivalent, although it satisfies a restricted form of score equivalence which allows us to use it to search not only in the DAG space but also in the RPDAG space. Nevertheless, for future research we would like to develop a scoring function which is based on the same MIT principles but which satisfies the score equivalence property, to be used by learning algorithms that search in the space of essential graphs. Furthermore, the expression of the MIT score depends on a free parameter: the confidence level α associated with the chi-square independence tests. Although experimental results confirm our previous analysis which states that this parameter should be set to a high value (much higher than is usual for a single statistical test), it would also be interesting to find some guidelines in order to automatically select an appropriate value of α depending on the characteristics of the problem domain being considered.

Acknowledgments

I would like to acknowledge support for this work from the Spanish 'Consejería de Innovación Ciencia y Empresa de la Junta de Andalucía', under Project TIC-276. I am grateful to the entire Elvira system development team, especially to my colleagues Silvia Acid, Javier G. Castellano, Serafín Moral and José M. Puerta. Their collaboration in previous work and their contributions to the Elvira system have made the experimental part of this paper possible. I am also grateful to the anonymous reviewers for useful comments and suggestions.

Appendix A

Proof of Theorem 2. We should first explain what a Shur-concave function is. Let us consider two n-dimensional vectors $\mathbf{x} = (x_1,\ldots,x_n)$ and $\mathbf{y} = (y_1,\ldots,y_n)$, and let $\mathbf{x}^{\downarrow} = (x_1^{\downarrow},\ldots,x_n^{\downarrow})$ and $\mathbf{y}^{\downarrow} = (y_1^{\downarrow},\ldots,y_n^{\downarrow})$ be the vectors whose entries are the entries of $\mathbf{x}$ and $\mathbf{y}$, arranged in decreasing order, that is, $x_1^{\downarrow} \geq x_2^{\downarrow} \geq \ldots \geq x_n^{\downarrow}$ and $y_1^{\downarrow} \geq y_2^{\downarrow} \geq \ldots \geq y_n^{\downarrow}$. If $\sum_{j=1}^{m} x_j^{\downarrow} \leq \sum_{j=1}^{m} y_j^{\downarrow}\ \forall m \leq n$, then it is said that $\mathbf{x}$ is majorized by $\mathbf{y}$, written $\mathbf{x} \prec \mathbf{y}$. A function $f : \mathcal{N}^n \longrightarrow \mathcal{R}$ is Shur-concave if for every vector $\mathbf{x} = (x_1,\ldots,x_n)$ and $\mathbf{y} = (y_1,\ldots,y_n)$ such that $\mathbf{x} \prec \mathbf{y}$, then $f(x_1,\ldots,x_n) \geq f(y_1,\ldots,y_n)$. This is one

of the essential properties of entropy and establishes that the more uniform a distribution is, the greater the entropy.

Let us assume that the function $f_{i,\alpha}(l_1,\ldots,l_{s_i}) = \sum_{j=1}^{s_i} \chi_{\alpha,l_j}$ is Shur-concave, and we shall prove the result stated in the theorem. For any permutation σ_i, let us consider the vector $\mathbf{l_{i\sigma_i}} = (l_{i\sigma_i(1)},\ldots, l_{i\sigma_i(s_i)})$. As $r_{ik} \geq 2 \forall k$, then $l_{i\sigma_i(j)} = (r_i-1)(r_{i\sigma_i(j)}-1)\prod_{k=1}^{j-1} r_{i\sigma_i(k)} \leq (r_i-1) r_{i\sigma_i(j)} \prod_{k=1}^{j-1} r_{i\sigma_i(k)} \leq (r_i-1)\ (r_{i\sigma_i(j+1)}-1) r_{i\sigma_i(j)} \prod_{k=1}^{j-1} r_{i\sigma_i(k)} = (r_i-1)(r_{i\sigma_i(j+1)}-1)\prod_{k=1}^{j} r_{i\sigma_i(k)} = l_{i\sigma_i(j+1)}$. Therefore $l_{i\sigma_i(s_i)} \geq \ldots \geq l_{i\sigma_i(2)} \geq l_{i\sigma_i(1)}$, that is, $l^{\downarrow}_{i\sigma_i(1)} = l_{i\sigma_i(s_i)}$,..., $l^{\downarrow}_{i\sigma_i(s_i)} = l_{i\sigma_i(1)}$.

Then, the values of $\sum_{j=1}^{m} l^{\downarrow}_{i\sigma_i(j)}$ can be expressed as follows:

$$\sum_{j=1}^{m} l^{\downarrow}_{i\sigma_i(j)} = \sum_{j=s_i-m+1}^{s_i} l_{i\sigma_i(j)} = \sum_{j=s_i-m+1}^{s_i} \left((r_i-1)(r_{i\sigma_i(j)}-1) \prod_{k=1}^{j-1} r_{i\sigma_i(k)} \right)$$

$$= (r_i-1) \sum_{j=s_i-m+1}^{s_i} \left(r_{i\sigma_i(j)} \prod_{k=1}^{j-1} r_{i\sigma_i(k)} - \prod_{k=1}^{j-1} r_{i\sigma_i(k)} \right) = (r_i-1) \sum_{j=s_i-m+1}^{s_i} \left(\prod_{k=1}^{j} r_{i\sigma_i(k)} - \prod_{k=1}^{j-1} r_{i\sigma_i(k)} \right)$$

$$= (r_i-1) \left(\prod_{k=1}^{s_i} r_{ik} - \prod_{k=1}^{s_i-m} r_{i\sigma_i(k)} \right).$$

As the permutation σ_i^* ranks the variables in decreasing order of the number of states, $\prod_{k=1}^{s_i-m} r_{i\sigma_i(k)} \leq \prod_{k=1}^{s_i-m} r_{i\sigma_i^*(k)}$ and therefore $\sum_{j=1}^{m} l^{\downarrow}_{i\sigma_i^*(j)} \leq \sum_{j=1}^{m} l^{\downarrow}_{i\sigma_i(j)}$, that is, $\mathbf{l_{i\sigma_i^*}} \prec \mathbf{l_{i\sigma_i}}$. By applying the Shur-concavity of $f_{i,\alpha}$, we then obtain $\sum_{j=1}^{s_i} \chi_{\alpha,l_{i\sigma_i(j)}} \leq \sum_{j=1}^{s_i} \chi_{\alpha,l_{i\sigma_i^*(j)}}\ \forall \sigma_i$, hence $\sum_{j=1}^{s_i} \chi_{\alpha,l_{i\sigma_i^*(j)}} = \max_{\sigma_i} \sum_{j=1}^{s_i} \chi_{\alpha,l_{i\sigma_i(j)}}$. ■

Argument supporting Conjecture 3. We try to prove that the functions $f_{i,\alpha}$ are Shur-concave. We shall use the well-known result (Marshall and Olkin, 1979) which states that $\mathbf{x} \prec \mathbf{y}$ if and only if $F(\mathbf{x}) \geq F(\mathbf{y})$, where $F(\mathbf{x}) = \sum_{i=1}^{n} g(x_i)$, for all concave functions f. In our case $F(\mathbf{l}) = f_{i,\alpha}(l_1,\ldots,l_{s_i})$ $= \sum_{j=1}^{s_i} \chi_{\alpha,l_j}$, so that we must only prove that the function $f_\alpha(l) = \chi_{\alpha,l}$ is concave in order to obtain the result. A function $f(l)$ is concave if and only if $\forall l_1 \leq l_2 \leq l_3$, $\frac{f(l_2)-f(l_1)}{l_2-l_1} \geq \frac{f(l_3)-f(l_1)}{l_3-l_1}$, which is equivalent to

$$\forall h,k \geq 0, \forall l,\ (h+k) f(l) \geq k f(l+h) + h f(l-k).$$

We could prove the concavity of f by using induction on the 'distances' h and k. The base case is $h = k = 1$, that is,

$$2 f(l) \geq f(l+1) + f(l-1),\ \forall l. \tag{23}$$

Let us assume that $\forall h \leq h_0, \forall k \leq k_0$, with $k_0 \leq h_0$, $(h+k) f(l) \geq k f(l+h) + h f(l-k) \forall l$. For the values $[l,\ h = h_0,\ k = k_0]$, we then obtain

$$(h_0+k_0) f(l) \geq k_0 f(l+h_0) + h_0 f(l-k_0). \tag{24}$$

Using the values $[l-k_0,\ h = k_0,\ k = 1]$, we now obtain

$$(k_0+1) f(l-k_0) \geq f(l) + k_0 f(l-k_0-1).$$

Simple algebraic manipulations of these two inequalities lead to $(h_0+k_0+1) f(l) \geq (k_0+1) f(l+h_0) + h_0 f(l-k_0-1)$.

Similarly, using the values $[l+h_0,\, h=1,\, k=h_0]$ instead of $[l-k_0,\, h=k_0,\, k=1]$, we obtain

$$(h_0+1)f(l+h_0) \geq h_0 f(l+h_0+1) + f(l). \tag{25}$$

Once again, after algebraic manipulations of the inequalities (24) and (25), we obtain $(h_0+k_0+1)f(l) \geq k_0 f(l+h_0+1) + (h_0+1)f(l-k_0)$. The induction step is therefore complete.

We must still prove the base case. Unfortunately, we have not been able to analytically prove the inequality in Equation 23 when $f(l) = f_\alpha(l) = \chi_{\alpha,l}$. Therefore, in order to prove it empirically, we have built a computer program that computes the values $\chi_{\alpha,l}$ and tests the truth of the inequality. It is obvious that while we cannot compute $\chi_{\alpha,l}$ for all the values of l and α, we can for all the values of practical interest. More specifically, we have tested all the values of l from 2 to 1000 and all the values of α from 0.1000 to 0.9999 with a stepsize of 0.0001. The results of these experiments are as follows: the inequality in Equation 23 is always true from $\alpha = 0.5827$ to 0.9999; from $\alpha = 0.5429$ to 0.5826, it is always true except for the case $l = 2$; from $\alpha = 0.4922$ to 0.5428, the inequality is false for many values of l (the lower α is, the more frequent the number of failures), and from $\alpha = 0.1000$ to 0.4921 it is always false. It can be seen that since the behavior of the function $f_\alpha(l)$ is quite homogeneous, we do not expect it to behave differently for the intermediate values of α which have not been tested. We may therefore conclude that $f_\alpha(l)$ is concave for all the values of α that may be of interest when computing the MIT score. ■

Proof of Theorem 4. We shall use induction on the number of variables in $Pa_G(X_i)$. The base case, where $|Pa_G(X_i)| = 1$, is obviously true. Let us suppose that the result is true when the size of the parent set of X_i is equal to $s_i - 1$ and consider a case where $|Pa_G(X_i)| = s_i$. Then, if σ_{ij} denotes a permutation of the variables in the set $Pa_G(X_i) \setminus \{X_{ij}\}$, we have

$$\begin{aligned}
g_r(X_i, Pa_G(X_i) : D) &= \min_{X_{ij} \in Pa_G(X_i)} \Big\{ g_r(X_i, Pa_G(X_i) \setminus \{X_{ij}\} : D) + \\
&\qquad 2N\,MI_D(X_i, X_{ij} | Pa_G(X_i) \setminus \{X_{ij}\}) - \chi_{\alpha, l^r_{ij}} \Big\} \\
&= \min_{X_{ij} \in Pa_G(X_i)} \Big\{ 2N\,MI_D(X_i, Pa_G(X_i) \setminus \{X_{ij}\}) - \max_{\sigma_{ij}} \sum_{k=1}^{s_i-1} \chi_{\alpha, l_{i\sigma_{ij}(k)}} + \\
&\qquad 2N\,MI_D(X_i, X_{ij} | Pa_G(X_i) \setminus \{X_{ij}\}) - \chi_{\alpha, l^r_{ij}} \Big\} \\
&= \min_{X_{ij} \in Pa_G(X_i)} \Big\{ 2N\,MI_D(X_i, Pa_G(X_i)) - \max_{\sigma_{ij}} \sum_{k=1}^{s_i-1} \chi_{\alpha, l_{i\sigma_{ij}(k)}} - \chi_{\alpha, l^r_{ij}} \Big\} \\
&= 2N\,MI_D(X_i, Pa_G(X_i)) - \max_{X_{ij} \in Pa_G(X_i)} \Big\{ \max_{\sigma_{ij}} \sum_{k=1}^{s_i-1} \chi_{\alpha, l_{i\sigma_{ij}(k)}} + \chi_{\alpha, l^r_{ij}} \Big\} \\
&= 2N\,MI_D(X_i, Pa_G(X_i)) - \max_{X_{ij} \in Pa_G(X_i)} \Big\{ \max_{\sigma_{ij}} \Big\{ \sum_{k=1}^{s_i-1} \chi_{\alpha, l_{i\sigma_{ij}(k)}} + \chi_{\alpha, l^r_{ij}} \Big\} \Big\}.
\end{aligned}$$

The value $\sum_{k=1}^{s_i-1} \chi_{\alpha, l_{i\sigma_{ij}(k)}} + \chi_{\alpha, l^r_{ij}}$ in the last expression can be seen as the value associated with a permutation of the variables in $Pa_G(X_i)$ where the last element is restricted to be X_{ij}, that is, if we define a permutation $\sigma_{i\setminus j}$ as $\sigma_{i\setminus j}(k) = \sigma_{ij}(k)$, $\forall k = 1, \ldots, s_i - 1$ and $\sigma_{i\setminus j}(s_i) = j$, then $\sum_{k=1}^{s_i-1} \chi_{\alpha, l_{i\sigma_{ij}(k)}} + \chi_{\alpha, l^r_{ij}} = \sum_{k=1}^{s_i} \chi_{\alpha, l_{i\sigma_{i\setminus j}(k)}}$.

The union of the sets of permutations of $Pa_G(X_i)$ where the last element is fixed to X_{ij}, for all X_{ij}, is the set of all the permutations of $Pa_G(X_i)$, hence

$$\max_{X_{ij}\in Pa_G(X_i)} \max_{\sigma_{ij}} \left\{ \sum_{k=1}^{s_i-1} \chi_{\alpha,l_{i\sigma_{ij}(k)}} + \chi_{\alpha,l'_{ij}} \right\} = \max_{X_{ij}\in Pa_G(X_i)} \max_{\sigma_{i\setminus j}} \sum_{k=1}^{s_i} \chi_{\alpha,l_{i\sigma_{i\setminus j}(k)}} = \max_{\sigma_i} \sum_{k=1}^{s_i} \chi_{\alpha,l_{i\sigma_i(k)}}.$$

Therefore, we have $g_r(X_i, Pa_G(X_i) : D) = 2N\,MI_D(X_i, Pa_G(X_i)) - \max_{\sigma_i} \sum_{k=1}^{s_i} \chi_{\alpha,l_{i\sigma_i(k)}}$ and the result is also true for parent sets of X_i with size equal to s_i. This completes the induction step. ■

Proof of Theorem 5. This result is evident as the scoring function is, by definition, a sum of local scores. ■

Proof of Theorem 6. As all DAGs that are represented by the same RPDAG have the same skeleton and the same head-to-head patterns (either coupled or uncoupled), then the differences between these DAGs can only be due to the different direction of certain arcs linking two nodes X_i and X_j that have at most a single parent. In such cases, the chi-square value associated with the local score of the corresponding node (either X_i or X_j) is always the same, $\chi_{\alpha,l}$, with $l = (r_i - 1)(r_j - 1)$. ■

References

B. Abramson, J. Brown, A. Murphy, and R. L. Winkler. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12:57–71, 1996.

S. Acid and L. M. de Campos. Learning right sized belief networks by means of a hybrid methodology. *Lecture Notes in Artificial Intelligence*, 1910:309–315, 2000.

S. Acid and L. M. de Campos. A hybrid methodology for learning belief networks: Benedict. *International Journal of Approximate Reasoning*, 27:235–262, 2001.

S. Acid and L. M. de Campos. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

S. Acid, L. M. de Campos, and J. G. Castellano. Learning Bayesian network classifiers: searching in a space of partially directed acyclic graphs. *Machine Learning*, 59:213–235, 2005.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

S. Andersson, D. Madigan, and M. Perlman. A Characterization of Markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25:505–541, 1997.

I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the European Conference on Artificial Intelligence in Medicine*, pages 247–256, 1989.

J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

C. L. Blake and C. J. Merz. UCI Repository of machine learning databases. *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

R. Blanco, I. Inza, and P. Larrañaga. Learning Bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, 18:205–220, 2003.

R. R. Bouckaert. Belief networks construction using the minimum description length principle. *Lecture Notes in Computer Science*, 747:41–48, 1993.

R. R. Bouckaert. *Bayesian Belief Networks: from Construction to Inference*. PhD thesis, University of Utrecht, 1995.

W. Buntine. Theory refinement of Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

J. Cheng, R. Greiner, J. Kelly, D. A. Bell, and W. Liu. Learning Bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, 137:43–90, 2002.

J. Cheng and R. Greiner. Comparing Bayesian network classifiers. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 101–108, 1999.

D. M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 87–98, 1995.

D. M. Chickering. Learning equivalence classes of Bayesian network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128, 1995.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–348, 1992.

D. Dash and M. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 142–149, 1999.

L. M. de Campos. Independency relationships and learning algorithms for singly connected networks. *Journal of Experimental and Theoretical Artificial Intelligence*, 10:511–549, 1998.

L. M. de Campos, J. M. Fernández-Luna, J. A. Gámez, and J. M. Puerta. Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, 31:291–311, 2002.

L. M. de Campos, J. M. Fernández-Luna, and J. M. Puerta. Local search methods for learning Bayesian networks using a modified neighborhood in the space of dags. *Lecture Notes in Computer Science*, 2527:182–192, 2002.

L. M. de Campos, J. M. Fernández-Luna, and J. M. Puerta. An iterated local search algorithm for learning Bayesian networks with restarts based on conditional independence tests. *International Journal of Intelligent Systems*, 18:221–235, 2003.

L. M. de Campos, J. A. Gámez, and J. M. Puerta. Learning Bayesian networks by ant colony optimization: Searching in two different spaces. *Mathware and Soft Computing*, IX:251–268, 2002.

L. M. de Campos and J. F. Huete. A new approach for learning belief networks using independence criteria. *International Journal of Approximate Reasoning*, 24:11–37, 2000.

L. M. de Campos and J. F. Huete. Stochastic algorithms for searching causal orderings in Bayesian networks. In *Technologies for Constructing Intelligent Systems 2 - Tools*, B. Bouchon-Menieur, J. Gutiérrez-Rios, L. Magdalena, R.R. Yager (Eds.), Physica-Verlag, pages 327–340, 2002.

L. M. de Campos and J. M. Puerta. Stochastic local and distributed search algorithms for learning belief networks. In *Proceedings of the III International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Model*, pages 109–115, 2001.

L. M. de Campos and J. M. Puerta. Stochastic local search algorithms for learning belief networks: Searching in the space of orderings. *Lecture Notes in Artificial Intelligence*, 2143:228–239, 2001.

Elvira Consortium. Elvira: An environment for probabilistic graphical models. In *Proceedings of the First European Workshop on Probabilistic Graphical Models*, pages 222–230, 2002. Available at http://www.leo.ugr.es/∼elvira.

M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions*, Second edition. Wiley, 1993.

K. Ezawa, M. Singh, and S. Norton. Learning goal oriented Bayesian networks for telecommunications risk management. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 139–147, 1996.

U. M. Fayyad and K. B. Irani. Multi-valued interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 252–262, 1996.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

N. Friedman and D. Koller. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–126, 2003.

I. J. Good. *The Estimation of Probabilities*. MIT Press, 1965.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

E. Herskovits and G. F. Cooper. Kutató: An entropy-driven system for the construction of probabilistic expert systems from databases. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 54–62, 1990.

I. D. Hill and M. C. Pike. Algorithm 299: Chi-squared integral. *Communications of the ACM*, 10:243–244, 1965.

I. D. Hill and M. C. Pike. Remark on algorithm 299: Chi-squared integral. *ACM Transactions on Mathematical Software*, 11:185–185, 1985.

R. V. Hogg and A. T. Craig. *Introduction to Mathematical Statistics*, 5th Edition. Prentice Hall, New York, 1994.

F. V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, 1996.

M. Kayaalp and G. F. Cooper. A Bayesian network scoring metric that is based on globally uniform parameter priors. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 251–258, 2002.

T. Kocka and R. Castelo. Improved learning of Bayesian networks. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 269–276, 2001.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1137–1143, 1995.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: A machine learning library in C++. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, pages 740–743, 1994.

S. Kullback. *Information Theory and Statistics*. Dover Publication, 1968.

W. Lam and F. Bacchus. Learning Bayesian belief networks. An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:912–926, 1996.

P. Larrañaga, C. Kuijpers, and R. Murga. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on System, Man and Cybernetics*, 26:487–493, 1996.

D. Madigan, S. A. Andersson, M. D. Perlman, and C. T. Volinsky. Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Communications in Statistics – Theory and Methods*, 25:2493–2520, 1996.

A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and Its Applications.* Academic Press, New York, 1979.

C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 403–410, 1995.

J. W. Myers, K. B. Laskey, and T. Levitt. Learning Bayesian networks from incomplete data with stochastic search algorithms. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 476–485, 1999.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.

J. Pearl and T. S. Verma. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1990.

J. Pearl and T. S. Verma. A theory of inferred causation. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 441–452, 1991.

L. K. Rasmussen. Bayesian network for blood typing and parentage verification of cattle. PhD thesis, Research Centre Foulum, Denmark, 1995.

J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

M. Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338, 1996.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

M. Singh and M. Valtorta. Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. *International Journal of Approximate Reasoning*, 12:111–131, 1995.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Lecture Notes in Statistics 81, Springer Verlag, New York, 1993.

P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 294–299, 1995.

J. Suzuki. A construction of Bayesian networks from databases based on the MDL principle. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 266–273, 1993.

J. Tian. A branch-and-bound algorithm for MDL learning Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 580–587, 2000.

T. Verma and J. Pearl. Causal networks: Semantics and expressiveness. In *Uncertainty in Artificial Intelligence, 4*, R.D. Shachter, T.S. Lewitt, L.N. Kanal, J.F. Lemmer (Eds.), North-Holland, Amsterdam, pages 69–76, 1990.

N. Wermuth and S. Lauritzen. Graphical and recursive models for contingence tables. *Biometrika*, 72:537–552, 1983.

M. L. Wong, W. Lam, and K. S. Leung. Using evolutionary computation and minimum description length principle for data mining of probabilistic knowledge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:174–178, 1999.

S. Yang and K. Chang. Comparison of score metrics for Bayesian network learning. *IEEE Transactions on System, Man and Cybernetics–Part A: Systems and Humans*, 32:419–428, 2002.

Journal of Machine Learning Research 7 (2006) 2189-2213 Submitted 1/06; Revised 7/06; Published 10/06

Noisy-OR Component Analysis and its Application to Link Analysis

Tomáš Šingliar TOMAS@CS.PITT.EDU
Miloš Hauskrecht MILOS@CS.PITT.EDU
Department of Computer Science
5329 Sennott Square
University of Pittsburgh
Pittsburgh, PA 15260

Editor: David Maxwell Chickering

Abstract

We develop a new component analysis framework, the *Noisy-Or Component Analyzer* (NOCA), that targets high-dimensional binary data. NOCA is a probabilistic latent variable model that assumes the expression of observed high-dimensional binary data is driven by a small number of hidden binary sources combined via noisy-or units. The component analysis procedure is equivalent to learning of NOCA parameters. Since the classical EM formulation of the NOCA learning problem is intractable, we develop its variational approximation. We test the NOCA framework on two problems: (1) a synthetic image-decomposition problem and (2) a co-citation data analysis problem for thousands of CiteSeer documents. We demonstrate good performance of the new model on both problems. In addition, we contrast the model to two mixture-based latent-factor models: the probabilistic latent semantic analysis (PLSA) and latent Dirichlet allocation (LDA). Differing assumptions underlying these models cause them to discover different types of structure in co-citation data, thus illustrating the benefit of NOCA in building our understanding of high-dimensional data sets.

Keywords: component analysis, vector quantization, variational learning, link analysis

1. Introduction

Latent variable (or *latent factor*) models (MacKay, 1995; Bishop, 1999a) provide an elegant framework for modeling dependencies in high-dimensional data. Suppose that two observed random variables x_i, x_j are marginally dependent. A latent variable model explains their dependency by positing the presence of a hidden variable s representing their common cause. Examples of latent factor models include probabilistic principal component analysis (Tipping and Bishop, 1997; Bishop, 1999b), mixtures of factor analyzers (Attias, 1999), multinomial PCA (or *aspect*) models (Buntine, 2002; Hofmann, 1999a; Blei et al., 2003), the multiple cause model (Ghahramani and Jordan, 1995; Ross and Zemel, 2002) and independent component analysis frameworks (Attias, 1999; Miskin, 2000). The models are most often used for component analysis, where we want to identify a small number of underlying components (factors, sources, or signals) whose effects combine to form the observed data. Once a model is learned, it can be used to make inferences on hidden factors, such as to identify the document topics in the aspect model (Hofmann, 1999a; Blei et al., 2003) or regulatory signals in the microarray DNA data (Lu et al., 2004). In addition to their role in understanding the structure of high-dimensional data, latent factor models can be applied in dimensionality reduction, where the hidden factor values are a low-dimensional representation of the data sample.

Factor and principal component analysis methods (Bartholomew and Knott, 1999; Jolliffe, 1986) and other component analysis frameworks (Attias, 1999) are traditionally applied to high-dimensional continuous-valued data. More recently, multinomial mixture models (Hofmann, 1999a; Blei et al., 2003) were shown to handle many-valued discrete variables successfully. However, component analysis methods specifically tailored to binary data remain scarce. In this work, we investigate a latent factor model designed for analysis of high-dimensional *binary* data. The dependencies between observables are represented using a small number of hidden binary factors whose effects are combined through noisy-or units. We therefore refer to the model as to "noisy-or component analyzer" (NOCA). Binary variables can, for instance, represent failures or congestions in transportation networks, spread of disease in epidemiology, or the presence of a link in a citation graph.

The principal limitation of latent factor models is the complexity of their learning (or *parameter estimation*), as the standard EM formulation becomes exponential in the number of hidden factors. To address the problem, we adopt a variational inference algorithm for bipartite noisy-or (B2NO) networks (Jaakkola and Jordan, 1999) and derive the corresponding learning algorithm for the model with hidden sources.

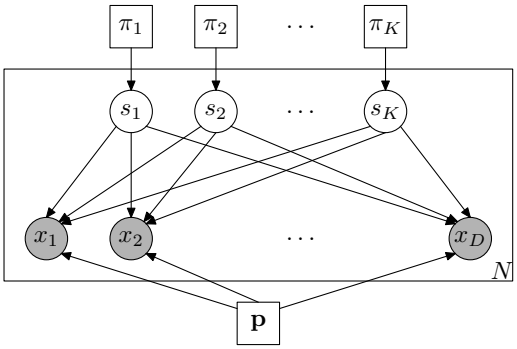
Two aspects of the new method are evaluated: (1) the quality of the approximate learning algorithm and (2) the adequacy of the model for real-world data. We use two different data sets to evaluate NOCA and its learning algorithm: a synthetic image-decomposition problem and a co-citation data analysis problem. The knowledge of the underlying model and hidden factors in the first problem (image data) enables us to assess the performance of the learning algorithm and its ability to recover the model. We judge the quality of the recovery both qualitatively and quantitatively in terms of the likelihood of test data and data reconstruction error. Running-time analysis verifies the expected polynomial scale-up.

The second evaluation problem is an application of NOCA to link and citation analysis. Citation data from over 6000 CiteSeer documents were extracted and analyzed with NOCA. To measure how well NOCA's hidden sources capture the co-citation relationships, we use a cosine-distance based metric and an inspection by a human judge. Perplexity of the testing set is used to gauge the predictive power of the learned model. NOCA results are compared to mixture-based latent variable models, represented by probabilistic latent semantic analysis (Hofmann, 1999a; Cohn and Chang, 2000) and its Bayesian extension—latent Dirichlet allocation (Blei et al., 2003). The mixture models view a document differently from NOCA. In consequence, each model class sees different facets of the data structure. NOCA's benefit is in the discovery of publication subcommunities in the data that the mixture models tend to overlook.

2. Noisy-OR Component Analysis

Technically, the noisy-or component analysis (NOCA) is a latent variable model with binary variables defined by a bipartite belief network structure in Figure 1.

The nodes in the top layer represent a vector of latent factors $\mathbf{s} = \{s_1, s_2, \ldots, s_K\}$ ("sources") with binary values $\{0,1\}$ and the nodes in the bottom layer an observable vector of binary features $\mathbf{x} = \{x_1, x_2, \ldots, x_D\}$. The connections between the two layers represent dependencies among the observables: the nodes coupled by a latent factor can exhibit a local dependency pattern. Parameterizing the bottom-layer nodes with noisy-or units reduces the model's parameter space to $KD+K+D$ free parameters:



Notation:
D – observable dimensionality
K – latent dimensionality, $D > K$
N – number of data points
$\mathbf{x}$ – observables, indexed by j: x_j
$\mathbf{s}$ – latent sources, indexed by i: s_i
Parameters (square nodes):
$\mathbf{p}$ – loading matrix (with leak terms)
$\{\pi_i\}$ – source priors

Figure 1: The NOCA model in plate notation. Shaded nodes correspond to observables. (In the entire text, boldface letters will denote vectors or matrices.)

- a set of K prior probabilities π_i parameterizing the (Bernoulli) prior distributions $P(s_i)$ for every hidden factor s_i;
- a set of DK parameters $\mathbf{p} = \{p_{ij}\}_{j=1,\ldots,D}^{i=1,\ldots,K}$ of the noisy-or conditional probability tables, one for each pair of hidden factor i and observed component j.
- a set of D parameters p_{0j} representing "other causes." These can be incorporated into $\mathbf{p}$ by positing a latent factor s_0 with $p(s_0 = 1) = 1$, where notationally convenient.

The NOCA model resembles the QMR-DT model (Shwe et al., 1991) in the structure and type of nodes used. However, it is from the outset assumed to be fully connected. The model is simplified during learning by setting the weight of most connections to zero.[1] NOCA makes no assumption as to the interpretation of random variables. For example, although features might correspond to words when analyzing text documents; citation indicator variables will be used when analyzing references among scholarly articles.

2.1 The Joint Distribution over Observables

The joint probability of an observation vector $P(\mathbf{x})$ exemplifies and subsumes the probabilistic queries we need to evaluate. Given the bipartite model, $P(\mathbf{x})$ is obtained as

$$P(\mathbf{x}) = \sum_{\{\mathbf{s}\}} \left(\prod_{j=1}^{d} P(x_j|\mathbf{s}) \right) \left(\prod_{i=1}^{K} P(s_i) \right), \tag{1}$$

where $\{\mathbf{s}\}$ denotes the sum over all configurations of $\mathbf{s}$, and $P(s_i)$ is the prior probability of a hidden factor s_i. Given a vector of hidden binary factors $\mathbf{s}$, the conditional probability $p(x_j|\mathbf{s})$ for an

1. This is in contrast with the structure-learning algorithm proposed by Kearns and Mansour (1998). Their algorithm is exponential in the maximum number of hidden factors contributing to any observable variable. Therefore, they limit the in-degree of the bottom layer nodes to obtain a polynomial algorithm. Our algorithm does not make any such structural assumption.

observable random component $x_j \in \{0,1\}$ is obtained through the noisy-or model:

$$P(x_j|\mathbf{s}) = \left[1-(1-p_{0j})\prod_{i=1}^{K}(1-p_{ij})^{s_i}\right]^{x_j}\left[(1-p_{0j})\prod_{i=1}^{K}(1-p_{ij})^{s_i}\right]^{(1-x_j)}, \quad (2)$$

where p_{0j} is the leak probability that models "all other" causes.

Equation 2 can be reparameterized with $\theta_{ij} = -\log(1-p_{ij})$ to obtain:

$$P(x_j|\mathbf{s}) = \exp\left[x_j \log\left(1-\exp\left\{-\theta_{0j}-\sum_{i=1}^{k}\theta_{ij}s_i\right\}\right)+(1-x_j)\left(-\theta_{0j}-\sum_{i=1}^{K}\theta_{ij}s_i\right)\right]. \quad (3)$$

This reparameterization will prove useful in the following description of the variational lower bound.

2.2 The Factorized Variational Bound

The bottleneck in computing the joint probability over observables, $P(\mathbf{x})$ in Equation 1, is the sum that ranges over all possible latent factor configurations. However, it is easy to see that if $P(x_j|\mathbf{s})$ for both $x_j = 0$ and $x_j = 1$ could be expressed in a factored form as:

$$P(x_j|\mathbf{s}) = \prod_{i=1}^{K} h(x_j|s_i), \text{ such that } \forall i,j : h(x_j|s_i) \geq 0, \quad (4)$$

then the full joint $P(\mathbf{x},\mathbf{s})$ and the joint over the observables $P(\mathbf{x})$ would decompose:

$$\begin{aligned} P(\mathbf{x},\mathbf{s}) &= \prod_{j=1}^{d}P(x_j|\mathbf{s})\prod_{i=1}^{K}P(s_i) = \prod_{i=1}^{K}\left(P(s_i)\prod_{j=1}^{d}h(x_j|s_i)\right), \\ P(\mathbf{x}) &= \sum_{\{\mathbf{s}\}}\prod_{i=1}^{K}\left(P(s_i)\prod_{j=1}^{d}h(x_j|s_i)\right) = \prod_{i=1}^{K}\left(\sum_{\{s_i\}}P(s_i)\left[\prod_{j=1}^{d}h(x_j|s_i)\right]\right). \end{aligned}$$

Such decomposition would imply that the summation in Equation 1 can be performed efficiently. Note that the condition of Equation 4 is sufficient to ensure tractability of other inference queries, such as the posterior of a hidden factor s_i:

$$P(s_i|\mathbf{x}) \propto P(s_i)\prod_{j=1}^{d}h(x_j|s_i). \quad (5)$$

However, while Equation 3 defining $P(x_j|\mathbf{s})$ decomposes for $x_j = 0$, it does not factorize for $x_j = 1$. Thus, in general, it is impossible to compute $P(\mathbf{x})$ efficiently. We approximate $P(x_j|\mathbf{s})$ for $x_j = 1$ with a factored variational lower bound (Jaakkola and Jordan, 1999):

$$P(x_j=1|\mathbf{s}) \geq \quad (6)$$
$$\tilde{P}(x_j|\mathbf{s}) = \prod_{i=1}^{K}\exp\left\{q_j(i)s_i\left[\log(1-e^{-\theta_{0j}-\frac{\theta_{ij}}{q_j(i)}})-\log(1-e^{-\theta_{0j}})\right]+q_j(i)\log(1-e^{-\theta_{0j}})\right\},$$

where q_js represent sets of variational parameters defining a multinomial distribution. Each component $q_j(i)$ of the distribution can be viewed as a responsibility of a latent factor s_i for observing $x_j = 1$. If we denote the complex expression inside the product on the right-hand side of Equation 6 by $h(x_j|s_i)$, we have the sought-after decomposition.

Incorporating the variational bound into the first, nondecomposing term in Equation 3, we can obtain approximations $\tilde{P}(\mathbf{x}|\mathbf{s},\Theta,\mathbf{q}) \leq P(\mathbf{x}|\mathbf{s},\Theta)$, $\tilde{P}(\mathbf{x},\mathbf{s}|\Theta,\mathbf{q}) \leq P(\mathbf{x},\mathbf{s}|\Theta)$ and $\tilde{P}(\mathbf{x}|\Theta,\mathbf{q}) \leq P(\mathbf{x}|\Theta)$ that factorize along latent factors s_i.

3. The Variational Learning Algorithm

The key step of component analysis corresponds to the learning of the latent factor model from data. The problem of learning of bipartite noisy-or networks has been addressed only in the fully observable setting; that is, when both the sources and observations are known. The learning methods take advantage of the decomposition of the model created by the introduction of special hidden variables (Heckerman, 1993; Vomlel, 2003; Diez and Gallan, 2003). The EM algorithm is then used to estimate the parameters of the modified network, which translate directly into the parameters of the original model. However, to our knowledge, no learning algorithm for B2NO networks has been derived for the case of unobservable source layer.

In this section, we motivate and detail the derivation of the variational learning algorithm, following the EM-framework. We identify the crucial hurdles in deriving an efficient algorithm and show how the variational approximation overcomes them.

3.1 Classical EM Formulation

Let $D = \{\mathbf{x}^1, \mathbf{x}^2, \cdots \mathbf{x}^N\}$ be a set of N i.i.d. vectors of observable variables. Our objective is to find parameters Θ that maximize the likelihood of the data, $P(D|\Theta)$. The standard approach to learn the parameters of the model in the presence of hidden variables is the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). EM computes the parameters iteratively by taking the following parameter update step:

$$\Theta^* = \arg\max_{\Theta} \sum_{n=1}^{N} \langle \log P(\mathbf{x}^n, \mathbf{s}^n|\Theta) \rangle_{P(\mathbf{s}^n|\mathbf{x}^n,\Theta')},$$

where Θ' denotes previous-step parameters.

The main problem in applying the EM to the noisy-or model is that the joint distribution over every "completed" sample $P(\mathbf{x}^n, \mathbf{s}^n|\Theta)$ does not decompose along hidden factors s_i and thus its expectation $\langle \log P(\mathbf{x}^n, \mathbf{s}^n|\Theta) \rangle_{P(\mathbf{s}^n|\mathbf{x}^n,\Theta')}$ requires iteration over all possible latent factor configurations. This is infeasible since the configuration space grows exponentially in the number of factors. Note that even if we could solve the inference query $P(\mathbf{s}^n|\mathbf{x}^n,\Theta')$ efficiently, we still cannot push the expectations inward over the nonlinearities—we also need to decompose the term inside the expectation.

3.2 Variational EM

The idea of variational methods is to approximate the likelihood terms with their imprecise, but structurally more convenient surrogates. In summary, an additional set of free variational parameters $\mathbf{q}$ (Section 2.2) is introduced that offers the flexibility to perform more efficient calculations of

the joint and posterior distributions within the EM algorithm. In particular, we replace the true conditional probabilities $P(\mathbf{x}^n|\mathbf{s}^n,\Theta)$ that do not factorize with their factored lower-bound variational approximation $\tilde{P}(\mathbf{x}^n|\mathbf{s}^n,\Theta,\mathbf{q}^n)$ as described in Section 2.2. As a consequence, the approximate posterior $\tilde{P}(\mathbf{s}^n|\mathbf{x}^n,\Theta,\mathbf{q}^n)$ also factorizes, which simplifies the expectation step of the algorithm. The new EM algorithm iteration becomes:

$$\Theta^* = \arg\max_{\Theta} \sum_{n=1}^{N} \left\langle \log \tilde{P}(\mathbf{x}^n,\mathbf{s}^n|\Theta,\mathbf{q}^n) \right\rangle_{\tilde{P}(\mathbf{s}^n|\mathbf{x}^n,\Theta',\mathbf{q}^{n'})},$$

where Θ' and $\mathbf{q}^{n'}$ denote previous-step model and variational parameters.

In ML learning, we maximize $\log P(D|\Theta)$ with respect to Θ. In NOCA, we maximize a lower bound on $\log P(D|\Theta)$ instead, to ease the computational complexity brought by hidden variables. First, let us simplify the expectation distribution—the hidden source posterior:

$$\begin{aligned}
\log P(D|\Theta) &= \log \prod_{n=1}^{N} P(\mathbf{x}^n|\Theta) \\
&= \sum_{n=1}^{N} \log \left[\sum_{\{\mathbf{s}^n\}} P(\mathbf{x}^n,\mathbf{s}^n|\Theta) \right] \\
&= \sum_{n=1}^{N} \log \left[\sum_{\{\mathbf{s}^n\}} P(\mathbf{x}^n,\mathbf{s}^n|\Theta,\mathbf{q}^n) \frac{Q(\mathbf{s}^n)}{Q(\mathbf{s}^n)} \right] \\
&\geq \sum_{n=1}^{N} \left[\sum_{\{\mathbf{s}^n\}} \langle \log P(\mathbf{x}^n,\mathbf{s}^n|\Theta) \rangle_{Q(\mathbf{s}^n)} - \langle \log Q(\mathbf{s}^n) \rangle_{Q(\mathbf{s}^n)} \right].
\end{aligned}$$

This lower bound follows from Jensen's inequality for any arbitrary distribution over the hidden sources $Q(H) = \prod_{n=1}^{N} Q(\mathbf{s}^n)$ (Jordan et al., 1999; Saul et al., 1996; Ghahramani and Jordan, 1997). However, even with a decomposable Q, we cannot take expectations of $\log P(\mathbf{x}^n,\mathbf{s}^n|\Theta)$ easily, because the noisy-or distribution is not in the exponential family and the s_i's reside inside nonlinearities. We apply Equation 6 to obtain a further lower bound:

$$\begin{aligned}
\log P(D|\Theta) &\geq \sum_{n=1}^{N} \left[\sum_{\{\mathbf{s}^n\}} \langle \log P(\mathbf{x}^n,\mathbf{s}^n|\Theta) \rangle_{Q(\mathbf{s}^n)} - \langle \log Q(\mathbf{s}^n) \rangle_{Q(\mathbf{s}^n)} \right] \\
&\geq \sum_{n=1}^{N} \left[\sum_{\{\mathbf{s}^n\}} \left\langle \log \tilde{P}(\mathbf{x}^n,\mathbf{s}^n|\Theta,\mathbf{q}^n) \right\rangle_{Q(\mathbf{s}^n)} - \langle \log Q(\mathbf{s}^n) \rangle_{Q(\mathbf{s}^n)} \right] \\
&= \sum_{n=1}^{N} \left[\sum_{\{\mathbf{s}^n\}} \left\langle \log \tilde{P}(\mathbf{x}^n|\mathbf{s}^n,\Theta,\mathbf{q}^n) P(\mathbf{s}^n|\Theta) \right\rangle_{Q(\mathbf{s}^n)} - \langle \log Q(\mathbf{s}^n) \rangle_{Q(\mathbf{s}^n)} \right] \\
&= \sum_{n=1}^{N} \mathcal{F}_n(\mathbf{x}^n, Q(\mathbf{s}^n)) \\
&= \mathcal{F}(D, Q(H)),
\end{aligned}$$

where $\mathbf{q}^n$ are parameters of the lower bound approximation described in Section 2.2.

The variational EM that optimizes the bound also proceeds, like standard EM, in two steps. The E-step computes the expectation distribution $Q(\mathbf{s}^n)$. We could in principle choose any distribution Q, but it is desirable to choose one that makes the variational bound as tight as possible. The variational bound of $\log P(D|\Theta)$ is the tightest at $Q(\mathbf{s}^n) = P(\mathbf{s}^n|\mathbf{x}^n,\Theta)$. Since that ideal posterior is intractable, we define $Q(\mathbf{s}^n)$ to be the tractable posterior probability $\tilde{P}(\mathbf{s}^n|\mathbf{x}^n,\Theta',\mathbf{q}^n)$, where Θ' are fixed previous step parameters and $\mathbf{q}^n$ are tuned to obtain the best approximation to the true posterior. (Alternatively, we could separately and explicitly optimize Q to maximize $\mathcal{F}(D,Q(H))$.)

The new $\mathbf{q}^n$ s are obtained that maximize $\tilde{P}(\mathbf{x}^n|\mathbf{s}^n,\Theta',\mathbf{q}^n)$ by an iterative procedure described in Figure 2. These iterative updates essentially form an embedded EM loop and are derived in Jaakkola et al. (1996). The subsequent computation of $\tilde{P}(\mathbf{s}^n|\mathbf{x}^n,\Theta',\mathbf{q}^n)$ decomposes along the hidden factors and can be performed in linear time according to Equation 5. Obtaining the posteriors on hidden sources concludes the E-step.

The M-step optimizes $\mathcal{F}(D,Q(H))$ with respect to Θ. Given the decomposable $Q(\mathbf{s}^n)$, $\mathcal{F}_n(\mathbf{x}^n,Q(\mathbf{s}^n))$ can be rewritten as:

$$\begin{aligned}
\mathcal{F}_n(\mathbf{x}^n,Q(\mathbf{s}^n)) &= \left\langle \log\tilde{P}(\mathbf{x}^n|\mathbf{s}^n,\mathbf{q}^n,\Theta)\right\rangle_{Q(\mathbf{s}^n)} - \left\langle Q(\mathbf{s}^n)\right\rangle_{Q(\mathbf{s}^n)} \\
&= \left[\sum_{i=1}^{K}\langle s_i^n\rangle_{Q(s_i^n)}\log\frac{\pi_i}{(1-\pi_i)} + \log(1-\pi_i)\right] \\
&+ \left[\sum_{j=1}^{d}\left(\sum_{i=1}^{K} -\langle s_i^n\rangle_{Q(s_i^n)}\theta_{ij}(1-x_j^n)\right) - \theta_{0j}(1-x_j^n)\right] \qquad (7)\\
&+ \sum_{j=1}^{d}\sum_{i=1}^{K}\left[\langle s_i^n\rangle_{Q(s_i^n)} q_j^n(i) x_j^n \log\left(1-e^{-\theta_{0j}-\frac{\theta_{ij}}{q_j^n(i)}}\right) + \left(1-\langle s_i^n\rangle_{Q(\mathbf{s}_i^n)}\right) q_j^n(i) x_j^n \log(1-e^{-\theta_{0j}})\right] \\
&- \left\langle Q(\mathbf{s}^n)\right\rangle_{Q(\mathbf{s}^n)}.
\end{aligned}$$

The last term is the entropy of the variational distribution, it does not depend on Θ and can be ignored in further M-step derivations.

For the rest of the paper all expectations are over $Q(\mathbf{s}^n)$—the variational posterior on hidden factors based on previous-step parameters. The simplified notation leaves the dependence on $\mathbf{x}$ and $\mathbf{q}$ implicit, but also expresses the intuition that by replacing the posterior by a variational distribution, we effectively "disconnected" the model.

Since $\log P(\mathbf{x}^n,\mathbf{s}^n|\Theta)$, the term inside expectation, is approximated using the same transformation of $P(\mathbf{x}|\mathbf{s})$ as the posterior distribution over the hidden sources, the $\mathbf{q}$ computed in the E-step can be reused in the M-step. The parameter updates for M-step can be derived straightforwardly by setting

$$\frac{\partial}{\partial\theta_{ij}}\mathcal{F}(D,Q(H)) = 0 \qquad \frac{\partial}{\partial\theta_{0j}}\mathcal{F}(D,Q(H)) = 0.$$

Unfortunately, no closed form solutions for these tasks exist. We update the parameters Θ simultaneously by setting them to the numerical solutions of the above equations and iterate the updates until convergence. The numerical solutions are obtained by bisection search (Figure 3). The parameters are set to random non-zero values in the first EM iteration. We note that the dependencies among parameters are relatively sparse and optimizations typically converge in very few optimization steps. The complete parameter update formulas we derived and use in our procedure are summarized in Figure 2.

Updates of variational parameters $q_j^n(i)$. Iterate until fixpoint:

$$q_j^n(i) \leftarrow \langle s_i^n \rangle_{Q(\mathbf{s}^n)} \frac{q_j^n(i)}{\log(1-e^{-\theta_{0j}})} \left[\log(1-A^n(i,j)) - \frac{\theta_{ij}}{q_j^n(i)} \frac{A^n(i,j)}{1-A^n(i,j)} - \log(1-e^{-\theta_{0j}}) \right]$$

subject to condition $\sum_{i=1}^K q_j^n(i) = 1$ ensured through normalization. $A^n(i,j) = e^{-\theta_{0j} - \frac{\theta_{ij}}{q_j^n(i)}}$.

Updates of θ_{ij}s. Find the root of $\partial \mathcal{F} / \partial \theta_{ij} = 0$ numerically:

$$\sum_{n=1}^N \langle s_i^n \rangle_{Q(\mathbf{s}^n)} \left[-1 + x_j^n \frac{1}{1-A^n(i,j)} \right] = 0$$

Updates of θ_{0j}s. Find the root of $\partial \mathcal{F} / \partial \theta_{0j} = 0$ numerically:

$$\sum_{n=1}^N \left[\sum_{i=1}^K \langle s_i^n \rangle_{Q(\mathbf{s}^n)} q_j^n(i) x_j^n \left(\frac{A^n(i,j)}{1-A^n(i,j)} - \frac{e^{-\theta_{0j}}}{1-e^{-\theta_{0j}}} \right) \right] + \left[-(1-x_j^n) + \sum_{i=1}^K x_j^n q_j^n(i) \frac{e^{-\theta_{0j}}}{1-e^{-\theta_{0j}}} \right] = 0$$

Updates of π_is:

$$\pi_i = \frac{1}{N} \sum_{n=1}^N \langle s_i^n \rangle_{Q(\mathbf{s}^n)}$$

Figure 2: A summary of iterative optimization steps for the variational learning method.

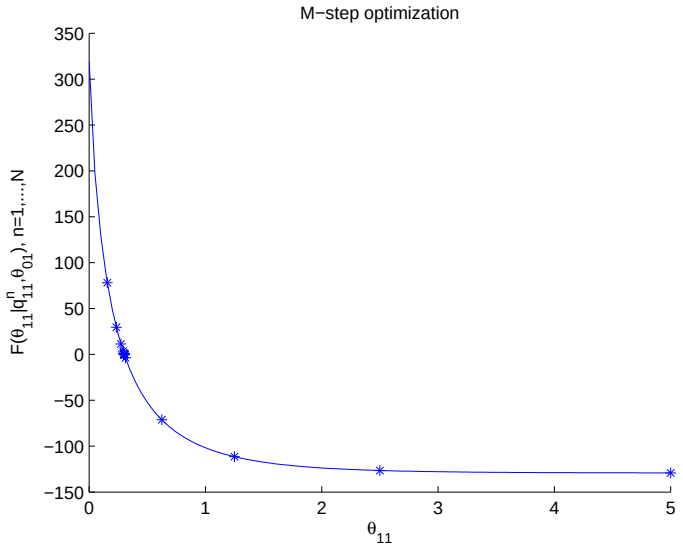
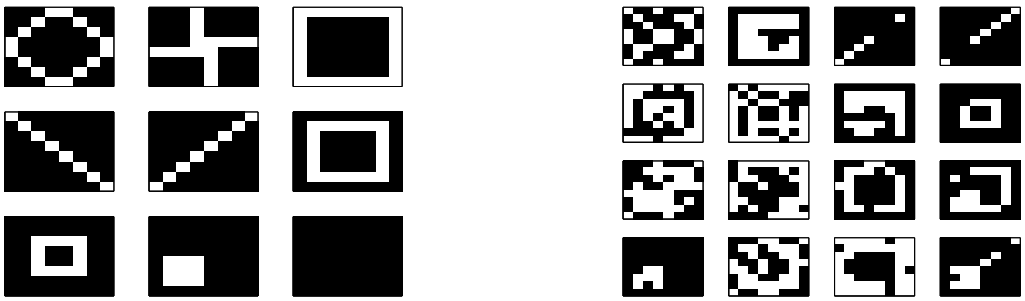


Figure 3: M-step optimization is simply a bisection-search procedure. The curve is the partial derivative of the objective function F w.r.t. θ_{11} plotted as a function of θ_{11}. The little stars on the curve represent iterations of the bisection algorithm. The advantages of the bisection algorithm come from its simplicity: no derivatives that would be costly to compute (we have to iterate through the data to compute F!) and good numerical stability. The search typically converges in few (~ 10) iterations.



(a) (b)

Figure 4: Model reconstruction experiments. **(a)** Image patterns associated with hidden sources used in the image decomposition problem. The ninth (bottom-right) pattern corresponds to the leak. **(b)** Example images generated by the NOCA model with parameters corresponding to patterns in panel (a).

3.3 Simplicity Bias

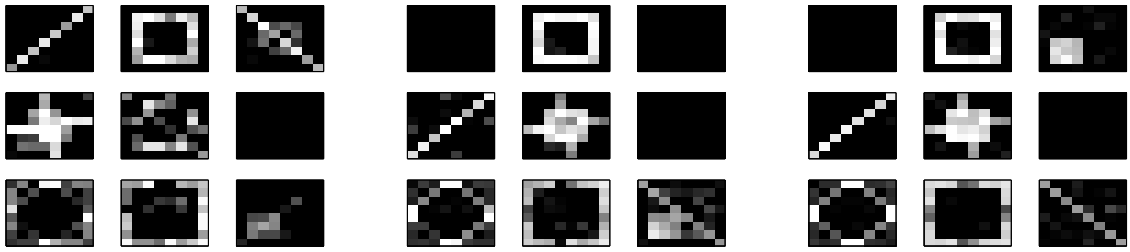
The empirical evaluation of the NOCA model revealed that the model is able to automatically shut off "unused" noisy-or links between sources and observations. This suggests the presence of a term encouraging sparse models in the functional $\mathcal{F}$. Indeed, the term: $-\langle s_i^n \rangle_{Q(\mathbf{s}^n)} \theta_{ij}(1-x_j^n)$ in Equation 7 can be viewed as a regularization-like penalty[2] assigned to large values of θ_{ij} if these are not supported by data. A penalty proportional to θ_{ij} and the posterior of a hidden source is added for each observable x_j^n that is equal to 0. This has an appealing intuitive interpretation: it is unlikely that the observation x_j is 0, if the source is on ($\langle s_i \rangle$ is high) and the link between s_i and x_j is strong $(\theta_{ij} >> 0)$. Consequently, the link in between the source j and observation i is driven to zero if not supported by the presence of positive observations. If all links between a source and observations are driven to zero, the source is effectively disconnected and can be pruned from the network. We demonstrate this effect in the experiments in Section 4.2.

4. Evaluation of NOCA

In this section, we will evaluate NOCA and its variational learning algorithm on a synthetic image data set built using NOCA model. The advantage of using a synthetic data set is that the true model as well as the instantiations of the hidden sources are known.

The image data sets used in the experiments are created by sampling from a NOCA model with 8 hidden sources. Each source is associated with an 8×8 image pattern. The patterns and examples of the convoluted input images are shown in Figure 4, panels (a) and (b).

2. Standard regularization framework involves a data-independent term that penalizes for non-zero parameters. However, here the penalty term depends on data and is a property of the model.



(a) (b) (c)

Figure 5: Examples of models learned from 50, 200 and 1000 samples (panels a through c). The differences among models illustrate the improvement in the model recovery with increasing sample size. Although some source images are identified quite well with as few as 50 samples, the noise in other images is apparent. Models learned from 200 and 1000 samples are visibly improved.

4.1 Model Reconstruction

Our first objective is to assess the ability of the variational algorithm to learn the NOCA model from observational data. In this experiment, we used data sets of size 50–5000 data points that were generated randomly from the model. The data sets were given to the learning algorithm and the learned models were compared to the original model.

Figure 5 visualizes the parameters of three models recovered by the learning algorithm for varied sample sizes. It is apparent that the increase in the number of samples leads to improved models that are closer to the original model. The model learned from 50 samples suffers from high variance caused by the low number of training examples. Nevertheless, it is still able to capture some of the original source patterns. Sample sizes of 200 and 1000 improve the pattern recovery. By learning from 1000 samples, we were able to recover almost all sources used to generate data with a relatively small distortion.

Figure 5 illustrates the dependency of the model quality on the sample size in qualitative terms. To measure this dependency more rigorously we use the training/testing validation framework and a metric based on the joint distribution of observable data. The NOCA model is always learned from a training set. We use training sets of size $50, 100, 200, 500, 1000, 2000, 5000$. The testing set (sample size 2000) is viewed as a sample from the true multivariate distribution. We calculate its log-likelihood with respect to the learned model. A better fit of the model will be reflected in improved log-likelihood of the test sample with respect to this model. Figure 6 shows the log-likelihoods for NOCA models averaged over 50 testing sets. The results demonstrate that an increased size of training sets leads to a better log-likelihood of test data and hence a better approximation of the true distribution.

4.2 Model Selection

In practice, the correct latent dimensionality is rarely known in advance. Model selection is typically addressed within the Bayesian framework. Marginal data likelihood (Cooper and Herskovits, 1992) or its approximations (such as the Laplace approximation) are typically used for this purpose.

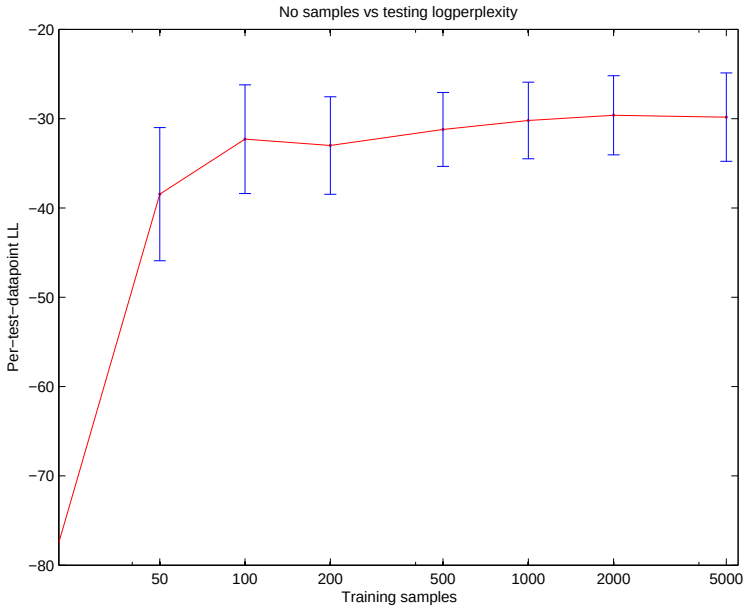


Figure 6: Average log-likelihoods of NOCA models on testing sets. The models are learned from training sets of size 50, 100, 200, 500, 1000, 2000 and 5000. The averages are over 50 trials. One-standard-deviation error bars are shown. The increase in the log-likelihood illustrates the improvement in the model recovery with an increasing sample size.

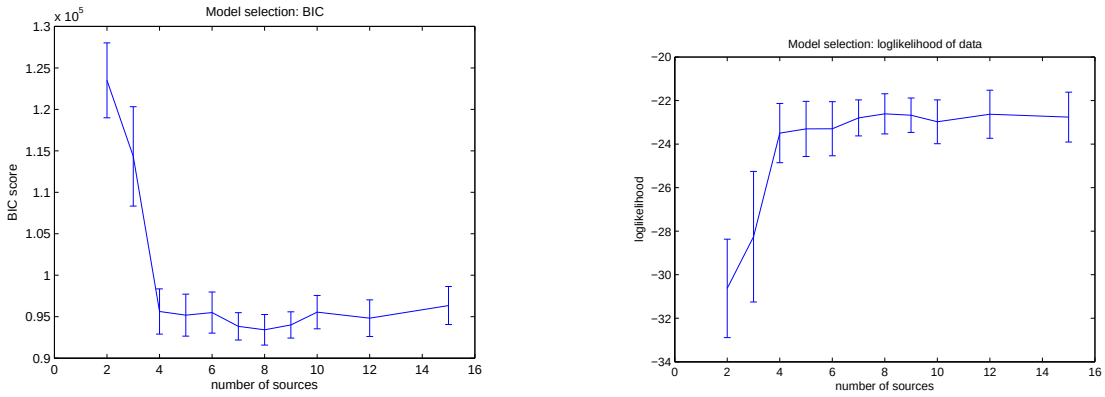
However, in presence of hidden variables it is intractable to compute the marginal likelihood. To address the model selection problem in NOCA we rely on the Bayesian Information Criterion (BIC). The BIC is a large-sample approximation to the integrated likelihood (Schwarz, 1978):

$$BIC(k) = -\ln p(\mathcal{D}|k,\hat{\Theta}_k) + \frac{1}{2}\psi_k \ln N$$

where $\hat{\Theta}_k$ is the ML estimate of NOCA parameters for the model with k hidden sources and ψ_k is the number of free parameters in this model.

Figure 7a shows the results of model selection experiments based on the BIC score. The results are averages of BIC scores obtained by learning the model using 2000 images generated by sampling from NOCA model with 8 hidden sources. In training on this data set, the number of assumed hidden sources varied from 2 to 15. To assure fair comparison, the same training data was used for all models in one train/test run. We see that the optimum BIC score is achieved at 8 sources which corresponds to number of sources in the original model.

The BIC score penalizes models with larger number of parameters. The penalty opposes the increase in the log-likelihood of training data we expect to see in more complex models with a larger number of hidden sources. However, in Section 3.3 we have pointed out the existence of an inherent "regularization" ability of NOCA, that is, its ability to shut down the influence of unnecessary sources once the true dimensionality of the model is reached. In such a case we would expect the log-likelihood of training data to level out for larger than the true number of sources. Figure 7b



(a) (b)

Figure 7: **(a)** The average BIC scores for the models with varied number of sources. **(b)** The average log-likelihood of data for model with varied number of sources. In both cases, the true number of sources K is fixed at 8. Averages are calculated from 20 trials (one-standard-deviation bars are shown). In each trial, the model was learned using 2000 data points. The BIC reaches its optimal value at, and log-likelihood levels at 8 sources, which corresponds to true number of sources.

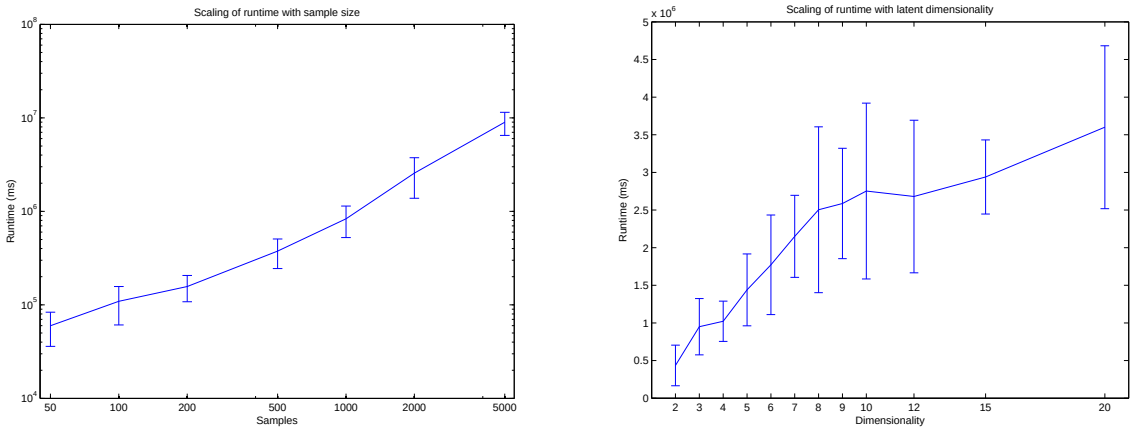
illustrates this point by plotting the log-likelihood of data for models with different number of sources. The setup of the experiment is the same as used in the BIC experiments. The log-likelihood score increases for models with fewer than 8 sources. The log-likelihood for more than 8 sources remains approximately the same. Visual inspection of the learned loading matrices reveals how this happens: many sources are disconnected from the model when the model learns their corresponding loading matrix rows to be identically 0. The models that were initialized with more than 8 sources most often stabilized at 7-8 active (connected) sources. The fact that in some instances the number of sources converged to 7 can be explained the ability of the leak factor to effectively model an additional source.

4.3 Running-time Analysis

Precise time-complexity analysis of the NOCA learning algorithm is impossible since both the expectation and maximization steps involve iterative procedures whose convergence properties are not well understood. Moreover, these are embedded in the EM loop itself and while eventual convergence is assured, its rate is not. Therefore we evaluate the time complexity empirically, with respect to N, the size of the training set and K, the number of latent sources.[3] We have observed no dependence between training set size, the assumed number of latent sources and the number of EM iterations performed in experiments.

The running time of the learning algorithm for different training set sizes is shown in Figure 8(a). A nearly straight line indicates that the complexity grows polynomially with the number of samples.

3. It follows from the form of the update equations that the algorithm is linear in D, the number of observable dimensions.



(a) (b)

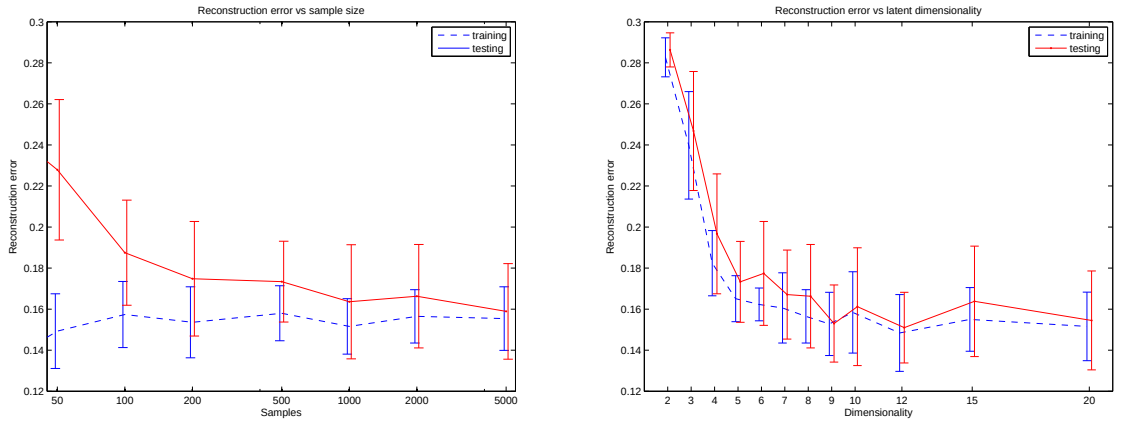
Figure 8: **(a)** Runtimes of NOCA as they scale with increasing size of the training set. K is fixed at 8. **(b)** Scale-up with the number of assumed latent sources, the data set size is fixed at 2000.

In fact, we have observed that the time complexity scales approximately linearly with the number of samples in the training set. The analysis of running times for different number of sources in Figure 8(b) shows that these scale roughly linearly with the number of assumed latent sources. This gives empirical support for the efficiency of the variational EM approximation as compared to the exact EM algorithm.

4.4 Dimensionality Reduction and Data Compression with NOCA

Latent variable models are inherently well suited for dimensionality reduction. Lossy compression of the data by the NOCA model can be achieved as follows. Given the learned NOCA model and an observed test-set image, we compute the posterior of each hidden source and pick the value with the higher posterior probability. The values of the hidden sources act as a low-dimensional representation of the test data. The high-dimensional data can then be recovered by sampling the observables given the stored values of sources and compared to the original test-set.

Figure 9(a) illustrates the data reconstruction error of the NOCA model learned for different sample sizes. The data reconstruction error is defined as the proportion of feature values in which the original data set differs from the reconstructed data. We measure data reconstruction error on both the training and the testing data. The training set is the data used to learned the model, the testing set is an additional sample from the model. The data reconstruction error for the training set is smaller for very small sample sizes and stabilizes for sample sizes over 200. This can be explained by "overfitting"—the use of free model parameters for memorization of training data—for small sample sizes, and saturation of the model to its stochastic limit for larger sample sizes. The data reconstruction error for test sets behaves inversely—it is worse from smaller training sets and stabilizes for larger training sets as the learned model improves.



(a) (b)

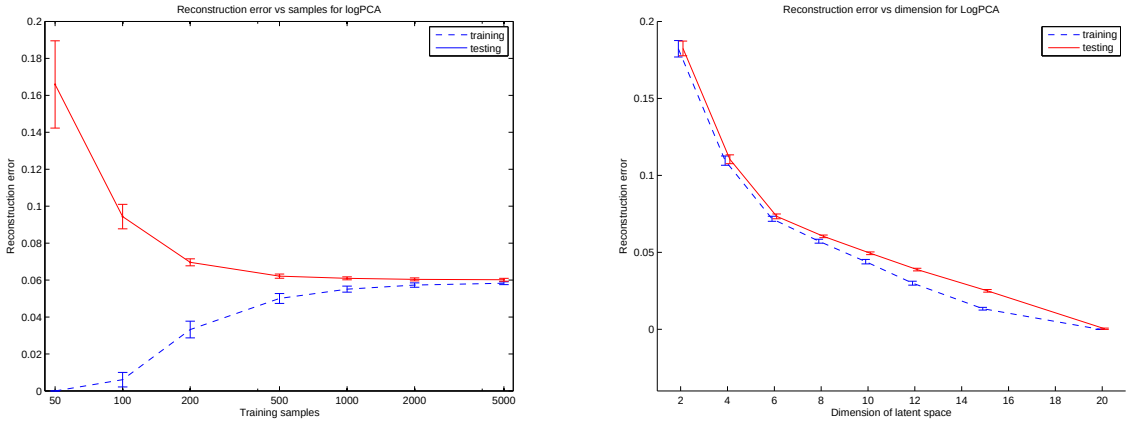
Figure 9: **(a)** Average data reconstruction errors obtained for varied training sample sizes. **(b)** Average data reconstruction error plotted against the number of assumed latent sources. All values are averaged over 50 runs.

Figure 9(b) shows the influence of the number of hidden sources on the data reconstruction error. The data reconstruction error goes down with increasing K and flattens out as the learned models use no more than the true number of sources (8), thanks to the effect described in Section 3.3.

A related dimensionality-reduction model tailored to binary data is offered by logistic PCA (Schein et al., 2003). In this model, each component x_j^n of a data point $\mathbf{x}^n$ is assumed to be sampled from a Bernoulli distribution whose parameter θ_j^n is determined by a logistic unit from the factors $\mathbf{v}$, latent coordinates $\mathbf{u}$ and the bias term Δ_j: $\theta_j^n = \sigma(\mathrm{v}_j.\mathrm{u}_j^n + \Delta_j)$. The crucial difference between NOCA and logistic PCA is that the latent space in NOCA is discrete while in logistic PCA it is continuous. As a result, logistic PCA uses a many-bit floating point representation to capture many one-bit feature values. Figure 10 illustrates data reconstruction errors for the same experiments as performed for NOCA in Figure 9. The results demonstrate better data reconstruction performance of the logistic PCA model. This is expected since the complexity of NOCA's latent space is much smaller (*finite* as opposed to continuous). The differences in performance demonstrate the tradeoff in between the complexity of the representation of the latent space and its accuracy. In particular, NOCA uses 8 bits to represent each data point in the latent space while the logistic PCA uses a vector of 8 floating point values per data point.

5. An Application of NOCA to Citation Analysis

The analysis of NOCA on image data sets confirms it can discover, fully unsupervised, the structure of the hidden components reasonably well. But does the method apply to the real world? Do its assumptions really fit the data it was designed for? To assess this aspect of NOCA we test it on a citation analysis problem. We first discuss the data set and proceed to report the results of three evaluation strategies: (1) evaluation by a human judge, (2) a cosine-distance based metric and (3) perplexity of a testing set.



(a) (b)

Figure 10: Reconstruction errors achieved by the logistic PCA, a) as they vary with training set size (fixed size testset), and b) as they vary with the latent dimension of the model.

5.1 Citation Data

We acquired a data set of approximately 17.000 documents from the CiteSeer online service. These are the HTML documents that place a scientific article within the lattice of citations; not to be mistaken for the actual text of the article. We chose forty authors active in these publication areas: *Introductions and tutorials*, *Markov chain Monte Carlo*, *Variational methods*, *Loopy belief propagation* and *Kernels and support vector machines*. Naturally, there are overlaps; for example, a publication discussing approximate inference in Bayesian networks is likely to mention both loopy belief propagation and MCMC techniques. This overlapping structure renders the task quite non-trivial. In addition, it makes it difficult to come up with an unambiguous "gold standard" clustering.

We selected all papers in the data set citing any of the selected authors. The data set was preprocessed into a binary matrix M_{ij}, where the element (i, j) is 1 if document i cites a paper authored by author j and 0 otherwise. Zero rows, that is documents that cite none of the authors, are discarded. There were 6592 non-zero rows in the matrix.

5.2 NOCA Formulation of Citation Analysis

The citation data set consists of N documents, each of which cites a number of authors. The individual authors publish on one or more topics. Our conjecture is that certain citation patterns are indicative of paper topics. We wish to discover these topics and their associated authors, in a fully unsupervised manner.

To analyze the data with NOCA, we assume that the topics are represented with the hidden binary variables $s_1, \ldots s_K \in \{0, 1\}$. Intuitively, $s_i = 1$ in the unobserved event that the document discusses topic i. The citation features correspond to the observed variables $x_1, \ldots x_D$. The n-th document in the corpus is thus represented by a D-dimensional binary vector $\mathbf{x}^n$. The event that document n cites author j is captured by observing $x_j^n = 1$. The "affinity" of author j and topic i

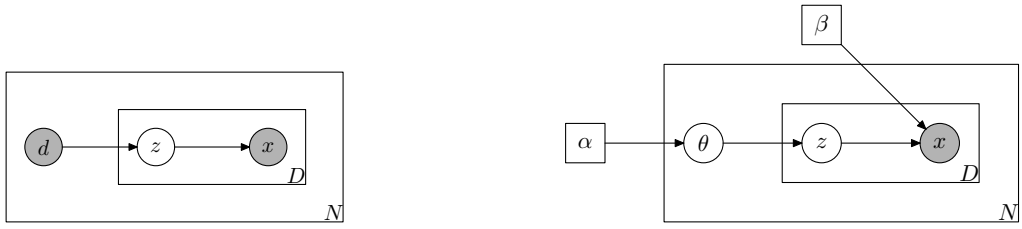


Figure 11: PLSA (a) and LDA (b) graphical models.

is expressed by the weights p_{ij} which parameterize the noisy-or CPD's of the bottom layer nodes. This defines a generative probabilistic model at the document feature level:

- For all $i = 1,\ldots,K$, sample s_i from $Bernoulli(\pi_i)$.
- For all $j = 1,\ldots,D$, sample x_j from the noisy-or distribution $p(x_j|\mathbf{s})$.

5.3 Mixture Models

Latent variable models have demonstrated good results in text and document analysis. Most of these are mixture models that view a document as a mixture of hidden topic factors. The topic factors are identified with distributions over words. The key assumption of a mixture model is that the occurrence of a specific word is determined by a *single* mixture component. These models share the bag-of-words view of a document and provide a probabilistic model for each word occurrence. NOCA offers a different view of a document: A document is a combination of *non-competing* topics and each word is determined by a *combination* of topics. NOCA does not define a model for generation of each single word, which makes it less suitable for applications such as text modeling, but it fits more naturally the type of data encountered in link analysis.

In the following, we briefly review two mixture models applied frequently in text modeling: PLSA and LDA. These state-of-the-art text models have also been recast for link analysis purposes (Cohn and Hofmann, 2001; Cohn and Chang, 2000).

Probabilistic Latent Semantic Analysis (PLSA) Hofmann (1999a), whose graphical model is shown in Figure 11(a), assumes that each document is represented by a convex combination (a mixture) of topics and that the features of the document are generated by the following process:

1. pick a document d according to Multinomial $P(d)$ (defined by a dummy indexing of the documents in the data set),
2. sample a topic z according to Multinomial $P(z|d)$,
3. generate a feature from $P(x|z)$.

The joint probability $P(d,x)$ factorizes as $P(d)\sum_z P(z|d)P(x|z)$. Since the topic variable z is unknown, the algorithm for learning PLSA derives from the EM framework.

Latent Dirichlet Allocation (LDA) Blei et al. (2003) adds Bayesian hyperparameters to the PLSA model so that the mixture proportions themselves are a Dirichlet-distributed random variate (Figure 11). The following process is assumed to generate the documents:

1. sample a parameter θ from the exchangeable Dirichlet distribution $Dir(\alpha)$,
2. sample a topic from Multinomial $P(z|\theta)$,
3. generate a feature from $P(x|z,\beta)$.

Both the parameter θ and the topic variable z are unobserved. The addition of the new hidden parameters makes the exact inference for LDA intractable. To alleviate this problem Blei et al. derive a variational inference algorithm which in turn allows them to develop an efficient variational EM learning procedure.

The conceptual difference between NOCA on the one hand and PLSA or LDA on the other is that NOCA views a document as a *set of features*, while the mixture methods regard it as a *bag of words*. More importantly, NOCA makes a different assumption about the nature of the topic factors. PLSA (Figure 11, left) and LDA (Figure 11, right) view the topic factors as points in the vector space spanned by the orthogonal basis which is the vocabulary. Moreover, all these points belong to a subspace of the $(D-1)$-dimensional word simplex since they correspond to normalized distributions. NOCA treats the topic as a separate type of entities that live in their own K-dimensional space which projects non-linearly into the vocabulary space. As opposed to PLSA, where one *aspect* is assumed to be responsible for the generation of a word, in NOCA, potentially all of the topic factors contribute to the generation of a single word feature. Additionally, the added freedom of the leak parameter allows NOCA to "put aside" the documents where no structure seems to stand out. These do not have to be accounted for in the output components. Clearer clustering is the outcome that we would expect from this organization.

5.4 Experiments

The evaluation of topic discovery in any of the frameworks relies on the identification of largest elements of output vectors or matrices. Since the semantics of the numeric values differs in the respective approaches, the only consistent way of comparing the outputs is by listing the most prominent elements of each of the identified clusters. We achieve this goal for different models as follows:

- Logistic PCA is parameterized by the loading matrix V and the constant bias vector Δ. We interpret rows of V as the component vectors and list the authors corresponding to the largest elements in each component vector.
- PLSA parameterization is not as in Figure 11(a), but instead the model is equivalently parameterized with $P(z)$, $P(d|z)$ and $P(x|z)$ (Hofmann, 1999a). We list the authors x with the highest $P(x|z)$ for each aspect z. Also reported is $P(z)$, to help assess the relative representation of the aspects.
- LDA provides the matrix β and the Dirichlet hyperparameter α. The reported components are the rows of β; the authors corresponding to the highest values in each row of β are listed. The components that LDA recovers are very stable, which is characteristic of the Bayesian

	Community				
Method	intro	MCMC	var'l	LBP	Kernel
LogPCA	40.0	42.5	15.0	10.0	67.5
PLSA	67.5	57.5	50.0	32.5	75.0
LDA	80.0	95.0	62.5	5.0	87.5
NOCA	85.0	15.0	92.5	82.5	75.0

Table 1: Success rates in recovering subcommunities in the citation data. The numbers are percentages averaged over 20 different random initializations.

approach taken in developing the model. Therefore we report results from 20 runs with different α (the initial exchangeable Dirichlet prior) instead, starting from $\alpha = 0.01$ and ending at $\alpha = 10$.

- For NOCA, the output consists of the cluster priors π_i, the loading matrix $\mathbf{p}$ and the "bias vector" $\mathbf{p}_0$. The authors listed under each component are those who received the highest weight in $\mathbf{p}_i$, the i-th row of the loading matrix $\mathbf{p}$. Again we report the priors π_i to compare the relative size of the clusters. Note that the priors need not sum to one, since each of them corresponds to a separate random variable.

5.4.1 QUALITATIVE EVALUATION

We ran all of the algorithms 20 times with different random initializations and visually judged the results from displays such as that in Figure 13. If a particular topic factor appeared and was determined to be of good "cluster purity", we assigned a score of 1 to the combination of community and analysis technique. If the cluster was identifiable with a community, but judged to be of mediocre purity, the score assigned was 1/2. Otherwise, the score assigned was 0. Whenever the community was captured in more than one factor, only one was counted. The maximum score is 20 as there were 20 experimental runs. The entries in Table 1 are the respective percentages.

The logistic PCA does not appear to be well suited for this task and is outperformed by the other methods. PLSA finds on average 2 communities in each run. LDA discovers the MCMC topic consistently, but fails to discover the LBP community. NOCA exhibits the opposite behavior: it reliably discovers LBP but fails to find the MCMC community most of the time. The SVM/kernel group and the variational methods community is consistently discovered by both NOCA and LDA, as well as the authors of widely cited overview and tutorial articles.

The difference observed for the LBP and MCMC communities is striking and should be explained by pointing out the characteristics of the respective communities. The LDA model is able to recover communities that have established their "market share" and have high enough prior probability that they are able to compete with the other groups for the direction that the topic simplex takes in the "vocabulary" space. LDA thus has a difficult time finding small, emerging areas. On the other hand, these nascent communities tend to be highly coherent, with a few pioneers that are very likely to be cited for their seminal papers. Such structure favors the NOCA model, which has a tendency to pick out tightly woven patterns and leave the more diffuse domains to be picked up

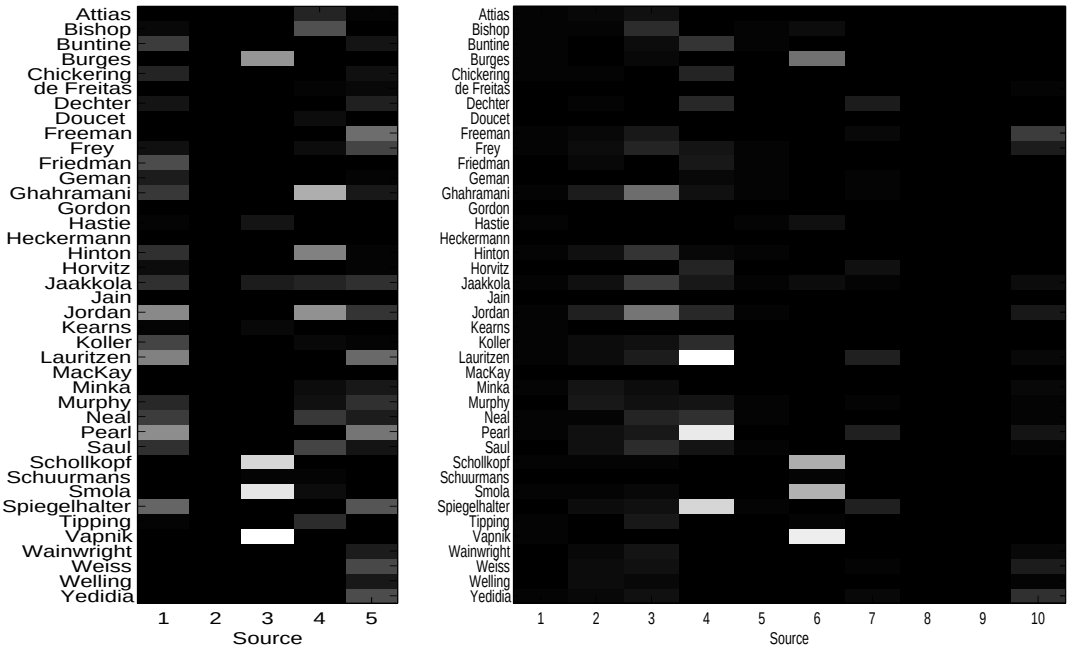


Figure 12: A result of noisy-or component analysis on the citation data set. The columns visualize the parameters of the noisy-or loading matrix after they are rescaled by the prior of the source. Black fields correspond to 0s in the loading matrix, while white ones correspond to 1s.

(a) With 5 components. The following components are discernible:

- The authors dominating the first component are: J. Pearl, M. Jordan, S. Lauritzen and D. Spiegelhalter. Weaker ties are to W. Buntine, N. Friedman and D. Koller. This component contains many respected authors of basic references and tutorials on Bayesian belief networks.
- The second source was shut down in this run.
- C. Burges, B. Schölkopf, A. Smola and V. Vapnik form the core of the third component. Without any doubt, this component represents the kernel and SVM research community.
- The authors prominent in the fourth factor are Z. Ghahramani, M. Jordan, G. Hinton, R. Neal, L. Saul, C. Bishop and M. Tipping. This source captures the variational approximation community.
- The last component consists of the following authors: B. Frey, W. Freeman, K. Murphy, S. Lauritzen, J. Pearl, Y. Weiss and J. Yedidia. All authors published extensively on loopy belief propagation, using J. Pearl's BP algorithm. The presence of an outlier in this set, S. Lauritzen, can be attributed to the fact that he is among the most frequently cited authors in the general context of Bayesian networks. We can conclude that our algorithm found the LBP community.

(b) A run with 10 components illustrates the regularization behavior. Four out of ten sources were completely or almost completely shut off.

Comp. 1	Comp. 2	Comp. 3	Comp. 4	Comp. 5
Vapnik	Doucet	Bishop	Jain	Pearl
Doucet	de Freitas	Vapnik	Spiegelhalter	Smola
Freeman	Ghahramani	Hastie	Friedman	Lauritzen
Kearns	Friedman	Jain	Gordon	Friedman
Smola	Murphy	Burges	Bishop	Freeman
Murphy	Hastie	Schollkopf	Geman	Horvitz
de Freitas	Jordan	Smola	Neal	Schollkopf
?	**?**	**kernel**	**?**	**?**

(a) logPCA

Aspect 1	Aspect 2	Aspect 3	Aspect 4	Aspect 5
0.1284	0.1640	0.3513	0.2321	0.1241
Bishop	Friedman	Jordan	Vapnik	Geman
Jain	Neal	Hinton	Smola	Doucet
Kearns	Hastie	Spiegelhalter	Pearl	Gordon
Tipping	Murphy	Lauritzen	Schollkopf	de Freitas
Hinton	Koller	Pearl	Burges	Koller
Schuurmans	Jaakkola	Freeman	Horvitz	Frey
Saul	Buntine	Weiss	Koller	Murphy
?	**?**	**?**	**kernel**	**MCMC**

(b) PLSA

Comp. 1	Comp. 2	Comp. 3	Comp. 4	Comp. 5
0.0022	0.0912	0.0858	0.0277	0.0102
Minka	Vapnik	Lauritzen	Jordan	Freeman
Jordan	Smola	Pearl	Ghahramani	Yedidia
Jaakkola	Schollkopf	Spiegelhalter	Hinton	Weiss
Yedidia	Burges	Jordan	Bishop	Frey
Ghahramani	Hastie	Buntine	Saul	Murphy
Freeman	Jaakkola	Koller	Jaakkola	Welling
Frey	Bishop	Dechter	Attias	Pearl
?	**kernel**	**intro**	**variat'l**	**LBP**

(c) NOCA

Comp. 1	Comp. 2	Comp. 3	Comp. 4	Comp. 5
$\alpha = 1$				
Vapnik	Jordan	Geman	Friedman	Pearl
Smola	Hinton	Doucet	Koller	Lauritzen
Schollkopf	Neal	de Freitas	Hastie	Jain
Bishop	Ghahramani	Murphy	Kearns	Spiegelhalter
Burges	Weiss	Gordon	Buntine	Dechter
Tipping	Jaakkola	Koller	Chickering	Freeman
Jaakkola	Horvitz	Welling	Schuurmans	Frey
kernel	**variat'l**	**MCMC**	**intro**	**intro**

(d) LDA

Figure 13: Typical outputs from the link analysis algorithms:
a) Logistic PCA.
b) Probabilistic latent semantic analysis. Also reported is the prior of each aspect $P(z = i)$.
c) Noisy-or component analysis. The prior on a source $P(s_i)$ is also shown.
d) Latent Dirichlet allocation with $\alpha = 1$.
Below each component, our evaluation of whether the component represents any of the the publication communities.

by the leak factor. Thus the broader MCMC community eluded the noisy-or analyzer, while it was reliably captured by LDA; and the NOCA brought to light the LBP community.

In summary, NOCA discovers on average as many clusters as LDA, but the clusters are of different nature. If one wishes to gain insight into this type of data, we advocate that both methods be used, as they discover distinct kinds of patterns.

5.4.2 THE COSINE METRIC

While we took great care to assure objective evaluation, the above approach is nevertheless open to criticism on the grounds of "subjectivity." We would like our recovered components to align with a "gold standard," a set of vectors defined by a human *before* he or she sees the result of the clustering algorithm. Therefore we defined 0-1 vectors corresponding to the established communities as we perceive them. For example, the vector corresponding to LBP community has 1s at positions corresponding to names such as Freeman, Frey, Yedidia, etc. and 0s elsewhere.

A standard distance metric for vectors is their *cosine distance*. The similarity of two vectors $\mathbf{x}, \mathbf{y}$ is the cosine of their angle: $\cos\alpha(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{|\mathbf{x}||\mathbf{y}|}$. With the cosine metric, one can evaluate the similarity of two vectors. However, *how do we quantitatively evaluate a component set X as a whole?* Recovered components need to be matched to the original components as they can be permuted without affecting the likelihood. To obtain a one-to-one match, each original component is paired with exactly one found by NOCA, so that the weighted sum of cosine distances is minimized. The weights u_i are defined so that they are proportional to the prior probabilities of the latent components and form a convex combination (sum to 1). The computation can be described by the formula

$$w_{\cos}(X,Y) = \min_{\pi,\rho} \sum_{i=1}^{K} u_{\pi(i)} \cos\alpha(\mathbf{x}_{\pi(i)}, \mathbf{y}_{\rho(i)}),$$

where π and ρ are permutations of the sets X and Y, respectively, and the minimization ranges over all possible permutations. Note that although this formula suggests evaluating exponentially many permutations, it effectively calls for finding a maximal-weight matching in a bipartite graph and can be computed efficiently. The resulting weighted cosine similarities are shown and commented in Figure 14.

5.4.3 PERPLEXITY

While the cosine scoring metric provides useful insights, using a standard probabilistic measure of model quality is in order to gauge how well the model estimates the joint density of the observable data. To assess this aspect of model recovery we rely on the cross-entropy of the "true" distribution and the distribution that the model entails. The testing set is viewed as a sample from the true multivariate distribution t and the *cross entropy* with the model distribution m is defined by $H(t,m) = -\sum_{\{\mathbf{x}\}} t(\mathbf{x}) \log m(\mathbf{x})$. Since the data points in the test set are by assumption independent and identically distributed, the cross entropy is approximately the average unconditional log-probability of data points in the test set (Cover and Thomas, 1991). *Perplexity* of the model m is defined as the quantity $2^{H(t,m)}$ and can be intuitively interpreted as the amount of information needed to predict the next data point. In short, the lower the cross-entropy is, the more precisely the model has learned the distribution of observables from the training set.

In the perplexity evaluation of NOCA and LDA, we use the tractable lower bounds on the document probabilities $P(\mathbf{x})$ (Equation 6 in this paper and Equation 13 in Blei et al. (2003)). The PLSA and logistic PCA models cannot be evaluated under the perplexity framework since they do not define a probability distribution on the test set. PLSA does define a distribution on the training set and the fold-in heuristic can be used (Hofmann, 1999b) so that it defines one on the test set. However, this heuristic gives PLSA an optical advantage over other models, as it allows it to refit the mixing proportions. As a baseline model, we will use a simple mixture of unigrams model. As NOCA provides no word-level model, but only a document-level probability model, we

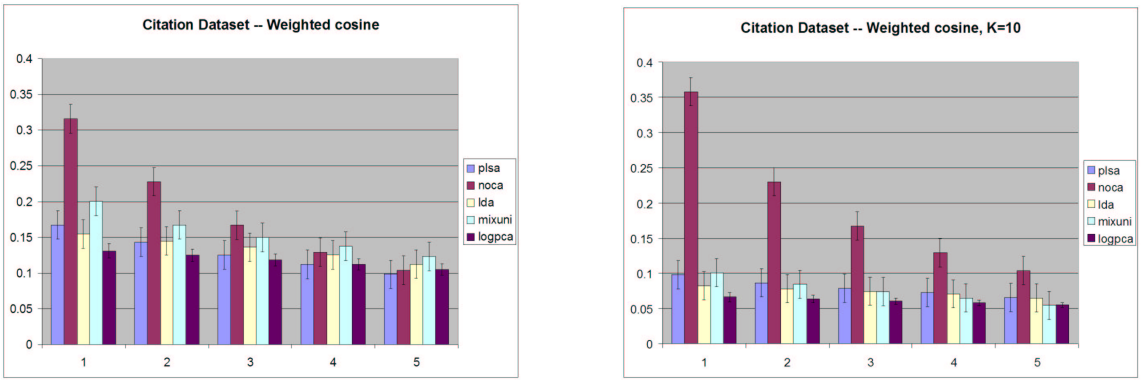


Figure 14: Weighted cosine similarities. On the horizontal axis is L, the number of components matched. The vertical axis shows the weighted cosine similarity. The left panel shows NOCA doing a superior job identifying the first few components, but it is soon overtaken by the mixture-based methods. The methods in the left panel operated with latent dimensionality 5, equal to the number of human-judged clusters. On the right, the picture changes when the latent dimensionality is increased to 10. While the performance of mixture-based methods deteriorates, NOCA's performance improves. This illustrates the difference in the assumptions about the data-generating process. The picture suggests that the more sophisticated methods do a better job in comparison with the baseline (a simple mixture of unigrams model) when the asumed latent dimensionality slightly exceeds the true number of clusters in the data.

	Cross-entropy			
K	NOCA	LDA	PLSA	MixUnigrams
5	$< 6.5 \pm 5.2$	$< 9.0 \pm 7.8$	6.1 ± 6.9	22.9 ± 31.3
10	$< 6.5 \pm 5.2$	$< 8.4 \pm 7.5$	4.9 ± 6.4	32.5 ± 46.0

Table 2: Cross-entropies between the model distribution and the empirical distribution induced by the test set. These numbers were obtained as mean and standard deviation on 20 train/test splits.

must compare all models in terms of document perplexity, instead of the standard approach that works at the level of words. Inspecting Table 2, we observe that the bound on perplexity of NOCA is significantly lower than that of LDA. PLSA shows a cross-entropy virtually on the level with NOCA, or slightly better. The cross-entropy of the baseline mixture-of-unigrams model is high, which is attributable to the data sparsity issue. Importantly, note that the values shown for LDA and NOCA are *lower bounds*, while the PLSA and MixUnigrams are exact.

6. Summary and Conclusions

We have presented NOCA: a new latent-variable component analysis framework for high-dimensional binary data. To learn the NOCA model we have devised and presented an EM-based variational algorithm that overcomes the complexity limitation of exact learning methods. The proposed algorithm makes no assumption about the structure of the underlying noisy-or network, the structure is fully recovered during the learning process.

In addition to the component analysis task and related structure discovery problems, NOCA can be also used as a dimensionality reduction (data compression) tool, as well as a probabilistic model of high-dimensional binary data. We have tested these aspects of the model on a synthetic image decomposition problem and on a citation analysis problem of CiteSeer documents. The model and the algorithm showed favorable scale-up behavior and a very good model recovery and error reconstruction performance.

The task of community discovery has a natural formulation as a NOCA learning problem. A data set of scientific paper citations in the field of machine learning was analyzed using the setup. The results, under several metrics, indicate that our algorithm performs on par with the current state-of-the-art mixture methods, but due to different data-generating assumptions it tends to expose different data structure. Such behavior is valuable as it enriches our insight into the intrinsic composition of the data set.

Acknowledgments

We would like to thank the reviewers for their helpful comments and suggestions for improvements of the paper. This research was supported in part by the National Science Foundation grants ANI-0325353 and CMS-0416754 and by the University of Pittsburgh award CDRF-36851.

References

Hagai Attias. Independent Factor Analysis. *Neural Computation*, 11(4):803–851, 1999. URL citeseer.nj.nec.com/attias99independent.html.

David Bartholomew and Martin Knott. *Latent Variable Models and Factor Analysis*, volume 7 of *Kendall's Library of Statistics*. Oxford University Press, 1999.

Christopher Bishop. Latent variable models. In Michael Jordan, editor, *Learning in Graphical Models*, pages 371–403. MIT Press, 1999a.

Christopher Bishop. Variational principal components. In *Proceedings of* 9^{th} *International Conference on Artificial Neural Networks*, volume 1, pages 509–514, 1999b.

David Blei, Andrew Ng, and Michael Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Jan 2003. URL http://www.cs.berkeley.edu/ blei/papers/blei03a.ps.gz.

Wray Buntine. Variational extensions to EM and multinomial PCA. In *Proceedings of the* 13^{th} *European Conference on Machine Learning*, 2002. URL citeseer.nj.nec.com/buntine02variational.html.

David Cohn and Huan Chang. Learning to probabilistically identify authoritative documents. In *Proceedings of the* 17^{th} *International Conference on Machine Learning*, pages 167–174. Morgan Kaufmann, San Francisco, CA, 2000. URL citeseer.ist.psu.edu/cohn00learning.html.

David Cohn and Thomas Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *Neural Information Processing Systems 13*, 2001. URL citeseer.ist.psu.edu/cohn01missing.html.

Gregory Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

Thomas Cover and Joy Thomas. *Elements of Information Theory*. John Wiley & sons, 1991.

Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood for incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1–38, 1977.

Francisco Diez and Severino Gallan. Efficient computation for the Noisy-Max. *International Journal of Intelligent Systems*, 2003.

Zoubin Ghahramani and Michael Jordan. Factorial hidden Markov models. In David Touretzky, Michael Mozer, and Michael Hasselmo, editors, *Proceedings of Advances in Neural Information Processing Systems*, volume 8, pages 472–478. MIT Press, 1995. ISBN 0262201070. URL citeseer.nj.nec.com/article/ghahramani97factorial.html.

Zoubin Ghahramani and Michael Jordan. Factorial hidden Markov models. *Machine Learning*, 29: 245–273, 1997.

David Heckerman. Causal independence for knowledge acquisition and inference. In *Proceedings of 9th Conference on Uncertainty in AI UAI'93*, San Francisco, CA, 1993. Morgan Kaufmann Publishers.

Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence*, Stockholm, 1999a. URL citeseer.nj.nec.com/hofmann99probabilistic.html.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, August 1999b. URL citeseer.nj.nec.com/article/hofmann99probabilistic.html.

Tommi Jaakkola and Michael Jordan. Variational probabilistic inference and the QMR-DT network. *Journal of Artificial Intelligence Research*, 10:291–322, 1999. URL citeseer.nj.nec.com/article/jaakkola99variational.html.

Tommi Jaakkola, Lawrence Saul, and Michael Jordan. Fast learning by bounding likelihoods in sigmoid type belief networks. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 528–534. The MIT Press, 1996. URL citeseer.ist.psu.edu/jaakkola96fast.html.

Ian Jolliffe. *Principal Component Analysis*. Springer, 1986.

Michael Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Michael Kearns and Yishay Mansour. Exact inference of hidden structure from sample data in noisy-or networks. In *Proceedings of the* 14^{th} *Conference on Uncertainty in Artificial Intelligence*, pages 304–310, 1998. URL citeseer.nj.nec.com/383491.html.

Xinghua Lu, Milos Hauskrecht, and Roger Day. Modeling cellular processes with variational Bayesian cooperative vector quantizer. In *Proceedings of Pacific Symposium on Biocomputing*, 2004.

David MacKay. Probable networks and plausible predictions - a review of practical Baysian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995.

James Miskin. *Ensemble Learning for Independent Component Analysis*. PhD thesis, Selwyn College, University of Cambridge, 2000.

David Ross and Richard Zemel. Multiple cause vector quantization. In *Proceedings of Advances in Neural Information Processing Systems 16*, 2002. URL citeseer.ist.psu.edu/ross02multiple.html.

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

Andrew Schein, Lawrence Saul, and Lyle Ungar. A generalized linear model for principal component analysis of binary data. In *Proceedings of the* 9^{th} *International Workshop on Artificial Intelligence and Statistics*, 2003. URL citeseer.nj.nec.com/546431.html.

Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

Michael Shwe, Blackford Middleton, David Heckerman, Max Henrion, Eric Horvitz, Harold Lehmann, and Gregory Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: Part I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255, 1991.

Tomáš Šingliar and Miloš Hauskrecht. Variational learning for noisy-or component analysis. In *Proceedings of the SIAM International Conference on Data Mining*, pages 370–379, 2005.

Michael Tipping and Christopher Bishop. Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, September 1997. URL citeseer.nj.nec.com/article/tipping97probabilistic.html.

Jiří Vomlel. Noisy-or classifier. In *Proceedings of the* 6^{th} *Workshop on Uncertainty Processing*, pages 291–302, 2003. URL http://lisp.vse.cz/wupes2003/.

Journal of Machine Learning Research 7 (2006) 2215-2236 Submitted 12/05; Revised 8/06; Published 10/06

Learning a Hidden Hypergraph

Dana Angluin ANGLUIN@CS.YALE.EDU
Department of Computer Science
Yale University
P.O. Box 208285
New Haven, CT 06520, USA

Jiang Chen CRIVER@CS.COLUMBIA.EDU
Center for Computational Learning Systems
Columbia University
475 Riverside Drive
850 Interchurch MC 7717
New York, NY 10115, USA

Editor: Manfred Warmuth

Abstract

We consider the problem of learning a hypergraph using edge-detecting queries. In this model, the learner may query whether a set of vertices induces an edge of the hidden hypergraph or not. We show that an r-uniform hypergraph with m edges and n vertices is learnable with $O(2^{4r}m \cdot poly(r,\log n))$ queries with high probability. The queries can be made in $O(\min(2^r(\log m+r)^2, (\log m+r)^3))$ rounds. We also give an algorithm that learns an almost uniform hypergraph of dimension r using $O(2^{O((1+\frac{\Delta}{2})r)} \cdot m^{1+\frac{\Delta}{2}} \cdot poly(\log n))$ queries with high probability, where Δ is the difference between the maximum and the minimum edge sizes. This upper bound matches our lower bound of $\Omega((\frac{m}{1+\frac{\Delta}{2}})^{1+\frac{\Delta}{2}})$ for this class of hypergraphs in terms of dependence on m. The queries can also be made in $O((1+\Delta)\cdot\min(2^r(\log m+r)^2,(\log m+r)^3))$ rounds.

Keywords: query learning, hypergraph, multiple round algorithm, sampling, chemical reaction network

1. Introduction

A hypergraph $H=(V,E)$ is given by a set of vertices V and a set of edges E, which is a subset of the power set of V ($E \subseteq 2^V$). The dimension of a hypergraph H is the cardinality of the largest set in E. H is said to be r-uniform if E contains only sets of size r. In this paper, we are interested in learning a hidden hypergraph using *edge-detecting* queries of the following form

$$Q_H(S): \textit{does } S \textit{ include at least one edge of } H?$$

where $S \subseteq V$. The query $Q_H(S)$ is answered 1 or 0, indicating whether S contains all vertices of at least one edge of H or not. We abbreviate $Q_H(S)$ to $Q(S)$ whenever the choice of H is clear from the context. This type of query may be motivated by the following scenarios. We are given a set of chemicals, in which some groups of chemicals react and others don't. When multiple chemicals are

combined in one test tube, a reaction is detectable if and only if at least one group of chemicals in the tube react.

Considerable effort, for example, Grebinski and Kucherov (1998), Beigel et al. (2001), Alon et al. (2004), Angluin and Chen (2004), and Alon and Asodi (2005), has been devoted to the case when the underlying reaction network is a graph, that is, chemicals react in pairs. Among them, Grebinski and Kucherov (1998), Beigel et al. (2001) and Alon et al. (2004) study the case when the underlying networks are Hamiltonian cycles or matchings, which have specific applications to genome sequencing. In this application, DNA sequences are aligned according to the reactions that involve the two ends of pairs of DNA sequences in certain experimental settings. The reaction graph can be characterized as either a Hamiltonian cycle or path (if you consider each DNA sequence as a vertex) or a matching (if you consider each end of a DNA sequence as a vertex). Implementations of some of these algorithms are in practical use. Grebinski and Kucherov (2000) also study a somewhat different and interesting query model, which they call the *additive model*, where instead of giving a 1 or 0 answer, a query tells you the total number of edges contained in a certain vertex set.

Angluin and Chen (2004) generalize the problem of learning with edge-detecting queries to general reaction graphs and show that general graphs are efficiently learnable. In this work, we consider a more general problem when the chemicals react in groups of size more than two, that is, the underlying reaction network is a hypergraph. In Angluin and Chen (2004), they give an adaptive algorithm which takes $O(\log n)$ queries per edge, where n is the number of vertices. This is nearly optimal as we can easily show using an information-theoretic argument. For the problem of learning hypergraphs of bounded dimension and a given number of edges, a similar information-theoretic argument gives a lower bound that is linear in the number of edges. However, the lower bound is not achievable. It is shown in Angluin and Chen (2004) that $\Omega((2m/r)^{r/2})$ edge-detecting queries are required to learn a general hypergraph of dimension r with m edges. In the heart of the construction of Angluin and Chen (2004), edges of size 2 are deliberately arranged to hide an edge of size r. The discrepancy in sizes of different coexisting edges is the main barrier for the learner. However, this lower bound does not preclude efficient algorithms for classes of hypergraphs whose edges sizes are close. In particular, the question whether there is a learning algorithm for uniform hypergraphs using a number of queries that is linear in the number of edges is still left open, which is the main subject of this paper.

In this paper, we are able to answer this question affirmatively. Let n be the number of vertices and m be the number of edges in the hypergraph. We show that an r-uniform hypergraph is learnable with $O(2^{4r}m \cdot poly(r, \log n, \log \frac{1}{\delta}))$ queries with probability at least $1-\delta$.

We also obtain results for learning the class of hypergraphs that is almost uniform. Formally speaking,

Definition 1 *A hypergraph is (r,Δ)-uniform, where $\Delta < r$, if its dimension is r and the difference between its maximum and minimum edge sizes is Δ, or equivalently, the maximum and the minimum edge sizes are r and $r-\Delta$ respectively.*

The class of hypergraphs used in the construction of the lower bound in Angluin and Chen (2004) is in fact $(r, r-2)$-uniform. Therefore, they show that $\Omega((2m/r)^{r/2})$ edge-detecting queries are required to learn a $(r, r-2)$-uniform hypergraph. Based on this result, we show by a simple reduction that $\Omega((\frac{m}{1+\frac{\Delta}{2}})^{1+\frac{\Delta}{2}})$ queries are required to learn the class of (r,Δ)-uniform hypergraphs. On the other hand, we extend the algorithm that learns uniform hypergraphs to learning the class of

(r,Δ)-uniform hypergraphs with $O(2^{O((1+\frac{\Delta}{2})r)} \cdot m^{1+\frac{\Delta}{2}} \cdot poly(\log n, \log \frac{1}{\delta}))$ queries with probability at least $1-\delta$. The upper bound and lower bound have the same dependence on m.

Another important issue studied in the literature is the parallelism of algorithms. Since the queries are motivated by an experiment design scenario, it is desirable that experiments can be conducted in parallel. Alon et al. (2004) and Alon and Asodi (2005) give lower and upper bounds for 1-round algorithms for certain types of graphs. Beigel et al. (2001) describe an 8-round algorithm for learning a matching. Angluin and Chen (2004) give a 5-round algorithm for learning a general graph. In this paper, we show that in our algorithm for r-uniform hypergraphs, queries can be made in $O(\min(2^r(\log m+r)^2,(\log m+r)^3))$ rounds, and in our algorithm for (r,Δ)-uniform hypergraphs, queries can be made in $O((1+\Delta)\cdot\min(2^r(\log m+r)^2,(\log m+r)^3))$ rounds.

In the paper, we also introduce an interesting combinatorial object, which we call an *independent covering family*. Basically, an independent covering family of a hypergraph is a collection of independent sets that cover all non-edges. An interesting observation is that the set of negative queries of any algorithm that learns a hypergraph drawn from a class of hypergraphs that is closed under the operation of adding an edge is an independent covering family of that hypergraph. Note both the class of r-uniform hypergraphs and the class of (r,Δ)-uniform hypergraphs are closed under the operation of adding an edge. This implies that the query complexity of learning such a hypergraph is bounded below by the minimum size of its independent covering families. In the opposite direction, we give subroutines to find one arbitrary edge from a hypergraph. With the help of the subroutines, we show that if we can construct small-sized independent covering families for some class of hypergraphs, we are able to obtain an efficient learning algorithm for it. In this paper, we give a randomized construction of an independent covering family of size $O(r2^{2r}m\log n)$ for r-uniform hypergraphs with m edges. This yields a learning algorithm using a number of queries that is quadratic in m, which is further improved to give an algorithm using a number of queries that is linear in m.

As mentioned in Angluin and Chen (2004) and some other papers, the hypergraph learning problem may also be viewed as the problem of learning a monotone disjunctive normal form (DNF) boolean formula using membership queries only. Each vertex of H is represented by a variable and each edge by a term containing all variables associated with the vertices of the edge. A membership query assigns 1 or 0 to each variable, and is answered 1 if the assignment satisfies at least one term, and 0 otherwise, that is, if the set of vertices corresponding to the variables assigned 1 contains all vertices of at least one edge of H. An r-uniform hypergraph corresponds to a monotone r-DNF. An (r,Δ)-uniform hypergraph corresponds to a monotone DNF whose terms are of sizes in the range of $[r-\Delta,r]$. Thus, our results apply also to learning the corresponding classes of monotone DNF formulas using membership queries.

The paper is organized as follows. In Section 3, we formally define the concept of an independent covering family and give a randomized construction of independent covering families for general r-uniform hypergraphs. In Section 4, we show how to efficiently find an arbitrary edge in a hypergraph and give a simple learning algorithm using a number of queries that is quadratic in the number of edges. In Section 5, we give an algorithm that learns r-uniform hypergraphs using a number of queries that is linear in the number of edges. Then we derive a lower bound for almost uniform hypergraphs in Section 6. Finally, we show how to learn the class of (r,Δ)-uniform hypergraphs in Section 7.

2. Preliminaries

Let $H = (V, E)$ be a hypergraph. In this paper, we assume that edges do not contain each other, as there is no way to detect the existence of edges that contain other edges using edge-detecting queries. A subset of V is an *independent set* of H if it contains no edge of H. We use the term *non-edge* to denote any set that is a candidate edge in some class of hypergraphs but is not an edge in the target hypergraph. For example, in an r-uniform hypergraph, any r-set that is not an edge is a non-edge. In an (r,Δ)-uniform hypergraph, any set of size in the range of $[r-\Delta, r]$ that is not an edge is a non-edge. The *degree* of a set $\chi \subseteq V$ in a hypergraph H denoted as $d_H(\chi)$ is the number of edges of H that contain χ. In particular, $d_H(\emptyset) = |E|$ is the number of all edges in H.

Throughout the paper, we omit the ceiling and floor signs whenever they are not crucial.

3. An Independent Covering Family

Definition 2 An independent covering family *of a hypergraph H is a collection of independent sets of H such that every non-edge not containing an edge is contained in one of these independent sets.*

When H is a uniform hypergraph, the above only requires that every non-edge is contained in one of the independent sets in the independent covering family. An example is shown below.

Example 1 *Let $V = [1,6]$. Let $H = (V, \{\{1,2,3\}, \{4,5,6\}, \{2,4,5\}\})$ be a 3-uniform hypergraph.*

$$\mathcal{F} = \{\{1,2,4,6\}, \{1,2,5,6\}, \{1,3,4,5\}, \{1,3,4,6\}, \{2,3,4,6\}, \{2,3,5,6\}\}$$

is an independent covering family of H. As we can easily verify, all sets in $\mathcal{F}$ are independent sets, and every triple except $\{1,2,3\}, \{4,5,6\}, \{2,4,5\}$ is contained in some set in $\mathcal{F}$.

The concept of independent covering families is central in this paper. This can be appreciated from two aspects.

First, we observe that if the target hypergraph is drawn from a class of hypergraphs that is closed under the operation of adding an edge (e.g., the class of all r-uniform hypergraphs), the set of negative queries of any algorithm that learns it is an independent covering family of this hypergraph. This is because if there is a non-edge not contained in any of the sets on which these negative queries are made, we will not be able to distinguish between the target hypergraph and the hypergraph with this non-edge being an extra edge. Therefore, the minimum size of independent covering families bounds the query complexity from below. Furthermore, any learning algorithm gives a construction of an independent covering family of the target hypergraph. Therefore, in order to learn the hypergraph, we have to be able to construct an independent covering family for it.

Second, although the task of constructing an independent covering family seems substantially easier than that of learning, since the hypergraph is known in the construction task, we show that efficient construction of small-sized independent covering families yields an efficient learning algorithm. In Section 4, we will show how to find an arbitrary edge out of a hypergraph of dimension r using $O(r \log n)$ queries. Imagine a simple algorithm in which at each iteration we maintain a sub-hypergraph of the target hypergraph which contains edges that we have found, and construct an independent covering family for it and ask queries on all the sets in the family. If there is a set whose query is answered positively, we can find at least one edge out of this set. The edge must

be a new edge as the set is an independent set of the sub-hypergraph that we have found. We repeat this process until we have collected all the edges in the target hypergraph, in which case the independent covering family we construct is a proof of this fact. Suppose that we can construct an independent covering family of size at most $f(m)$ for any hypergraph with at most m edges drawn from certain class of hypergraphs. The above algorithm learns this class of hypergraphs using only $O(f(m) \cdot m \cdot r \log n)$ queries.

In the rest of this section, we give a randomized construction of a linear-sized (linear in the number of edges) independent covering family of an r-uniform hypergraph which succeeds with probability at least 1/2. By the standard probabilistic argument, the construction proves the existence of an independent covering family of size linear in the number of edges for any uniform hypergraph. This construction leads to a quadratic algorithm described in Section 4, and is also a central part of our main algorithm given in Section 5.

Our main theorem in this section is as follows.

Theorem 3 *Any r-uniform hypergraph with m edges has an independent covering family of size* $O(r2^{2r}m\log n)$.

Before giving the construction, we introduce some notation and definitions. We call a vertex set $\chi \subseteq V$ *relevant* if it is contained in at least one edge in the hypergraph. Similarly, a vertex is *relevant* if it is contained in at least one edge in the hypergraph. Let $p_H(\chi) = 1/(2^{r+1}d_H(\chi))^{1/(r-|\chi|)}$, where χ is a relevant vertex set. We will call $p_H(\chi)$ the *discovery probability* of χ, as this is a probability that will help in discovering edges containing χ in our learning algorithms.

Definition 4 *A* (χ, p)-sample *is a random set of vertices that contains* χ *and contains each other vertex independently with probability p.*

We will abbreviate (χ, p)-sample as χ-sample when the choice of p is clear or not important in the context.

In the construction, we draw $(\chi, p_H(\chi))$-samples independently for each relevant set χ. Each $(\chi, p_H(\chi))$-sample deals only with non-edges that contain χ. Let us take a look at the probability that a $(\chi, p_H(\chi))$-sample P_χ covers some non-edge $z \supseteq \chi$ while excluding all edges. Due to our choice of $p_H(\chi)$,

$$Pr[z \subseteq P_\chi] = p_H(\chi)^{r-|\chi|} = \frac{1}{2^{r+1}d_H(\chi)}.$$

Therefore, if we draw $2^{r+1}d_H(\chi)$ many χ-samples, the probability that z is contained in at least one χ-sample is $\Omega(1)$. However, such a χ-sample is not necessarily an independent set. Especially when z contains a high degree subset χ', it is likely that such a χ-sample contains an edges that contains χ'. But since we will also draw $(\chi', p_H(\chi'))$-samples, it is reasonable to hope that a $(\chi', p_H(\chi'))$-sample has better chance of success in dealing with z. In fact, in our construction, we show that the set of χ-samples, where $\chi \subseteq z$ has the minimum discovery probability among all relevant subsets of z, has an independent set that contains z with probability at least $1/2$.

A construction of an independent covering family is given in Algorithm 1, which succeeds with probability at least 1/2 as shown by Lemma 5.

Lemma 5 *$\mathcal{F}_H$ (constructed in Algorithm 1) contains an independent covering family of H with probability at least 1/2.*

Algorithm 1 Construction of an independent covering family

1: $\mathcal{F}_H \leftarrow$ a set containing $4(\ln 2 + r \ln n) \cdot 2^r d_H(\chi)$ $(\chi, p_H(\chi))$-samples drawn independently for every relevant set χ.
2: Output the independent sets contained in $\mathcal{F}_H$ as an independent covering family.

Proof Suppose z is a non-edge and χ is a subset of z with the minimum discovery probability. Let P_χ be a χ-sample. As argued before,

$$Pr[z \subseteq P_\chi] = \frac{1}{2^{r+1} d_H(\chi)}.$$

Since χ has the minimum discovery probability, the degree of any subset $\chi' \subseteq z$ is at most $1/(2^{r+1} p_H(\chi)^{r-|\chi'|})$. By the union bound,

$$\begin{aligned} Pr[P_\chi \text{ is independent} | z \subseteq P_\chi] &\geq 1 - \sum_{\chi' \subseteq z} d_H(\chi') p_H(\chi)^{r-|\chi'|} \\ &\geq 1 - \sum_{\chi' \subseteq z} \frac{1}{2^{r+1} p_H(\chi)^{r-|\chi'|}} p_H(\chi)^{r-|\chi'|} \\ &= 1/2. \end{aligned}$$

The probability that a χ-sample contains z and is independent is at least $1/(2^{r+2} d_H(\chi))$. Therefore, the probability that such a χ-sample exists in $\mathcal{F}_H$ is at least

$$\begin{aligned} & 1 - (1 - \frac{1}{2^{r+2} d_H(\chi)})^{4(\ln 2 + r \ln n) \cdot 2^r d_H(\chi)} \\ \geq & 1 - e^{-(r \ln n + \ln 2)} \\ = & 1 - \frac{1}{2n^{-r}}. \end{aligned}$$

Thus, the probability that every non-edge is contained in some negative sample in $\mathcal{F}_H$ is at least $1 - \binom{n}{r}/(2n^r) \geq 1/2$. ■

Theorem 3 is then established by the fact that the size of $\mathcal{F}_H$ is bounded by $\sum_\chi 4(\ln 2 + r \ln n) \cdot 2^r d_H(\chi) = O(r 2^{2r} m \log n)$.

4. A Simple Quadratic Algorithm

In this section, we first give an algorithm that finds an arbitrary edge in a hypergraph of dimension r using only $r \log n$ edge-detecting queries. The algorithm is adaptive and takes $r \log n$ rounds. The success probability in the construction of independent covering families in the previous section can be easily improved by drawing more samples. Using the high-probability version of the construction, we obtain an algorithm using a number of queries that is quadratic in m that learns an r-uniform hypergraph with m edges with high probability. Although the first algorithm for finding one edge is deterministic and simple, the round complexity $r \log n$ might be too high when n is much larger than m. We then improve the round complexity to $O(\log m + r)$ using only $O(\log m \log n)$ more queries. The improved algorithm is randomized and succeeds with high probability.

4.1 Find One Edge

We start with a simpler task, finding just one relevant vertex in the hypergraph. The algorithm FIND-ONE-VERTEX is shown in Algorithm 2.

Algorithm 2 FIND-ONE-VERTEX

$S \leftarrow V, A \leftarrow V$.
while $|A| > 1$ **do**
  Divide A arbitrarily into A_0 and A_1, such that $|A_0| = \lceil |A|/2 \rceil$, $|A_1| = \lfloor |A|/2 \rfloor$.
  if $Q(S \backslash A_0) = 0$ **then**
    $A \leftarrow A_0$.
  else
    $A \leftarrow A_1$, $S \leftarrow S \backslash A_0$.
  end if
end while
Output the element in A.

Lemma 6 *FIND-ONE-VERTEX finds one relevant vertex in a non-empty hypergraph with n vertices using at most* $\log n$ *edge-detecting queries.*

Proof First we show that the following equalities hold for each iteration (see Figure 1).

$$Q(S) = 1, Q(S \backslash A) = 0.$$

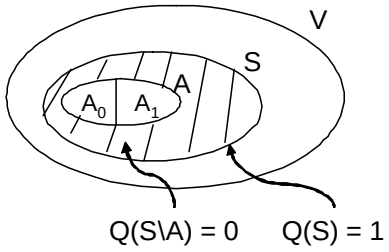


Figure 1: An illustration of FIND-ONE-VERTEX

These equalities guarantee that A contains at least one relevant vertex. Since we assume that the hypergraph is non-empty, the above equalities clearly hold for our initial assignment of S and A. Let's assume $Q(S) = 1$ and $Q(S \backslash A) = 0$ at the beginning of an iteration. There are two cases:

1. $Q(S \backslash A_0) = 0$, clearly the equalities hold for S and A_0.
2. $Q(S \backslash A_0) = 1$, since $Q((S \backslash A_0) \backslash A_1) = Q(S \backslash (A_0 \cup A_1)) = Q(S \backslash A) = 0$, the equalities hold for $S \backslash A_0$ and A_1.

Since the size of A halves at each iteration, after at most $\log n$ iterations, A has exactly one relevant vertex. The algorithm takes at most $\log n$ edge-detecting queries in total, as it makes one query in each iteration. ■

Using FIND-ONE-VERTEX as a subroutine, FIND-ONE-EDGE (Algorithm 3) is a recursive algorithm that finds one edge from a non-empty hypergraph, which is not necessarily uniform. Note knowledge of r is not required in FIND-ONE-EDGE. It is included in the description of the algorithm for the purpose of explanation.

Algorithm 3 FIND-ONE-EDGE

Let $r > 0$ be the dimension of the hypergraph.
Call FIND-ONE-VERTEX and let v be the found vertex.
Make a query on $\{v\}$.
if the query is answered 1 **then**
    Output $\{v\}$.
else
    FIND-ONE-VERTEX also computes a set S such that $Q(S) = 1$ and $Q(S \backslash \{v\}) = 0$. That is, S contains only edges incident with v.
    Call FIND-ONE-EDGE on the hypergraph induced on S with the vertex v removed. The hypergraph is of dimension at most $r-1$. Let e be the found edge.
    Output the edge $e \cup \{v\}$.
end if

Edge-detecting queries for recursive calls of FIND-ONE-EDGE can be simulated recursively. To make an edge-detecting query for a next-level recursive call of FIND-ONE-EDGE, we just need to make an edge-detecting query at the current level on the union of a subset of S and $\{v\}$. In fact, each time, we make edge-detecting queries on the union of a subset of S and the set of vertices already found.

In FIND-ONE-EDGE, because S contains only edges incident with v, $e \cup \{v\}$ is an edge in the hypergraph. This establishes its correctness. The following lemma shows that it uses only $r \log n$ queries.

Lemma 7 *FIND-ONE-EDGE finds one edge in a non-empty hypergraph of dimension r with n vertices using $r \log n$ edge-detecting queries.*

Proof When $r = 1$, the problem is exactly that of finding one relevant vertex and hence solvable using $\log n$ queries. It is evident that if FIND-ONE-EDGE uses $(r-1)\log n$ queries for a hypergraph with dimension $r-1$. then it only uses $(r-1)\log n + \log n = r \log n$ queries for a hypergraph with dimension r. ■

4.2 A Quadratic Algorithm

With the help of FIND-ONE-EDGE, we give the first learning algorithm for r-uniform hypergraphs. A sketch of the algorithm has been described in Section 3. Let $H = (V, E)$ be the sub-hypergraph the algorithm has found so far. Algorithm 4 learns a uniform hypergraph with probability at least $1 - \delta$. We will specify δ' later.

In the algorithm we draw $4(\ln(\frac{1}{\delta'}) + r \ln n) \cdot 2^r d_H(\chi)$ χ-samples. Using essentially the same argument as in Section 3, we can guarantee that $\mathcal{F}_H$ contains an independent covering family with

Algorithm 4 The quadratic algorithm

1: $e \leftarrow$ FIND-ONE-EDGE(V). $E \leftarrow \{e\}$.
2: **repeat**
3: $\mathcal{F}_H \leftarrow 4(\ln \frac{1}{\delta'} + r \ln n) \cdot 2^r d_H(\chi)$ $(\chi, p_H(\chi))$-samples drawn independently for every relevant set χ in H.
4: Make queries on sets of $\mathcal{F}_H$ that are independent in H.
5: Call FIND-ONE-EDGE on one positive sample if there exists any. Let e be the edge found. $E \leftarrow E \cup \{e\}$.
6: **until** no new edge found

probability at least $1-\delta'$. Algorithm 4 finds one new edge at each iteration because $\mathcal{F}_H$ is an independent covering family of the already found sub-hypergraph H. Thus, it ends after at most m iterations. If we we choose $\delta' = \delta/m$, it will succeed with probability at least $1-\delta$. As knowledge of m is not assumed, we will choose $\delta' = \delta/\binom{n}{r} \leq \delta/m$. The query complexity will be $O(2^{2r}m^2 \cdot poly(r, \log n) \cdot \log \frac{1}{\delta})$, which is quadratic in m.

4.3 An Improved FIND-ONE-EDGE

Despite the simplicity of FIND-ONE-EDGE, its queries have to be made in $r \log n$ rounds. When irrelevant vertices abound, that is, when n is much larger than m, we would like to arrange queries in a smaller number of rounds. In the following, we use a technique developed in Damaschke (1998) (for learning monotone boolean functions) to find one edge from a non-empty hypergraph with high probability using only $O(\log m + r)$ rounds and $O((\log m + r)\log n)$ queries. However, the algorithm is more involved.

The new algorithm is also based on FIND-ONE-VERTEX. The process of FIND-ONE-VERTEX can be viewed as a binary decision tree. At each internal node, a set A is split and a decision on which branch to follow is made based on query results. The FIND-ONE-VERTEX algorithm does not restrict how we split the set A as long as we divide it into halves. In the new algorithm, we will pre-determine the way A's will be divided at the very beginning of the algorithm.

Let us index each vertex by a distinct binary number $b_1 b_2 \ldots b_{\log n}$. Each split is based on a certain bit. We say that we split a set A according to its i^{th} ($i \in [1, \log n]$) bit, we will divide A into two sets, one containing vertices whose i^{th} bits are 0 and the other containing vertices whose i^{th} bits are 1. We will denote these two sets $A|_{b_i=0}$ and $A|_{b_i=1}$ respectively. If we split $A|_{b_i=0}$ further according to the j^{th} bit, we get another two sets $(A|_{b_i=0})|_{b_j=0}$ and $(A|_{b_i=0})|_{b_j=1}$. We will abbreviate these two sets as $A|_{b_i=0,b_j=0}$ and $A|_{b_i=0,b_j=1}$. In general, let s be a *partial assignment* that assigns some bits to 0 or 1, we use $A|_s$ to denote the set of vertices in A that match the assignments of bits in s.

Using this notation and our splitting scheme, at each iteration of FIND-ONE-VERTEX, A is equal to $V|_s$ for some partial assignment s, and A_0 and A_1 are equal to $A|_{b_i=0}$ and $A|_{b_i=1}$ if we split A according to the i^{th} bit. One of the key ideas in Damaschke (1998) is that because the splits are pre-determined, and the queries are monotone in terms of subset relation, we can make queries on pre-determined splits to make predictions. The idea will be made clear in the rest of the section. PARA-FIND-ONE-VERTEX (Algorithm 5) improves the round complexity of FIND-ONE-VERTEX.

Algorithm 5 PARA-FIND-ONE-VERTEX

$S \leftarrow V, A \leftarrow V, I \leftarrow [1, \log n]$.
while $|A| > 1$ **do**
  $\forall i \in I$, make queries on $(S \backslash A) \cup A|_{b_i=0}$ and $(S \backslash A) \cup A|_{b_i=1}$.
  Let $R_i = (Q((S \backslash A) \cup A|_{b_i=0}), Q((S \backslash A) \cup A|_{b_i=1}))$ be the query results for $i \in I$.
  case 1: $\exists i \in I$ such that $R_i = (0,0)$
    $A \leftarrow A|_{b_i=0}, I \leftarrow I \backslash \{i\}$.
  case 2: $\exists i \in I$ such that $R_i = (1,1)$
    Choose a from $\{0,1\}$ uniformly at random.
    $A \leftarrow A|_{b_i=a}$, $S \leftarrow (S \backslash A) \cup A|_{b_i=a}, I \leftarrow I \backslash \{i\}$.
  case 3: $\forall i \in I$, $R_i = (1,0)$ or $R_i = (0,1)$
    Swap the indices of vertices so that $R_i = (1,0)$ for every $i \in I$. (If $R_i = (0,1)$, we flip the i^{th} bit of all the indices, that is, for every vertex, if the i^{th} bit of its index is 0, we set the i^{th} bit to 1 and vice versa.)
    $\forall i \in I$, let $A^i = A|_{\forall j \in I, j \leq i, b_j=0}$ and make a query on $S^i = (S \backslash A) \cup A^i$.
    Let $i^* = \min\left\{i | Q(S^i) = 0\right\}$ if it exists and the largest index in I otherwise. Let $j^* = \max\{j | j < i^*, j \in I\}$.
    $I \leftarrow \{i | i > i^*, i \in I\}$.
    if all queries are answered 1 **then**
      $A \leftarrow A^{i^*}, S \leftarrow S^{i^*}$ (i^* is the largest index in I in this case).
    else
      $A \leftarrow A^{i^*}, S \leftarrow S^{j^*}$.
    end if
end while
Output the element in A.

In PARA-FIND-ONE-VERTEX, the equalities $Q(S) = 1, Q(S \backslash A) = 0$ are also preserved at all times, which establishes the correctness. We first make queries on $(S \backslash A) \cup A|_{b_i=0}$ ($= S \backslash A|_{b_i=1}$) and $(S \backslash A) \cup A|_{b_i=1}$ ($= S \backslash A|_{b_i=0}$) for every i. There are 3 possible query outcomes.

case 1: If there exists i such that $R_i = (0,0)$, that is, both queries are answered 0, all edges contained in S are split between $A|_{b_i=0}$ and $A|_{b_i=1}$, that is, the intersections of each edge with these two sets are a partition of the edge. We call this case an *edge-splitting event*. The iterations at which an edge-splitting event happens are *edge-splitting iterations*. Since we then set A to be $A|_{b_i=0}$, the intersection of A with any edge contained in S becomes strictly smaller. Because we will only shrink A in other cases, the intersections will never increase. Thus, there are at most $r-1$ edge-splitting iterations as each edge is of size at most r.

case 2: If there exists i such that $R_i = (1,1)$, that is, both queries are answered 1, we can set S to be either of the two sets $(S \backslash A) \cup A|_{b_i=0}$ and $(S \backslash A) \cup A|_{b_i=1}$ as they both contain edges, and set A to be $A|_{b_i=0}$ or $A|_{b_i=1}$ respectively. The equalities $Q(S) = 1, Q(S \backslash A) = 0$ are preserved in this case. However, we would like to choose whichever of the two sets $(S \backslash A) \cup A|_{b_i=0}$ and $(S \backslash A) \cup A|_{b_i=1}$ contains fewer edges. Because they do not share a common edge as their intersection $S \backslash A$ does not contain an edge, the sum of the numbers of edges contained in these two sets is at most the number of edges contained in S. If we choose the set with fewer

edges, we will cut the number of edges contained in S in half. With a random choice, this happens with probability $1/2$. We will call this case an *edge-separating event* and call the corresponding iteration an *edge-separating iteration*.

case 3: If neither of the two events happens, we need to deal with the third case where $\forall i \in I$, one of the queries is answered 0 and the other is answered 1. In this case, for convenience of exposition, we will flip the indices of all vertices, so that $R_i = (1,0)$ for every $i \in I$. Thus, $\forall i \in I, Q((S\backslash A) \cup A|_{b_i=0}) = 1$. In this case, we can set A to $A|_{b_i=0}$ for some $i \in I$, and the equalities $Q(S) = 1, Q(S\backslash A) = 0$ are preserved. However, this won't help us to reduce the round complexity as it only cuts A in half.

Consider the next split. We shall divide $A|_{b_i=0}$ further into $A|_{b_i=0,b_j=0}$ and $A|_{b_i=0,b_j=1}$ for some $j \in I, j \neq i$. Since we already know that $Q((S\backslash A) \cup A|_{b_j=1}) = 0$, the fact that $A|_{b_i=0,b_j=1}$ is a subset of $A|_{b_j=1}$ implies $Q((S\backslash A) \cup A|_{b_i=0,b_j=1}) = 0$. Therefore, we only need to know $Q((S\backslash A) \cup A|_{b_i=0,b_j=0})$.

(a) If it is 1, we can set A to be $A|_{b_i=0,b_j=0}$ and continue.

(b) Otherwise, it is 0, an edge-splitting event takes place.

In PARA-FIND-ONE-VERTEX, we choose the indices we use to split A in the increasing order of indices in I and make queries on $S^i = (S\backslash A) \cup A^i$ for every $i \in I$ all in parallel (recall that $A^i = A|_{\forall j \in I, j \leq i, b_j=0}$). If all queries are answered 1, i^* is the largest index in I and A^{i^*} is a singleton set containing a relevant vertex. Otherwise, we get an edge-splitting event, since $S^{j^*} = (S\backslash A) \cup A^{j^*}$ contains edges, but both $(S\backslash A) \cup A^{j^*}|_{b_{i^*}=0}$ and $(S\backslash A) \cup A^{j^*}|_{b_{i^*}=1}$ don't (note that j^* is the index right before i^* in the increasing order of indices in I and $A^{i^*} = A^{j^*}|_{b_{i^*}=0}$). In this case, it can be verified that our updates to A and S in the third case preserve the equalities $Q(S) = 1, Q(S\backslash A) = 0$.

By the above analysis, the first case and the third case both result in an edge-splitting event or succeed in finding a relevant vertex. There are at most r such iterations. The second case results in an edge-separating event, in which with probability $1/2$ we will cut the number of edges contained in S in half. We can show that in expectation there are $\log m$ edge-separating events. Therefore, there are $\log m + r$ iterations in expectation. At each iteration, we use at most $3\log n$ queries which are made in at most 2 rounds. Therefore,

Lemma 8 *In expectation, PARA-FIND-ONE-VERTEX finds one relevant vertex using $O((\log m + r)\log n)$ queries, and the queries can be made in $2(\log m + r)$ rounds.*

PARA-FIND-ONE-VERTEX can work with FIND-ONE-EDGE to find an edge using expected $O(r(\log m + r)\log n)$ queries in expected $2r(\log m + r)$ rounds. In fact, we can improve the round complexity further to $2(\log m + r)$ based on two observations, both of which use the fact that in the whole process we only shrink S.

The first observation is that edges removed from S in the edge-separating events will not be considered again. Therefore, the $\log m$ bounds not only the expected number of edge-separating iterations of PARA-FIND-ONE-VERTEX, but also that of the whole process.

The second observation is that the edge-splitting events can be remembered and reused when we try to find the next relevant vertex. Since we only shrink S, the bits that split all edges in S will continue to do so. Let I^* be the set of edge-splitting indices. In the new vertex finding process,

instead of starting with $A = V = S$ (recall in a recursive call of FIND-ONE-EDGE, we look for a relevant edge contained in the S. Therefore, in the recursive call, V is equal to the S we obtain in the previous call), we start with $A = S|_{i \in I^*, b_i = 0}$. Note that the equalities $Q(S) = 1, Q(S \backslash A) = 0$ are preserved. This helps us to bound the number of edge-splitting iterations by $r-1$ for the whole process.

Thus, we have the following lemma.

Lemma 9 *There is an algorithm that finds an edge in a non-empty hypergraph using expected $O((\log m + r)\log n)$ edge-detecting queries. Moreover, the queries can be made in expected $2(\log m + r)$ rounds.*

Since the algorithm terminates in expected $\log m + r$ iterations, according to Markov's Inequality, with probability at least $1/2$, the algorithm terminates in $2(\log m + r)$ iterations. We convert it to one that succeeds with high probability by running $\log \frac{1}{\delta}$ copies, each of which has its own independent random choices. All copies are synchronized at each iteration and the algorithm ends when one of them succeeds. This leads to an algorithm that succeeds with high probability. We will refer to this algorithm as PARA-FIND-ONE-EDGE.

Corollary 10 *With probability at least $1-\delta$, PARA-FIND-ONE-EDGE finds an edge using $O((\log m + r)\log n \log \frac{1}{\delta})$ edge-detecting queries, and the queries can be made in $4(\log m + r)$ rounds.*

5. A Linear-Query Algorithm

Reconstructing an independent covering family at the discovery of every new edge is indeed wasteful. In this section we show how to modify the quadratic algorithm to obtain an algorithm using a number of queries that is linear in the number of edges. Our algorithm is optimal in terms of the dependence on m. Moreover, the queries can be made in $O(\min(2^r(\log m + r)^2, (\log m + r)^3))$ rounds.

Before we begin to describe our algorithm, we introduce some notation and make some definitions. First we reduce the discovery probabilities. Let

$$p_H(\chi) = 1/(2^{r+|\chi|+2} d_H(\chi))^{1/(r-|\chi|)},$$

where χ is a relevant vertex set. Let the *best discovery probability* of χ be the minimum discovery probability among all its subsets. That is,

$$p_H^*(\chi) = \min_{\chi' \subseteq \chi} p_H(\chi').$$

Definition 11 *Let $\rho_\chi(p)$ be the probability that a (χ, p)-sample is positive, where χ is a relevant vertex set.*

Remark 12 *$\rho_\chi(p)$ is continuous and monotonically increasing.*

Angluin and Chen (2004) contains a proof of this fact.

Definition 13 *Let $p_\chi = \min\{p | \rho_\chi(p) = 1/2^{r+1}\}$ be the* threshold probability *of a relevant vertex set χ.*

Remark 14 *Due to the fact that* $\rho_\chi(0) = 0$, $\rho_\chi(1) = 1$ *and that* $\rho_\chi(p)$ *is continuous and monotonically increasing, the threshold probability uniquely exists.*

Note that both threshold probabilities and discovery probabilities reflect the degree of set χ or the degrees of its subsets. The difference is that discovery probabilities reflect degrees in the hypergraph we have found, while threshold probabilities reflect degrees in the target hypergraph. Threshold probabilities are only used in analysis.

5.1 Overview Of The Algorithm

An "obvious" improvement to the quadratic algorithm is that instead of calling FIND-ONE-EDGE on one positive sample at each iteration, we can call it on all positive samples. It is plausible that this will yield more edges. However, there is no guarantee that different calls to FIND-ONE-EDGE will output different edges. For instance, calls to FIND-ONE-EDGE on two sets that share a common edge will produce the same edge in the worst case. We use several standard tricks to circumvent this obstacle. In fact, the family of samples constructed here is more complex than that used in Section 4, so as to ensure with high probability that the algorithm will make a certain amount of progress at each iteration. By doing so, we are able to reduce the number of iterations from m to $O(\min(2^r(\log m + r), (\log m + r)^2))$. The number of queries will also be reduced.

First of all, the sampling probabilities are halved in order to accommodate more edges. More precisely, imagine that we draw $(\chi, \frac{1}{2}p_H(\chi))$-samples instead of $(\chi, p_H(\chi))$-samples in the quadratic algorithm. Take a look at a sample drawn several iterations ago, which the quadratic algorithm did not call FIND-ONE-EDGE on. Such a sample will still have reasonable probability of excluding all the edges that have been found, as long as the degree of χ has not been increased by a factor of $2^{r-|\chi|}$ or equivalently the discovery probability of χ has not been decreased by half.

Second, the algorithm uses the best discovery probability for each relevant set. We call a relevant vertex set *minimal* if it has the minimum discovery probability among its subsets. In the quadratic algorithm, the goal is that one of the samples will produce an edge. According to the proof of Lemma 5, in the quadratic algorithm, we actually only need to draw samples for minimal relevant sets. In this algorithm, we hope that samples drawn for every relevant set will produce edges. But drawing samples for non-minimal relevant sets with discovery probabilities is not sufficient to avoid edges we have already found. Therefore, the best discovery probabilities are used.

Finally, besides samples drawn proportional to degrees, the algorithm also draws samples proportional to the contribution of each relevant set. The idea is simple. Draw more samples for those relevant sets that are more likely to produce a new edge. The algorithm maintains a *contribution* counter $c(\chi)$ for each relevant set χ, which records the number of new edges that χ-samples have produced. As we have already said, different calls to FIND-ONE-EDGE at each iteration may output the same edge. As all calls to FIND-ONE-EDGE at each iteration are made in parallel, it is not clear which sample each new edge should be attributed to. To solve this problem, calls to FIND-ONE-EDGE are processed sequentially in an arbitrary order.

In the algorithm, $\mathcal{F}_H$ consists of two parts: $\mathcal{F}_H^1$ and $\mathcal{F}_H^2$. In $\mathcal{F}_H^1$, the algorithm draws samples proportional to the contribution of each relevant set. $\mathcal{F}_H^2$ is closer to $\mathcal{F}_H$ in Section 4. Intuitively, a high-degree relevant set in the target hypergraph (not necessarily a high-degree relevant set in H), or a relevant set with small threshold probability is important, because an edge or a non-edge may not be found if its important relevant subsets are not found. The smaller the threshold probability a relevant set is, the more important it is. The algorithm uses samples in $\mathcal{F}_H^1$ to find edges while

samples in $\mathcal{F}_H^2$ are mainly used to cover non-edges of H. $\mathcal{F}_H^2$ not only gives a short proof when H is indeed the target hypergraph, but also finds important relevant sets quickly. The design of $\mathcal{F}_H^2$ guarantees that if the contribution of the most important subset of an edge or a non-edge stops doubling, a more important relevant subset will be discovered with high probability.

5.2 The Algorithm

Let $H = (V,E)$ be the hypergraph the algorithm has found so far. δ' is a parameter we will specify later. The algorithm is shown in Algorithm 6. At each iteration, the algorithm operates in two phases, the query phase and the computation phase. In the query phase, the algorithm draws random samples and make queries. The queries can be made in $O(\log m + r)$ rounds, as queries of each call to PARA-FIND-ONE-EDGE can be made in $O(\log m + r)$ rounds. In the computation phase, the algorithm processes the query results to update the contribution counter of each relevant set and also adds newly found relevant sets.

Algorithm 6 The linear-query algorithm

All PARA-FIND-ONE-EDGE's are called with parameter δ'.
1: $e \leftarrow$ PARA-FIND-ONE-EDGE(V).
2: $E \leftarrow \{e\}$. $c(\emptyset) \leftarrow 1$.
3: **repeat**
QUERY PHASE
4: Let $\mathcal{F}_H^1$ be a family that for every known relevant set χ contains $c(\chi) \cdot 2^{r+2} \ln \frac{1}{\delta'}$ $(\chi, \frac{1}{2} p_H^*(\chi))$-samples.
5: Let $\mathcal{F}_H^2$ be a family that for every known relevant set χ contains $2^{3r+3} d_H(\chi) \ln \frac{1}{\delta'}$ $(\chi, \frac{1}{4} p_H^*(\chi))$-samples.
6: Let $\mathcal{F}_H = \mathcal{F}_H^1 \cup \mathcal{F}_H^2$. Make queries on sets in $\mathcal{F}_H$ that are independent in H.
7: Call PARA-FIND-ONE-EDGE on all positive samples.
COMPUTATION PHASE
8: For each known relevant set χ, divide χ-samples in $\mathcal{F}_H^1$ into $c(\chi)$ groups of size $2^{r+2} \ln \frac{1}{\delta'}$.
9: Process the samples in $\mathcal{F}_H^1$ group by group in an arbitrary order. Increase $c(\chi)$ by the number of new edges that χ-samples produce. Add newly found edges to E.
10: Process the samples in $\mathcal{F}_H^2$. Add newly found edges to E.
11: For every newly found relevant set χ, $c(\chi) \leftarrow 1$.
12: **until** no new edge is found

We will show that the algorithm terminates in $O(\min(2^r(\log m + r), (\log m + r)^2))$ iterations with high probability. Since $\sum_\chi d_H(\chi) \le 2^r m$ and $\sum_\chi c(\chi) \le (2^r + 1)m$ (note that $c(\chi)$ is one more than the number of new edges that χ-samples in $\mathcal{F}_H^1$ produce), the number of queries made at each iteration is at most $O(2^{4r} m \cdot poly(r, \log n, \log \frac{1}{\delta'}))$. Therefore, the total number of queries will be linear in the number of edges with high probability, as desired.

5.3 Analysis

Consider some iteration of the algorithm. Let H be the hypergraph the algorithm has found at the beginning of the iteration. Let e be an edge that has not yet been found. Let χ be a known subset of

e. χ can be either *active*, in which case a χ-sample is likely to contain an edge or *inactive* otherwise. Formally speaking,

Definition 15 *We say that χ is active if $\rho_\chi(\frac{1}{2}p^*_H(\chi)) \geq 1/2^{r+1}$ or, equivalently, $\frac{1}{2}p^*_H(\chi) \geq p_\chi$, and inactive otherwise.*

The following two assertions serve as the goals for each iteration.

Assertion 16 *Consider one group of χ-samples $\mathcal{G}$ in $\mathcal{F}^1_H$. Let H' be the hypergraph the algorithm has found before this group is processed. If χ is active, either $p^*_{H'}(\chi) < \frac{1}{2}p^*_H(\chi)$ or $\mathcal{G}$ will produce a new edge.*

Assertion 17 *If χ is inactive, then at the end of this iteration, either e has been found or a subset of e whose threshold probability is at most $\frac{1}{2}p_\chi$ has been found (a relevant subset is found when an edge that contains it is found).*

The following two lemmas show that both assertions hold with high probability.

Lemma 18 *Assertion 16 is true with probability at least $1-\delta'$.*

Proof If $p^*_{H'}(\chi) \geq \frac{1}{2}p^*_H(\chi)$, the probability that a χ-sample contains an edge in H' is at most

$$\sum_{\chi' \subseteq \chi} d_{H'}(\chi')(\frac{1}{2}p^*_H(\chi))^{r-|\chi'|} \leq \sum_{\chi' \subseteq \chi} d_{H'}(\chi')p^*_{H'}(\chi)^{r-|\chi'|} \leq \frac{2^{|\chi|}}{2^{r+|\chi|+2}} = \frac{1}{2^{r+2}}.$$

On the other hand, since χ is active, we have $\rho_\chi(\frac{1}{2}p^*_H(\chi)) \geq 1/2^{r+1}$. That is, with probability at least $1/2^{r+1}$ a χ-sample will contain an edge. Therefore the probability that a χ-sample contains a new edge is at least $1/2^{r+1} - 1/2^{r+2} = 1/2^{r+2}$. Recall that $\mathcal{G}$ contains $2^{r+2}\ln\frac{1}{\delta'}$ χ-samples. Therefore, with probability at least $1-\delta'$ there exists at least one sample in $\mathcal{G}$ that will produce a new edge. ■

Lemma 19 *Assertion 17 is true with probability at least $1-\delta'$.*

Proof Let $\chi^* \subseteq \chi$ have the minimum discovery probability among all subsets of χ at the beginning of the iteration. Thus, $p_H(\chi^*) = p^*_H(\chi)$ by the definition. Let us consider a χ^*-sample P_{χ^*} in $\mathcal{F}^2_H$. Let A be the collection of all subsets of e whose threshold probabilities are not less than $\frac{1}{2}p_\chi$. We do not want P_{χ^*} to contain any edge that contains χ' for any $\chi' \in A$ because they prevent us from discovering relevant sets with low threshold probabilities ($< \frac{1}{2}p_\chi$).

We observe that $\frac{1}{2}p_H(\chi^*) = \frac{1}{2}p^*_H(\chi) < p_\chi$ because χ is inactive. Thus, we have that $\forall \chi' \in A$,

$$\rho_{\chi'}(\frac{1}{4}p_H(\chi^*)) < \rho_{\chi'}(\frac{1}{2}p_\chi) \leq \rho_{\chi'}(p_{\chi'}) = 1/2^{r+1}.$$

Therefore,

$$Pr[\exists \text{ an edge } e' \subseteq P_{\chi^*}, e' \cap e \in A | e \subseteq P_{\chi^*}] \leq \sum_{\chi' \in A} \rho_{\chi'}(\frac{1}{4}p_H(\chi^*)) \leq 1/2.$$

Combining with the fact that

$$Pr[e \subseteq P_{\chi^*}] = (\frac{1}{4}p_H(\chi^*))^{r-|\chi^*|} = \frac{1}{2^{r+|\chi^*|+2+2r-2|\chi^*|}d_H(\chi^*)} \geq \frac{1}{2^{3r+2}d_H(\chi^*)},$$

we have that with probability at least $1/(2^{3r+3}d_H(\chi^*))$, P_{χ^*} contains e but does not contain any edge whose intersection with e is in A, in which case PARA-FIND-ONE-EDGE(P_{χ^*}) either outputs e or outputs an edge whose intersection with e has threshold probability at most $\frac{1}{2}p_\chi$. The probability that such a P_{χ^*} exists in $\mathcal{F}_H^2$ is at least $1-\delta'$, as we draw at least $2^{3r+3}d_H(\chi^*)\ln\frac{1}{\delta'}$ $(\chi^*, \frac{1}{4}p_H(\chi^*))$-samples. ■

Let H' be the hypergraph that has been found at the end of the iteration. Let $c_H(\chi)$ and $c_{H'}(\chi)$ be the values of $c(\chi)$ at the beginning and end of the iteration respectively. At each iteration, if no assertion is violated, one of the following two events happens.

1. Either $c_{H'}(\chi) \geq 2c_H(\chi)$ or $p^*_{H'}(\chi) < \frac{1}{2}p^*_H(\chi)$. ($c_H(\chi)$ doubles when each of the $c_H(\chi)$ groups of χ-samples in $\mathcal{F}_H^1$ succeeds in producing a new edge.)

2. Either e has been found or a subset of e whose threshold probability is at most $\frac{1}{2}p_\chi$ has been found.

That is, the two assertions guarantee that the algorithm makes definite progress at each iteration. The following lemma gives bound on the number of iterations of the algorithm.

Lemma 20 *Assuming no assertion is violated, the algorithm terminates in* $O(\min(2^r(\log m + r), (\log m + r)^2))$ *iterations.*

Proof First we remark that the minimum and maximum possible values for both discovery probabilities and threshold probabilities are $1/(2^{2r+1}m)$ and $1/2$ respectively, and the minimum and maximum possible values for $c(\chi)$ are 1 and $m+1$.

For each edge e, we divide the iterations into phases until e is found. Each phase is associated with a known relevant subset χ of e which has the minimum threshold probability at the beginning of the phase. A χ-phase ends when χ becomes inactive and then either e will be found or another relevant subset of e with at most half of χ's threshold probability will be found. Let us associate χ's threshold probability with a χ-phase. There are certainly at most 2^r phases because this is a bound on the number of subsets of e. Moreover, there are at most $O(\log m + r)$ phases as the associated threshold probability halves at the end of each phase. Furthermore, each phase takes at most $O(\log m + r)$ iterations, since either $c(\chi)$ doubles or the best discovery probability halves at each iteration. Therefore the algorithm terminates in $O(\min(2^r(\log m + r), (\log m + r)^2))$ iterations. ■

It is not hard to see that total number of assertions we need to satisfy before the algorithm succeeds is bounded by $poly(2^r, m)$, including the assertions that each PARA-FIND-ONE-EDGE will succeed. Choose $\delta' = \Theta(\delta/poly(2^r, m))$ and the algorithm will succeed with probability at least $1-\delta$. Although the choice of δ' requires knowledge of m, it is sufficient to use an upper bound of $\binom{n}{r}$, and we have that $\log\frac{1}{\delta'} \leq poly(r, \log n)\cdot\log\frac{1}{\delta}$. Since queries at each iteration are made in $O(\log m + r)$ rounds, it follows that

Theorem 21 *With probability at least $1-\delta$, Algorithm 6 learns an r-uniform hypergraph with m edges and n vertices, using $O(2^{4r}m \cdot poly(r, \log n, \log \frac{1}{\delta}))$ queries, in $O(\min(2^r(\log m + r)^2, (\log m + r)^3))$ rounds.*

6. Lower Bounds For Almost Uniform Hypergraphs

In this section, we derive a lower bound for the class of (r,Δ)-uniform hypergraphs. The following theorem is proved in Angluin and Chen (2004).

Theorem 22 $\Omega((2m/r)^{r/2})$ *edge-detecting queries are required to identify a hypergraph drawn from the class of all $(r, r-2)$-uniform hypergraphs with n vertices and m edges.*

We show that by a simple reduction this gives us a lower bound for general (r,Δ)-uniform hypergraphs.

Theorem 23 $\Omega((2m/(\Delta+2))^{1+\frac{\Delta}{2}})$ *edge-detecting queries are required to identify a hypergraph drawn from the class of all (r,Δ)-uniform hypergraphs with n vertices and m edges.*

Proof Given a $(\Delta+2,\Delta)$-uniform hypergraph $H=(V,E)$, let $H'=(V\cup V',E')$ be an (r,Δ)-uniform hypergraph, where $|V'| = r-\Delta-2$, $V'\cap V = \phi$ and $E' = \{e\cup V' | e \in E\}$. Any algorithm that learns H' can be converted to learn H with the same number of queries. ■

7. Learning Almost Uniform Hypergraphs

In this section, we extend our results to learning (r,Δ)-uniform hypergraphs. The query upper bound stated in the following theorem matches the lower bound of Theorem 23 in terms of dependence on m. The round upper bound is only $1+\Delta$ times more than that of Algorithm 6.

Theorem 24 *There is a randomized algorithm that learns an (r,Δ)-uniform hypergraph with m edges and n vertices with probability at least $1-\delta$, using $O(2^{O((1+\frac{\Delta}{2})r)} \cdot m^{1+\frac{\Delta}{2}} \cdot poly(\log n, \log \frac{1}{\delta}))$ queries. Furthermore, the queries can be made in $O((1+\Delta)\cdot \min(2^r(\log m + r)^2, (\log m + r)^3))$ rounds.*

7.1 The Algorithm

One of the main modifications is the use of new discovery probabilities. We first provide some intuition for the new discovery probabilities. We have been choosing the discovery probability for a relevant set χ to be inversely proportional to the $(r-|\chi|)^{th}$ root of its degree. It is so chosen that a χ-sample has good chance of excluding edges that contain χ. In an almost uniform hypergraph, we choose the discovery probabilities for the same purpose. In other words, we would like to choose p such that $\sum_{e\in E, e\supseteq\chi} p^{|e\setminus\chi|} \le 1/2^{r+2}$. Similarly, we should set p to be inversely proportional to the w^{th} root of $d_H(\chi)$, where $w = \min_{e\supseteq\chi} |e\setminus\chi|$ is the minimum difference in cardinalities between edges containing χ and χ. However, w is no longer equal to $r-|\chi|$ as in uniform hypergraphs. There are two cases. When $|\chi| < r-\Delta$, we have $w \ge r-\Delta-|\chi|$ because the minimum edge size is $r-\Delta$; when $|\chi| \ge r-\Delta$, w can be as small as 1.

The case when $w = 1$ is special, as it implies that there exists an edge e such that $|e \backslash \chi| = 1$ or e has only one vertex v that χ does not have. We will call e a *1-edge* of χ. On one hand, any χ-sample containing v contains e, and hence is not an independent set; on the other hand, by excluding every vertex whose union with χ contains an edge, we can easily exclude all corresponding edges. Thus we remove these vertices from each χ-sample and the resulting sample, which we call a *modified χ-sample*, is an improvement over the original one. (We remark that this improvement is available for the uniform hypergraph problem in the case when $|\chi| = r - 1$, but is not as important.) More specifically, let $\nu_H(\chi)$ be the set of all vertices v such that $\chi \cup \{v\}$ contains an edge in H. A modified (χ, p)-sample is a $(\chi, \nu_H(\chi), p)$-sample defined as follows.

Definition 25 *A (χ, ν, p)-sample ($\chi \cap \nu = \emptyset$) is a random set of vertices that contains χ and does not contain any vertex in ν and contains each other vertex independently with probability p.*

Algorithm 7 Learning an (r, Δ)-uniform hypergraph

All PARA-FIND-ONE-EDGE's are called with parameter δ'.

1: $e \leftarrow$ PARA-FIND-ONE-EDGE(V).
2: $E \leftarrow \{e\}$. $c(\emptyset) \leftarrow 1$.
3: **repeat**

QUERY PHASE

4: Let $\mathcal{F}_H^1$ be a family that for every known relevant set χ contains $c(\chi) \cdot 2^{r+2} \ln \frac{1}{\delta'}$ modified $(\chi, \frac{1}{2} p_H^*(\chi))$-samples and the same number of modified $(\chi, \frac{1}{2^{r+3+|\chi|} d_H(\chi)})$-samples.
5: Let $\mathcal{F}_H^2$ be a family that for every known relevant set χ contains $2(4/p_H(\chi))^{r-|\chi|} \ln \frac{1}{\delta'}$ modified $(\chi, \frac{1}{4} p_H^*(\chi))$-samples and $2^{r+4+|\chi|} d_H(\chi) \ln \frac{1}{\delta'}$ modified $(\chi, \frac{1}{2^{r+3+|\chi|} d_H(\chi)})$-samples.
6: Let $\mathcal{F}_H = \mathcal{F}_H^1 \cup \mathcal{F}_H^2$. Make queries on sets in $\mathcal{F}_H$ that are independent in H.
7: Call PARA-FIND-ONE-EDGE on all positive samples.

COMPUTATION PHASE

8: For each relevant set χ, divide χ-samples in $\mathcal{F}_H^1$ in $c(\chi)$ groups of $2^{r+2} \ln \frac{1}{\delta'}$ modified $(\chi, \frac{1}{2} p_H^*(\chi))$-samples and the same number of modified $(\chi, \frac{1}{2^{r+3+|\chi|} d_H(\chi)})$-samples.
9: Process the samples in $\mathcal{F}_H^1$ group by group in an arbitrary order. Increase $c(\chi)$ by the number of new edges that χ-samples produce. Add newly found edges to E.
10: Process the samples in $\mathcal{F}_H^2$. Add newly found edges to E.
11: **1-edge-finder**: For any χ-sample $P_\chi \in \mathcal{F}_H^2$, let e be the output of PARA-FIND-ONE-EDGE(P_χ). $\forall v \in e$, make a query on $\chi \cup \{v\}$ to test whether it is an edge. Add newly found edges to E.
12: For every newly found relevant set χ, $c(\chi) \leftarrow 1$.
13: **until** no new edge is found

We remark that we can use original χ-samples and obtain a much simpler algorithm than Algorithm 7. However, the query complexity will be roughly $m^{\Delta+2}$ instead of $m^{1+\frac{\Delta}{2}}$. The reduction of the complexity in the exponent of m is due to the fact that each modified χ-sample only needs to deal with edges that have at least 2 vertices that χ does not have. This leads to the definition of the

new discovery probability as follows.

$$p_H(\chi) = \begin{cases} 1/(2^{r+|\chi|+2}d_H(\chi))^{1/(r-\Delta-|\chi|)}, & \text{if } |\chi| \le r-\Delta-2 \\ 1/(2^{r+|\chi|+2}d_H(\chi))^{1/2}, & \text{otherwise.} \end{cases}$$

We use the new discovery probabilities in Algorithm 7. Although we use modified samples, special care is still needed for 1-edges in order to parallelize the edge finding process. In fact, the majority of effort in developing Algorithm 7 is devoted to dealing with 1-edges.

In both $\mathcal{F}_H^1$ and $\mathcal{F}_H^2$, we draw $(\chi, \frac{1}{2^{r+3+|\chi|}d_H(\chi)})$-samples in addition. The reason for the design will be made clear in the analysis section. A group of χ-samples in $\mathcal{F}_H^1$ will consist of both $(\chi, \frac{1}{2}p_H^*(\chi))$-samples and $(\chi, \frac{1}{2^{r+3+|\chi|}d_H(\chi)})$-samples. $\mathcal{F}_H^2$ contains $(\chi, \frac{1}{4}p_H^*(\chi))$-samples as in Algorithm 6. Although, the number of $(\chi, \frac{1}{4}p_H^*(\chi))$-samples appears to be different from that of Algorithm 6, we remark that $2(4/p_H(\chi))^{r-|\chi|}\ln\frac{1}{\delta'}$ is bounded by $2^{3r+3}d_H(\chi)\ln\frac{1}{\delta'}$ under the definition of discovery probabilities in Section 5 and this group of samples are designed for essentially the same purpose as those for Algorithm 6. We also use a subroutine called *1-edge-finder*, specified in Algorithm 7.

7.2 Analysis

Round complexity

The following two definitions are analogous to those in Section 5. The extra subscript indicates that the new definitions depend on the already found sub-hypergraph H, while the previous definitions don't.

Definition 26 *Let $\rho_{\chi,H}(p)$ be the probability that a $(\chi, \nu_H(\chi), p)$-sample is positive, where χ is a vertex set that does not contain an edge.*

Definition 27 *Let $p_{\chi,H} = \min\left\{p|\rho_{\chi,H}(p) = 1/2^{r+1}\right\}$ be the* threshold probability *of χ.*

Now we bound the number of iterations of Algorithm 7. We divide the process of the algorithm into $(1+\Delta)$ phases, each of which is indexed by a number in $[r-\Delta, r]$. The phase l begins when all edges of size less than l have been found. Phase $r-\Delta$ is the first phase because there is no edge of size less than $r-\Delta$.

Let e be an edge of size l and χ be a known relevant subset of e. We need to deal with two cases : $|\chi| = l-1$ and $|\chi| \le l-2$, the latter of which is simpler as every 1-edge of χ has been discovered. We make the following definition.

Definition 28 *χ is active if it satisfies either of the following two conditions.*

1. *$|\chi| \le l-2$ and $\rho_{\chi,H}(\frac{1}{2}p_H^*(\chi)) \ge 1/2^{r+1}$.*
2. *$|\chi| = l-1$ and $\rho_{\chi,H}(\frac{1}{2}p_H^*(\chi)) \ge 1/2^{r+1}$ and $\rho_{\chi,H}(\frac{1}{2^{r+3+|\chi|}d_H(\chi)}) \ge 1/2^{r+1}$.*

It is inactive otherwise.

The definition is analogous to that in Section 5, and so are the following assertions. The assertions are made at phase l.

Assertion 29 *Consider one group of χ-samples $\mathcal{G}$ in $\mathcal{F}_H^1$. Let H' be the hypergraph the algorithm has found before the group is processed. If χ is active, one of the following three events happens.*

1. $p^*_{H'}(\chi) < \frac{1}{2} p^*_H(\chi)$,
2. $d_{H'}(\chi) > 2d_H(\chi)$, *or*
3. $\mathcal{G}$ *will produce a new edge.*

Assertion 30 *If χ is inactive, at the end of this iteration, e has been found or a subset of e whose threshold probability is at most $\frac{1}{2} p_{\chi,H}$ has been found.*

The two assertions guarantee that Algorithm 7 makes a certain progress at each iteration.

Lemma 31 *If no assertion is violated, phase l terminates in $O(\min(2^r(\log m + r), (\log m + r)^2))$ iterations.*

Proof We need to prove that every edge of size l can be found in the specified number iterations. Let e be an edge of size l. The proof proceeds similarly to that of Lemma 20. We divide the iterations into sub-phases, each of which is associated with a subset of e (we use sub-phases here to avoid confusion). Using an argument similar to that used in the proof of Lemma 20, we can show that each sub-phase takes $O(\log m + r)$ iterations. The only exception is that in this proof, the threshold probability of a set χ might not be fixed (it depends on the already found sub-hypergraph H). When more 1-edges of χ are found, $\rho_{\chi,H}(p)$ will decrease as more vertices are excluded from the sample. Therefore, $p_{\chi,H}$ might increase. After such a sub-phase, the associated threshold probability might not halve. However, this exception only happens when the subset associated with the sub-phase is of size $l-1$ and only happens $l \leq r$ times as there are at most l such subsets and causes at most l additional sub-phases. Therefore, we get the same asymptotic bound on the number of sub-phases, which is $O(\min(2^r, \log m + r))$. This establishes the lemma. ■

Now we show the two assertions are true with high probability.

Lemma 32 *Assertion 29 is true for Algorithm 7 with probability at least $1 - \delta'$.*

Proof $\mathcal{G}$ consists of two subgroups of samples with different sampling probabilities. In the analysis we will only consider one subgroup. In the case that $|\chi| \leq l-2$, we use only $(\chi, \frac{1}{2} p^*_H(\chi))$-samples. In the case that $|\chi| = l-1$, we will use the subgroup with the smaller sampling probability. Let η be the sampling probability of the subgroup we consider. We have $\eta = \frac{1}{2} p^*_H(\chi)$ when $|\chi| \leq l-2$ and $\eta = \min(\frac{1}{2} p^*_H(\chi), \frac{1}{2^{r+3+|\chi|} d_H(\chi)})$ when $|\chi| = l-1$. By our definition of active, in both cases $\rho_{\chi,H}(\eta) \geq 1/2^{r+1}$. The probability that a modified (χ,η)-sample contains an edge in H' is at most

$$\sum_{\chi' \subseteq \chi} d_{H'}(\chi') \cdot \eta^{\max(r-\Delta-|\chi'|,2)} + |\nu_{H'}(\chi) \backslash \nu_H(\chi)| \cdot \eta. \tag{1}$$

- When $|\chi| = l-2$, $|\nu_{H'}(\chi) \backslash \nu_H(\chi)| = 0$. Therefore, if $\eta = \frac{1}{2} p^*_H(\chi) \leq p^*_{H'}(\chi)$, Equation (1) is at most $1/2^{r+2}$.
- When $|\chi| = l-1$, since every 1-edge of χ must contain χ in phase l, Equation (1) is bounded by

  $$\sum_{\chi' \subset \chi} d_{H'}(\chi') \cdot \eta^{\max(r-\Delta-|\chi'|,2)} + d_{H'}(\chi) \cdot \eta.$$

  If $\frac{1}{2} p^*_H(\chi) \leq p^*_{H'}(\chi)$ and $d_{H'}(\chi) \leq 2d_H(\chi)$, the above is bounded by $1/2^{r+2}$.

With probability at least $1/2^{r+2}$, a (χ,η)-sample contains an edge that is not contained in H'. Thus, with probability at least $1-\delta'$, $\mathcal{G}$ will produce a new edge. ■

Lemma 33 *Assertion 30 is true for Algorithm 7 with probability at least* $1-\delta'$.

Proof First we remark that if e has not been found, the probability that e is contained in a modified χ-sample is the same as for an unmodified one. This is because e does not contain any vertex in $\nu_H(\chi)$. Otherwise, e contains an edge in H, which violates our assumption that edges do not contain each other.

If $\rho_{\chi,H}(\frac{1}{2}p^*_H(\chi)) < 1/2^{r+1}$, the proof proceeds similarly to that of Lemma 19. We remark that the differences are that $\rho_{\chi,H}$ and $p_{\chi,H}$ are used instead of ρ_χ and p_χ and we draw more samples in $\mathcal{F}^2_H$ in Algorithm 7.

The remaining case is when $|\chi| = l-1$ and $\rho_{\chi,H}(\frac{1}{2^{r+3+|\chi|}d_H(\chi)}) < 1/2^{r+1}$. Consider a $(\chi, \frac{1}{2^{r+3+|\chi|}d_H(\chi)})$-sample P_χ in $\mathcal{F}^2_H$. Since e is of size l, we have $|e\backslash\chi| = 1$. Let $\{v\} = e\backslash\chi$. We have that

$$Pr[v \in P_\chi] = \frac{1}{2^{r+3+|\chi|}d_H(\chi)}$$

and

$$Pr[\exists \text{ an edge } e' \subseteq P_\chi \text{ such that } v \notin e' \mid v \in P_\chi] \leq \rho_{\chi,H}(\frac{1}{2^{r+3+|\chi|}d_H(\chi)}) < 1/2^{r+1}.$$

Therefore, with probability at least $\frac{1}{2^{r+3+|\chi|}d_H(\chi)} \cdot (1-1/2^{r+1})$, P_χ contains v and contains only edges that are incident with v. Our 1-edge-finder will find e in this case. As we draw $2^{r+4+|\chi|}d_H(\chi)$ $\ln\frac{1}{\delta'}$ samples, e will be found with probability at least $1-\delta'$. ■

Since the algorithm has only $1+\Delta$ phases, the algorithm ends after $O((1+\Delta)\cdot\min(2^r(\log m+r),(\log m+r)^2))$ iterations. If no assertion is violated, the round complexity of Algorithm 7 is

$$O((1+\Delta)\cdot\min(2^r(\log m+r)^2,(\log m+r)^3))$$

We can choose δ' so that the algorithm succeeds with probability $1-\delta$ and $\log\frac{1}{\delta'} \leq poly(r,\log n)\cdot\log\frac{1}{\delta}$.

Query complexity

The main discrepancy of the performance of this algorithm is due to the fact that in $\mathcal{F}^2_H$, the discovery probabilities are chosen as if all the edges were of minimum possible size, while the numbers of samples drawn are chosen as if all the non-edges (or potential edges) of H were of the maximum possible size. This causes the super-linear query complexity. At each iteration, the number of χ-samples in $\mathcal{F}^2_H$ is at most

$$2(4/p_H(\chi))^{r-|\chi|}\ln\frac{1}{\delta'} = \begin{cases} O((2^{O(r)}\cdot d_H(\chi))^{\frac{r-|\chi|}{r-\Delta-|\chi|}}\cdot\log\frac{1}{\delta'}) & \text{if } |\chi| \leq r-\Delta-2 \\ O((2^{O(r)}\cdot d_H(\chi))^{1+\frac{\Delta}{2}}\cdot\log\frac{1}{\delta'}) & \text{otherwise.} \end{cases}$$

Note that $(r-|\chi|)/(r-\Delta-|\chi|)$ is at most $1+\frac{\Delta}{2}$ when $|\chi| \leq r-\Delta-2$. Therefore, the number of modified χ-samples in $\mathcal{F}^2_H$ is at most $O((2^{O(r)}\cdot d_H(\chi))^{1+\frac{\Delta}{2}}\cdot\log\frac{1}{\delta'})$. Because $\sum_\chi d_H(\chi) \leq 2^r m$

and $\forall \chi, d_H(\chi) \leq m$, we have $\sum_\chi d_H(\chi)^{1+\frac{\Delta}{2}} \leq (2^r m)^{1+\frac{\Delta}{2}}$. Therefore, the total number of queries the algorithm makes is bounded by

$$O(2^{O((1+\frac{\Delta}{2})r)} \cdot m^{1+\frac{\Delta}{2}} \cdot poly(\log n, \log \frac{1}{\delta})).$$

This finishes the proof of Theorem 24.

Acknowledgments

The conference version of this paper appears in Angluin and Chen (2005). The authors would like to thank the referees for helpful comments.

References

Noga Alon and Vera Asodi. Learning a hidden subgraph. *SIAM Journal on Discrete Mathematics*, 18(4):697–712, 2005.

Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM Journal of Computing*, 33(2):487–501, 2004.

Dana Angluin and Jiang Chen. Learning a hidden graph using O(log n) queries per edge. In *Conference on Learning Theory*, pages 210–223. Springer, 2004.

Dana Angluin and Jiang Chen. Learning a hidden hypergraph. In *Conference on Learning Theory*, pages 561–575. Springer, 2005.

Richard Beigel, Noga Alon, Simon Kasif, Mehmet Serkan Apaydin, and Lance Fortnow. An optimal procedure for gap closing in whole genome shotgun sequencing. In *RECOMB*, pages 22–30, 2001.

Peter Damaschke. Adaptive versus nonadaptive attribute-efficient learning. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 590–596. ACM Press, 1998.

Vladimir Grebinski and Gregory Kucherov. Reconstructing a Hamiltonian cycle by querying the graph: Application to DNA physical mapping. *Discrete Applied Mathematics*, 88(1-3):147–165, 1998.

Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000.

Journal of Machine Learning Research 7 (2006) 2237-2257 Submitted 10/05; Revised 2/06; Published 10/06

An Efficient Implementation of an Active Set Method for SVMs

Katya Scheinberg KATYAS@US.IBM.COM
Mathematical Science Department
IBM T. J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

We propose an active set algorithm to solve the convex quadratic programming (QP) problem which is the core of the support vector machine (SVM) training. The underlying method is not new and is based on the extensive practice of the Simplex method and its variants for convex quadratic problems. However, its application to large-scale SVM problems is new. Until recently the traditional active set methods were considered impractical for large SVM problems. By adapting the methods to the special structure of SVM problems we were able to produce an efficient implementation. We conduct an extensive study of the behavior of our method and its variations on SVM problems. We present computational results comparing our method with Joachims' SVM^{light} (see Joachims, 1999). The results show that our method has overall better performance on many SVM problems. It seems to have a particularly strong advantage on more difficult problems. In addition this algorithm has better theoretical properties and it naturally extends to the incremental mode. Since the proposed method solves the standard SVM formulation, as does SVM^{light}, the generalization properties of these two approaches are identical and we do not discuss them in the paper.

Keywords: active set methods, support vector machines, quadratic programming

1. Introduction

In this paper we introduce an active set method to solve the following convex quadratic programming (QP) optimization problem which is defined by the classic soft margin SVM formulation (see, for example, Cristianini and Shawe-Taylor, 2000).

$$\begin{aligned} \max \quad & -\frac{1}{2}\alpha^T Q\alpha - c^T\xi \\ & -Q\alpha + by + s - \xi = -e, \\ & 0 \le \alpha \le c,\ s \ge 0,\ \xi \ge 0, \end{aligned} \tag{1}$$

where $\alpha \in \mathbf{R}^n$ is the vector of the dual variables, b is the bias (scalar) and s and ξ are the n-dimensional vectors of the slack and the surplus variables, respectively. y is a vector of the labels, ± 1. Q is the label encoded kernel matrix, $Q_{ij} = y_i y_j K(x_i, x_j)$, e is the vector of all 1's of length n and c is the penalty vector associated with the errors (in standard soft margin SVMs the vector c is a product of vector e and a scalar penalty C, but here we will allow for any nonnegative vector c). The dual of this problem is

$$\min \quad \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

$$\begin{aligned} \text{s.t.} \quad & y^T \alpha = 0, \\ & 0 \leq \alpha \leq c. \end{aligned} \tag{2}$$

To confirm that problem (1) is equivalent to the traditional soft margin SVM formulation

$$\begin{aligned} \min \quad & \frac{1}{2} w^T w + c^T \xi \\ \text{s.t.} \quad & y_i (w^T x_i - b) - s_i + \xi_i \geq 1, \quad i = 1, \ldots, n \\ & s \geq 0, \; \xi \geq 0, \end{aligned} \tag{3}$$

observe that (2) is the same as the dual of (3) and from optimality conditions of (3) and (2) we have $w = \sum_{i=1}^{n} y_i \alpha_i x_i$. Substituting this expression for w to (3) and denoting $Q_{ij} = y_i y_j \alpha_i \alpha_j x_i^T x_j$ (or $Q_{ij} = y_i y_j \alpha_i \alpha_j K(x_i, x_j)$ in the kernel case) we obtain the convex QP formulation (1), which we will consider in this paper. Hence, (1) and soft margin SVM enjoy the *same generalization properties.*

General convex QPs are typically solved by one of the two approaches: *interior point method* approach or *active set method* approach. If the Hessian of an objective function (matrix Q in the case of SVM) and/or the constraint matrix of the QP problem is large and sparse then an interior point method is usually the method of choice. If the problem is of moderate size but the matrices are dense, then active set method is preferable. In SVM problems the Q matrix is typically dense. Thus, large SVM problems present a challenge for both approaches. It was shown by Fine and Scheinberg (2001) and Ferris and Munson (2000) that for some classes of SVMs, for which Q is dense but low-rank, one can adapt an interior point method to work very efficiently. However, if the rank of Q is high, an active set approach seems to remain the only main alternative.

One of the most "traditional" active set methods in the optimization literature is the *Simplex* method for linear programming (LP) problems. The Simplex method is known to have very good practical performance. The QP analogues, though not as extensively tested in practice, are also considered to be very efficient. There are a few methods based on the Simplex method idea for solving QP problems (see Fletcher, 1971; Goldfarb, 1972; Goldfarb and Idnani, 1983). Many of them are theoretically equivalent, meaning that they produce the same sequence of iterations, but they have different numerical properties (such as per-iteration complexity and numerical stability). In this paper we derive an implementation targeted to SVM problems based on the framework described in Fletcher (1971), Goldfarb (1972) and Nocedal and Wright (1999).

The main idea of this method in the context of SVM is to fix, at each iteration, all variables in the current dual active set[1] at their current values (0 or c), and then to solve the reduced dual problem. After obtaining a solution - decide whether it is optimal for the overall dual problem (same as being feasible for the overall primal problem), or if any of the dual variables should be released from the active set.

When applied to SVM, this approach poses the following problem: if the complement of the dual active set (the set of "free" variables) has large cardinality, then solving the restricted subproblems may be too expensive, since Q is completely dense. Also determining the next variable to leave the active set may be expensive for the same reason. Therefore, updating all "free" variables at once was considered impractical.

The most common approach to large SVM problems is to use a restricted active set method, such as chunking (Boser et al., 1992) or decomposition (Osuna et al., 1997; Joachims, 1999) where

1. The dual active set is the set of dual variables α whose values are at their bound.

at each iteration only a small number of variables are allowed to be varied. The size of such "chunk" is determined heuristically or is chosen by the user. There are a few skillfully implemented SVM solvers based on this type of restricted active set methods (Joachims, 1999; Platt, 1999). The main disadvantage of these methods is that they tend to have slow convergence when getting closer to the optimal solution. Moreover, their performance is sensitive to the changes in the chunk size and there is no good way of predicting a good choice for the size of the chunks for a particular problem.[2]

A full active set method, such as the one presented in this paper, avoids these disadvantages. The method itself is not new (see Nocedal and Wright, 1999). Our contribution is to adapt it to the SVM framework and provide an efficient implementation.

First we notice that a support vector that violates the margin constraint (that is the ξ surplus variable is positive) corresponds to a variable α which is at its upper bound and therefore is in the dual active set. The complement of the dual active set contains variables α that are strictly between the upper and lower bounds. Such variables correspond only to the support vectors that are exactly on the margin (that is both the corresponding slack and the surplus variables are zero). The current number of such support vectors, n_s, is the size of the reduced QP (RQP). Solving such RQP directly (say, by an IPM method, as it is done in SVM^{light}) requires at least $O(n_s^3)$ operations, which might be prohibitively expensive if repeated over and over again and if n_s is relatively large. We do not solve RQP directly, but only make one step toward its solution at each iteration. Moreover, the active set is incremented only by one variable at a time (either one variable leaving, or one entering the active set), hence we can store and update a factorization of the reduced matrix Q. Each update takes $O(n_s^2)$ operations and so does solving a system of equations with the reduced matrix Q.

At each such step toward the optimal solution of the RQP, we either find that solution or encounter a bound on one of the "free" variables. In the latter case this variable is included into the active set and the process repeats.

This process does not always produce an optimal solution to the subproblem, but usually produces a good approximation of it. Typically this does not affect the overall number of iterations significantly, whereas the reduction of the per-iteration cost is significant.

The RQP may sometimes have an infinite solution, if reduced matrix Q is singular. In this case an infinite descent direction is computed and a step is taken along this direction until one of the variable bounds is encountered. We provide the full treatment of the various cases for solving the RQP subproblem. We use the approach described in Frangioni (1996).

If the search for the optimum of the RQP subproblem is terminated then our method determines whether the primal feasibility was achieved and if not, which dual variable should leave the active set. To do that we need to compute a product of a submatrix of Q that corresponds to the variables at their upper bounds and the unit vector of an appropriate length. This can be very expensive to compute at each iteration, instead one should rather store and update the result of this multiplication. Another advantage of using "one-variable-at-a-time" increments is in potentially reducing the cost of such updates.

The multiple updates to the active set, which are used in "chunking" and "decomposition" methods could still have an advantage if the overall number of iterations were significantly smaller than in the case of single updates. But as our computational results indicate this is not the case. We offer some intuition to support this claim. Assume that your data contains 10 identical data points which at the current iteration are the most violated examples and we would like to introduce them into the

2. See Section 12.1.1 in Platt (1999) for a similar discussion which motivated Platt's SMO. Essentially SMO is an active set method in which the chunk size is fixed to be the smallest possible, namely 2.

next "chunk". Introducing all 10 at once implies 10 times more work than introducing just one. Yet since they are identical, introducing just one produces the same result as introducing all ten. Since the training data is often somewhat repetitive (there may not be *identical* points, but rather very similar points, for instance, in clustered data sets) this example is not too far fetched.

As the computational results show, our method has particular advantage over SVM^{light} on problems where the number of the support vectors or the number of outliers is large (but not necessarily excessive, such as ~ 1000 out of 20000 vectors). Our algorithm currently requires the storage of the Cholesky factors of the reduced matrix Q, which might require excessive amount of memory for problems where the number of unbounded support vectors is very large. However, this often means that the chosen kernel suffers from overfitting the data, so the problem is badly posed in some sense, unless the entire test set is very large, in which case one should consider a different implementation, and, possibly, a more powerful computer.

The most expensive step of our algorithm (and of SVM^{light}, in fact) is pricing the primal constraints and choosing the next constraint to enter the active set. We will compare two approaches. One of these approaches is *shrinking*, which is used by SVM^{light}, and the other is *sprint* which is an industry standard in advanced implementations of LP solvers (Bixby et al., 1992). We observe that *sprint* appears to work better than *shrinking* on difficult SVM problems.

The proposed method enjoys several theoretical advantages compared to the methods based on *chunking*. First of all it converges in a finite number of iterations (Fletcher, 1971; Frangioni, 1996). In the worst case this number might be exponential, but it is hardly the case in practice. The method is also well suited for analysis of various situations. For instance, in Balcazar et al. (2001) a randomized active set algorithm for SVM is introduced and shown to have a quasi-linear average complexity. Our algorithm can be easily adapted to fit the randomized framework of Balcazar et al. (2001), hence similar average case results apply.

Recently, active set methods for SVM similar to ours were used in Cauwenberghs and Poggio (2001) for incremental learning and in Hastie et al. (2004) for generating the entire regularization path. Their methods, unlike ours, require primal and dual feasibility to be satisfied at every iteration and progress by changing the optimization problem itself (in a manner dictated by the respective uses of their methods). However, many of the efficiency issues of the algorithms are similar, such as the possible singularity of the reduced matrix Q and efficient updates of its Cholesky factorization. Though we choose to focus on one specific active set method in this paper, we believe the the experience we present here will be useful for other active set methods for SVM problems. Some of the ideas presented in the paper to improve the efficiency of the active set methods for SVMs were also suggested in Kaufman (1998).

The paper is organized as follows. In the next section we introduce the dual active set method for the soft-margin SVM problem and describe the details of solving the reduced QP problem. In Section 3 we will present the results of comparing our method to SVM^{light} on a selection of classification problems from the UCI repository (Blake and Merz, 1998). In Section 4 we will focus on various implementational issues that arise in the attempt to improve the performance of the method. In Subsection 4.4 we apply our method to the incremental case. Section 5 contains some conclusions.

2. Dual Active Set Method for SVMs

Any optimal solution to problems (1) or (2) must satisfy the Karush-Kuhn-Tucker (KKT) necessary and sufficient optimality conditions:

$$
\begin{aligned}
&\mathbf{1} \quad \alpha_i s_i = 0, \quad i = 1,\ldots,n \\
&\mathbf{2} \quad (c_i - \alpha_i)\xi_i = 0, \quad i = 1,\ldots,n \\
&\mathbf{3} \quad y^T \alpha = 0, \\
&\mathbf{4} \quad -Q\alpha + by + s - \xi = -e, \\
&\mathbf{5} \quad 0 \le \alpha \le c, \\
&\mathbf{6} \quad s \ge 0,\ \xi \ge 0.
\end{aligned}
$$

Let us introduce some notation. A primal-dual solution (α, b, s, ξ) is called *dual basic feasible* if it satisfies condition **1-5** of the KKT system, but may violate condition **6**. For a given dual basic feasible solution, (α, b, s, ξ), we partition the index set $I = \{1,\ldots,n\}$ into three sets I_0, I_c and I_s in the following way: $\forall i \in I_0\ s_i \ge 0$ and $\alpha_i = 0$, $\forall i \in I_c\ \xi_i \ge 0$ and $\alpha_i = c_i$ and $\forall i \in I_s\ s_i = \xi_i = 0$ and $0 < \alpha_i < c_i$. It is easy to see that $I_0 \cup I_c \cup I_s = I$ and $I_0 \cap I_c = I_c \cap I_s = I_0 \cap I_s = \emptyset$. We will refer to I_s as the *primal active set* and to $I_0 \cup I_c$ as the *dual active set.* Let $n_s = |I_s|$, $n_0 = |I_0|$ and $n_c = |I_0|$,

Based on the partition (I_0, I_c, I_s) we define Q_{ss} (Q_{cs} Q_{sc} Q_{cc}, Q_{0s}, Q_{00}) as the submatrix of Q whose columns are the columns of Q indexed by the set I_s (I_c, I_s, I_c, I_0, I_0) and whose rows are the rows of Q indexed by I_s (I_s, I_c, I_c, I_s, I_0). We also define y_s (y_c, y_0) and α_s (α_c, α_0) and the subvectors of y and α whose entries are indexed by I_s (I_c, I_0). c_c is the part of vector c indexed by I_c and by e we denote a vector of all ones whose size is determined by the context.

To initiate the algorithm we assume that we have a dual basic feasible solution α^0, b, s^0, ξ^0 and the corresponding partition $(I_0{}^0, I_c{}^0, I_s{}^0)$. For example setting $\alpha^0 = 0$ and $I_0 = \{1,\ldots,n\}$ produces a starting point for the algorithm.

We know that $\forall i \in I_0\ \alpha_i = 0$ and $\forall i \in I_c\ \alpha_i = c_i$. Then if we fix the variables in the dual active set then our dual problem reduces to

$$
\begin{aligned}
\min_{\alpha_s} \quad & \frac{1}{2}\alpha_s^T Q_{ss}\alpha_s + c_c^T Q_{cs}\alpha_s - e^T\alpha_s \\
\text{s.t.} \quad & y_s^T\alpha_s = -y_c^T c_c, \\
& 0 \le \alpha_s \le c.
\end{aligned}
$$

The outline of the algorithm is the following:

Step 0 Given α^0, β^0, s^0, ξ^0 find initial I_s, I_0 and I_c.

Step 1 If $I_s = \emptyset$, go to Step 2, otherwise:

(i) Solve

$$
\begin{aligned}
\min_{\alpha_s} \quad & \frac{1}{2}\alpha_s^T Q_{ss}\alpha_s + c^T Q_{cs}\alpha_s - e^T\alpha_s \\
\text{s.t.} \quad & y_s^T\alpha_s = -y_c^T c_c.
\end{aligned} \tag{4}
$$

If a finite solution, α_s^*, exists, then set $d = \alpha_s^* - \alpha_s$, otherwise find d - an infinite descent direction.

(ii) From the current iterate make a step along direction d until for some $i \in I_s$ $\alpha_i = 0$ or $\alpha_i = c_i$ or until solution is reached. α_s^{k+1} is the new point.

(iii) If for some $i \in I_s$, $\alpha_i^{k+1} = 0$,
then update $I_s := I_s \backslash \{i\}$, $I_0 := I_0 \cup \{i\}$, $k := k+1$ and go to the beginning of **Step 1**.

(iv) If for some $i \in I_s$, $\alpha_i^{k+1} = c_i$,
then update $I_s := I_s \backslash \{i\}$, $I_c := I_c \cup \{i\}$, $k := k+1$ and go to the beginning of **Step 1**.

(v) If the optimum is reached in step (ii); that is $\alpha_s^{k+1} = \alpha_s^*$, proceed to **Step 2**.

Step 2 Partition I_0 into I_0' and I_0'' and partition I_c into I_c' and I_c''

(i) Compute s_0', the subvector of s indexed by I_0':

$$s_0' = -Q_{0s}'\alpha_s^{k+1} - y_0'\beta + 1 - Q_{0c}'c_c$$

and ξ_c', the subvector of ξ indexed by I_c':

$$\xi_c' = Q_{cs}'\alpha_s^{k+1} + y_c'\beta - 1 + Q_{cc}'c_c,$$

where Q_{0s}' and Q_{0c}' (Q_{cs}' and Q_{cc}') are the submatrices of Q_{0s} and Q_{0c}, respectively, (Q_{cs} and Q_{cc}, respectively) with rows index by $I_0{}'$ ($I_c{}'$).

(ii) Find $i_0 = \text{argmin}_i\{s_i : i \in I_0{}'\}$.
Find $i_c = \text{argmin}_i\{\xi_i : i \in I_c{}'\}$.

(iii) If $s_{i_0} \geq 0$ and $\xi_{i_c} \geq 0$ then if $I_0{}' \neq I_0$ or $I_c{}' \neq I_c$ then let $I_0{}' := I_0$ and $I_c{}' := I_c$ and go to Step 2(i). Else, the current solution is optimal, **Exit**.
If $s_{i_0} \leq \xi_{i_c}$, then $I_s := I_s \cup \{i_0\}$ and $I_0 := I_0 \backslash \{i_0\}$.
Else, $I_s := I_s \cup \{i_c\}$ and $I_c := I_c \backslash \{i_c\}$.
$k := k+1$, go to **Step 1**.

We will now discuss in details the implementation of the steps of the algorithm.

2.1 Solving the Quadratic Subproblem

When matrix Q_{ss} is strictly positive definite then problem (4) has a unique finite solution. This solution satisfies the KKT conditions:

$$\begin{aligned} -Q_{ss}\alpha_s + y_s^T\beta &= -e^T + c_c^T Q_{cs}\alpha_s \\ y_s^T\alpha_s &= -c_c^T y_c, \end{aligned}$$

or, in matrix form,

$$\begin{bmatrix} -Q_{ss} & y_s \\ y_s^T & 0 \end{bmatrix} \begin{pmatrix} \alpha_s \\ \beta \end{pmatrix} = \begin{pmatrix} -e + Q_{sc}c_c \\ -c_c^T y_c \end{pmatrix}. \tag{5}$$

Since we are considering the case when Q_{ss} is nonsingular, we can find β by taking the Schur complement of the above system

$$(y_s^T Q_{ss}^{-1} y_s)\beta = y^T Q_{ss}^{-1}(-e + Q_{sc}c_c) - c_c^T y_c.$$

Consider the Cholesky factorization $Q_{ss} = L_s L_s^T$ and denote $L_s^{-1} y_s$ by r_1 and $L_s^{-1}(-e + c_c^T Q_{cs})$ by r_2. Then the solution to (5) is

$$\beta = \frac{r_1^T r_2 - c_c^T y_c}{r_1^T r_1}, \quad \alpha_s = L_s^{-T}(r_1 \beta - r_2).$$

It is often the case, however, that Q_{ss} is not strictly positive definite. This can even occur when an RBF kernel (which is strictly positive definite for distinct data points) is used, if the set I_s contains indices of two identical data points with different labels.

If, due to singularity of Q_{ss}, system (5) is underdetermined, this means that problem (4) has an unbounded solution. In this case Step 1(i) should produce an infinite descent direction for (4). A direction d is an infinite direction if it satisfies $Q_{ss} d = 0$ and $y_s^T d = 0$. Depending on the sign of $(-e + c e^T Q_{cs})^T d$ either d or $-d$ is chosen as the infinite *descent* direction. Variable β remains unchanged in this case. We use the approach for positive semidefinite QP problems described in Frangioni (1996) and Kiwiel (1989).

We consider several cases.

Case 1

Let Q_{ss} have only one zero eigenvalue. Then, subject to permutation and without loss of generality, its Cholesky factorization can be written as

$$Q_{ss} = \begin{bmatrix} L_s & 0 \\ l_s^T & 0 \end{bmatrix} \begin{bmatrix} L_s^T & l_s \\ 0 & 0 \end{bmatrix},$$

where $L_s \in \mathbf{R}^{(n_s-1)\times(n_s-1)}$ and $L_s \in \mathbf{R}^{n_s-1}$. Then system (5) can be written as

$$\begin{bmatrix} -L_s L_s^T & -L_s l_s & y_{1:n_s-1} \\ -l_s^T L_s^T & -l_s^T l_s & y_{n_s} \\ y_{1:n_s-1} & y_{n_s} & 0 \end{bmatrix} \begin{pmatrix} \alpha_{1:n_s-1} \\ \alpha_{n_s} \\ \beta \end{pmatrix} = \begin{pmatrix} (-e + Q_{sc} c_c)_{1:n_s-1} \\ (-e + Q_{sc} c_c)_{n_s} \\ -c_c^T y_c \end{pmatrix},$$

where, following Matlab notation, $y_{1:n_s-1}$ ($\alpha_{1:n_s-1}$, $(-e + Q_{sc} c_c)_{1:n_s-1}$) denote the first $n_s - 1$ elements of vector y (α, $(-e + c Q_{sc} e)$) and y_{n_s} (α_{n_s}, $(-e + Q_{sc} c_c)_{n_s}$) denotes the last component of this vector.

Let $r_1 = L_s^{-1} y_{1:n_s-1}$ and $r_2 = L_s^{-1}(-e + Q_{sc} c_c)_{1:n_s-1}$. By expressing $\alpha_{1:n_s-1}$ in the above system through α_{n_s} and β, and by consecutively eliminating α_{n_s} we obtain

$$(-l_s^T r_1 + y_{n_s})\beta = (-e + Q_{sc} c_c)_{n_s} - l_s^T r_2.$$

We now have two cases.

(a) If $l_s^T r_1 \neq y_{n_s}$ then system (5) still has a unique solution

$$\begin{aligned} \beta &= \frac{(-e + Q_{sc} c_c)_{n_s} - l_s^T r_2}{-l_s^T r_1 + y_{n_s}}, \\ \alpha_{n_s} &= \frac{c_c^T y_c + r_1^T r_2 - \beta r_1^T r_1}{-r_1^T l_s + y_{n_s}} \\ \alpha_s &= L_s^{-T}(-l_s \alpha_{n_s} + r_1 \beta - r_2). \end{aligned}$$

(b) If $l_s^T r_1 + y_{n_s} = 0$ then system (5) is singular, hence we are looking for an infinite direction d. $d_s = ((L_s{}^{-1} l_s)^T, -1)^T$ is such a direction. It can be easily shown that $Q_{ss} d_s = 0$ from the form of the factorization of Q_{ss}, and it can be easily shown that $y_s^T d_s = 0$ from the fact that $l_s^T r_1 + y_{n_s} = 0$.

Case 2

Let us now consider the case when Q_{ss} has exactly two zero eigenvalues. Then, again w.l.o.g., we can write its Cholesky factorization as

$$Q_{ss} = \begin{bmatrix} L_s & 0 & 0 \\ l_{s_1}^T & 0 & 0 \\ l_{s_2}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} L_s{}^T & l_{s_1} & l_{s_2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where $L_s \in \mathbf{R}^{n_s-2 \times n_s-2}$, $l_{s_1}, l_{s_2} \in \mathbf{R}^{n_s-2}$. The system (5) is always underdetermined in this case, hence an infinite direction always exists. There are, again, two possible cases.

(a) If $l_{s_2}^T r_1 \neq y_{n_s}$ then the following direction

$$d = (L_s{}^{-T}(l_{s_1} - \rho l_{s_2}), -1, \rho), \text{ where } \rho = \frac{y_{n_s-1} - l_{s_1}^T r_1}{y_{n_s} - l_{s_2}^T r_1}$$

is an infinite direction. $Q_{ss} d = 0$ follows from the form of the Cholesky factorization and $y^T d = 0$ is also easily shown by substitution.

(b) If $l_{s_2}^T r_1 = y_{n_s}$ then

$$d = (L_s{}^{-T} l_{s_2}, 0, -1)$$

is an infinite direction. This case can be shown similarly to Case 1(b).

Case 3

Finally, let us consider the case when Q_{ss} has more than two zero eigenvalues. First, we observe that this case can only happen in the early stage of the algorithm. Whenever Q_{ss} has more than one zero eigenvalue, then system (5) is underdetermined and an infinite direction is found during Step 1(i). Hence, during Step 1(ii) a boundary is always encountered. This means that the set I_s gets reduced by one element and the number of zero eigenvalues of Q_{ss} may only decrease or remain the same. Step 1 repeats until Q_{ss} has at most one zero eigenvalue. Hence, the only way that Q_{ss} may have more than two zero eigenvalues is if a starting solution with such Q_{ss} matrix is given to the algorithm. Such case arises when a warm start is used to initiate the algorithm, as described in Subsection 4.3, therefore, we consider this case here. Let $k > 2$ be the number of zero eigenvalues of Q_{ss}; as before we write, w.l.o.g., the factorization of Q_{ss}:

$$Q_{ss} = \begin{bmatrix} L_s & 0 & 0 & 0 \\ l_{s_1}^T & 0 & 0 & 0 \\ l_{s_2}^T & 0 & 0 & 0 \\ H_s & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} L_s{}^T & l_{s_1} & l_{s_2} & H_s^T \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

where $L_s \in \mathbf{R}^{n_s-k\times n_s-k}$, $l_{s_1}, l_{s_2} \in \mathbf{R}^{n_s-k}$ and $H_s \in \mathbf{R}^{n_s-k\times k-2}$. We generate the infinite direction for the first n_s-k+2 variables exactly as it is done in Case 2 and we do not change the last $k-2$ variables. During each application of Case 3 of Step 1 we reduce I_s by one elements until Q_{ss} has at most 1 nonzero eigenvalue, and Case 3 does not arise again for that problem.

2.2 Rank-one Updates to Q_{ss}

On each iteration of the algorithm the set I_s can decrease by one element only and/or increase by one element only. Hence, from each iteration to the next, Q_{ss} changes by an addition and/or a deletion of one row and column. Instead of recomputing the Cholesky factorization each time, which would require $O(n_s^3)$ operations, it is more efficient to keep the Cholesky factorization of Q_{ss} and update it accordingly when a row and a column are added to or deleted from Q_{ss}. Each such update requires only $O(n_s^2)$ operations. These updates can be found in Golub and Van Loan (1996), but we present them here for completeness.

Increasing I_s. Assume first that Q_{ss} is nonsingular and $Q_{ss} = L_s {L_s}^T$ is its Cholesky factorization. Let $q_s \in \mathbf{R}^{n_s+1}$ be the new row (column) that is added to Q_{ss}. Aside from possible numerical issues, which we discuss later, q_s can be added as the last row and column of Q_{ss}. Then the Cholesky factorization of the new matrix is

$$\begin{bmatrix} L_s & {L_s}^{-1}(q_s)_{1:n_s} \\ 0 & (q_s)^2_{n_s+1} - (q_s)^T_{1:n_s} {L_s}^{-T} {L_s}^{-1} (q_s)_{1:n_s} \end{bmatrix},$$

where $(q_s)_{1:n_s}$ are the first n_s components of the vector q_s and $(q_s)_{n_s+1}$ is its last component. It is easy to see that obtaining the new factorization requires $O(n_s^2)$ operations.

If Q_{ss} is singular, then from the discussion in Case 3 of the previous subsection, it can only have one nonzero eigenvalue, since I_s is increased and hence Step 2 was performed. In this case we permute the rows and columns of Q_{ss} so that the dependent column and row are at the end of Q_{ss} and inserted column and row are placed in the one before last positions. The the last two rows of Cholesky factorization may need to be updated in a similar manner to above, however the total work is still $O(n_s^2)$. In case when Q_{ss} is nonsingular, but nearly so, it is sometimes important for numerical stability to use pivoting during its Cholesky factorization procedure (Fine and Scheinberg, 2001). In such a case refactorization of several rows of L_s might be required even if only one row and column are added to Q_{ss}. However, we did not encounter such situations in our computational experiments.

Decreasing I_s. When I_s is decreased by one element, then a row and a column are removed from Q_{ss} which corresponds to removing a row from the Cholesky factor L_s. If, say, k-th row was removed from L_s then it is no longer lower triangular. In fact it is nearly lower triangular, except for the elements in positions $(j, j+1)$ for $j = k+1, \ldots, n_s-1$. To zero out these elements we apply Givens rotations (Golub and Van Loan, 1996) to the new matrix L_s; in other words we multiply L_s on the right by orthogonal matrices of the form

$$\begin{bmatrix} 1 & \cdots & 0 & 0 & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & c & -s & 0 \\ 0 & 0 & s & c & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}.$$

Each such matrix multiplication takes $O(n_s)$ operations and zeros out one off-diagonal elements, hence we need $O(n_s - k)$ such multiplications, which results in the total work of $O(n_s^2)$ to update the Cholesky factorization of Q_{ss} when an elements is removed from I_s.

Remark 1 *In Frangioni (1996) there are efficient updates for the vectors r_1 and r_2, that we introduced in Subsection 2.1. These vectors are results of backsolves with the Cholesky factors of Q_{ss} and given right hand side vectors. In the case of Frangioni (1996) the right hand side vectors remain the same throughout the algorithm and only the Cholesky factors change. In our case this is true only for r_1 but not for r_2, which changes each time the set I_c changes. These updates can also improve the efficiency of the algorithm when these backsolves have a noticeable contribution to the overall workload of the algorithms. Since this does not occur very frequently we do not get into further details in this paper.*

Remark 2 *If n_s is very large and is comparable to n then even storing and updating the Cholesky factors of Q_{ss} become too expensive compared to solving the entire problem. Our method is not practical on such problems. However, it is questionable whether such problems should ever be solved, since the resulting classifiers is most likely overfitting the data and its generalization properties are expected to be very poor.* [3]

Updating I_c Finally we discuss a trivial but useful updates to $Q_{sc}c_c$, $Q_{0c}c_c$ and $Q_{cc}c_c$ when the set I_c is either increased or decreased by one element. We maintain vector $Q_c c_c$ throughout the algorithm, when index i is added to I_c then a c_i multiple of the i-th column of Q is added to $Q_c c_c$. If index i is removed from I_c, then such a vector is subtracted from $Q_c c_c$.

3. Comparison to SVM^{light}

In this section we compare our implementation of the proposed algorithm, which we call SVM-QP, to SVM^{light}. SVM-QP currently is implemented in Fortran 77, although a C++ version is under development. SVM-QP is an open source software and is available from the *www.coin-or.org* website. We used a high-end IBM RS/6000 workstation in our experiments. We made the same amount of memory available to both methods. Just as in SVM^{light} the sparsity of the examples is exploited by SVM-QP during the kernel evaluations. Unlike SMO (Platt, 1999) there is no special handling for the case of linear kernel.

We used the following data sets in our experiments:

- Letter-G: The Letter Image Recognition data set from the UCI Repository (Blake and Merz, 1998) - A large number of black-and-white character images were randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. We examined performances on an arbitrary binary classification problem which was set to separate the letter "G" from all the other letters.

3. See for example Cristianini and Shawe-Taylor (2000), Theorem 4.25, for the generalization power of compression schemes, and the discussion right after and in Chapter 6.

- OCR: USPS (United States Postal Service) data set of hand written digits. This data set comprises 7291 training and 2007 test patterns, represented as 257 dimensional vectors with entries between 0 and 255. T0(T9) stand for the binary classification problem in which the target is the digit 0(9) vis. the all the other digits.

- Web and Adult[4] : We used the tasks that was compiled by Platt and available from the SMO home page[5]

  – Adult - The goal is to predict whether a household has an income greater than $50000. After discretization of the continuous attributes, there are 123 binary features, with ≈ 14 non-zeros per example.

  – Web - A text classification problem with binary representation based on 300 keyword features. This representation is extremely sparse. On the average there are only ≈ 12 non-zero features per example.

  For both problems we chose the test cases with half of the overall available example. We did so to enable to complete many computational tests in a reasonable amount of time. We also present a table with the results of comparing only the runtime of SVM-QP and SVM^{light} on the full test sets for these two problems.

- Abalone: The Abalone data set from the UCI Repository (Blake and Merz, 1998). Since, we were not interested in evaluating generalization performances, we fed the training algorithm with increasing subsets up to the whole set (of size 4177). The gender encoding (male/female/infant) was mapped into $\{$(1,0,0),(0,1,0),(0,0,1)$\}$. Then data was scaled to lie in the [-1,1] interval.

- Spam: This is another data set from the UCI Repository. It was created by M. Hopkins, E. Reeber, G. Forman and J. Suermondt of Hewlett-Packard Labs. It contains 4601 examples of emails roughly 39% of which are classified as spam. There are 57 attributes for each example, most of which represent how frequently certain words or characters appear in the email.

For each data set we used a selection of kernels and parameters to demonstrate how the performance of the methods is affected by n_s - the number of support vectors at the margin, and n_c - the number of support vectors at the upper bound. For the same reason we use various values of C. We use RBF kernel with parameter σ. We also use the linear kernel for a Letter-G and Spam problems and polynomial kernel of degree 5 for the Abalone data set. In the tables of results we indicate the kernel and the value of C in the name of the test case. For instance **web_100_10** stands for the web data set with parameter $\sigma = 100$ and $C = 10$. Name **letter_lin_100** stands for the Letter-G set with linear kernel and $C = 100$, finally **abalone_p5_100** stands for the Abalone set with polynomial kernel of degree 5 and $C = 100$.

We provide two columns of CPU times for SVM^{light}. The first one, SVM^{light}, contains the time of the runs with default accuracy 10^{-3}. The second column, SVM$^{light}_{\varepsilon}$ contains the CPU time of the runs with the accuracy set to 10^{-6} which is the accuracy of SVM-QP. Both algorithms apply the accuracy tolerance to the constraints $-Q\alpha + by + s - \xi = -e$. Specifically, SVM-QP applies a given tolerance ε on **Step 2**(iii) of the algorithm (see Section 2) to determine if $s_{i_0} \geq -\varepsilon$ and $\xi_{i_c} \geq \varepsilon$.

4. Original data set is from the UCI Repository (Blake and Merz, 1998).
5. http://www.research.microsoft.com/~jplatt/smo.html

Name	n	k	n_s	n_c	SVM^{light}	SVM$_e^{light}$	SVM-QP
web_100_100	24692	300	980	453	380	918	65
web_40_10	24692	300	1037	568	241	377	68
web_40_100	24692	300	1214	313	368	685	84
web_100_10	24692	300	679	835	203	358	40
letter_100_100	20000	16	241	39	19	26	3
letter_40_1	20000	16	250	266	6	7	5
letter_40 _100	20000	16	346	8	11	16	4
letter_100_10	20000	16	193	146	10	15	4
letter_40_10	20000	16	320	57	8	10	4
letter_lin_100	20000	16	17	1056	1052	1190	35
ocr9_256_100	7291	256	378	0	13	13	5
ocr0_256_100	7291	256	309	0	8	9	4
abalone_4_100	4177	10	64	1863	135	198	5
abalone_p5_100	4177	10	304	1520	-	-	22
spam_300_100	4601	57	1417	181	90	-	64
spam_lin_100	4601	57	58	822	-	-	11
adult_100_1	16100	123	97	5996	153	154	81
adult_100_100	16100	123	871	4823	515	856	175
adult_200_1	16100	123	168	5785	159	152	85
adult_200_100	16100	123	483	5219	332	447	140
adult_50_10	16100	123	615	5143	207	243	120

Table 1: Performance comparison of SVM-QP and SVM^{light}.

We chose CPU time (in seconds) as the most reasonable performance measure in our setting. The "-" in the table indicates the failure of SVM^{light} on that problem.

Table 1 contains the results for the test problems that we examine in this paper. As we can see, SVM-QP is faster than even the lower accuracy SVM^{light}, on all of the problems. It is faster by at least a factor of 2 on almost all of the problems and by a factor of 5 or more on a few problems.

In Table 2 we present the comparison of SVM-QP and SVM^{light} on the full test sets for **adult** and **web**. These results were obtained by Hans Mittleman, at Arizona State University using a high-end Unix workstation. The "*" in the last column of the last row indicates that SVM-QP ran out of memory, since it was trying to store an $n_s \times n_s$ matrix, with $n_s \approx 10000$. The stopping tolerance was set to be 10^{-6} for both codes, but it is interesting to note that the resulting support vector sets differed significantly. For example, the number of active support vectors for **web_10_100** reported by SVM-QP was 3446, while the same number reported by SVM^{light} was 4025. This discrepancy is due to the fact that SVM^{light} converges to the optimal active set asymptotically, while SVM-QP steps from one feasible active set to another in an "exact" manner, until the optimal is found.

4. Implementation Issues

Now we will discuss some implementation choices.

Name	n	k	n_s	n_c	SVM$_e^{light}$	SVM-QP
web_1000_100	49749	300	297	1702	694	92
web_100_100	49749	300	1404	905	3581	174
web_10_100	49749	300	3446	527	1354	715
adult_1000_100	32561	123	143	11361	937	278
adult_100_100	32561	123	1317	9879	5685	460
adult_10_100	32561	123	9959	3200	14466	*

Table 2: Performance comparison of SVM-QP and SVM^{light} on large data sets.

4.1 Selecting the Incoming Element of I_s

In this subsection we discuss the implementation of Step 2(i). First of all we note that the computational cost of Step 2(i) depends on whether the kernel values are available in the memory or have to be computed. We need $O(|I_s|(|I_0'|+|I_c'|))$ kernel values at each iteration when Step 2 is invoked. Specifically we need the elements of matrix Q whose column indices are in I_s and whose row indices are in I_c' and I_0'.

We note that we *always* store the $n_s \times n_s$ matrix Q_{ss}. This can be a problems when n_s is large. Our algorithm requires storage of the Cholesky factor of Q_{ss}, hence even if we do not store Q_{ss} itself, the storage requirement can be reduced at most by half. In our experiments the size of Q_{ss} and its Cholesky factor was reasonable. For extremely large problems a different implementation may be necessary which solves the linear system in Step 1 by an iterative solver.

To reduce the computational cost it is best to be able to store the entire Q_s matrix (that is the submatrix of Q whose column indices are in I_s). In some cases this might be prohibitively expensive in terms of memory. In our experiments we were able to store Q_s in the space not exceeding 400MB. At the end of this subsection we will discuss the memory saving version of our code.

Let us assume for now that matrix Q_s is available. We will consider various ways of reducing the number of elements in I_0' and I_c' at each repetition of Step 2(i). One simple way to achieve this is to compute the elements of s_0' and ξ_c' until a negative element is encountered, hence, not looking for the maximum violation, but for any violation. This may reduce the per-iteration time, but greatly increases the number of iterations, as has been shown by the extensive practice of the Simplex method in linear programming (Vanderbei, 2001). We will demonstrate this in the section on the incremental mode, since the incremental mode lacks the ability to "look ahead" and select the maximum violated constraint. We conclude that it is important to select the most negative or nearly the most negative element of s_0' and ξ_c' during Step 2.

We use the following concepts, common in LP literature. The primal slack and surplus variables s_i and ξ_i are the reduced costs of the associated dual variable α_i, whose value is currently at a bound. Computing the values of the reduced costs (recall that for each i only one of the reduced costs is not equal to zero) is called *pricing* of the appropriate dual variable. Hence it is important to price all variables with indices in $I_0{}'$ and $I_c{}'$ and maintain these sets in such a way that they contain indices of substantially negative reduced costs.

The efficiency of the large-scale SVM training relies heavily on the fact that at the optimal solution the cardinality of I_s is often much smaller than the total number of data points n. Hence, the cardinalities of I_0 and, possibly, I_c are expected to be large in comparison to I_s. If in Step 2(i) $I_0{}'$ and

$I_c{}'$ are large, while I_s is not very small, then the complexity of this step, which is $O(|I_s|(|I_0'|+|I_c'|))$, might become too high.

Let us assume for a moment that we know some of the indices that at optimality belong to I_c and I_0. Then we can place these indices in I_0'' and I_c'' at the beginning of each Step 2. This can result in substantial savings in the run time, since Step 2(i) requires $O(|I_s|(|I_0'|+|I_c'|))$ operations and $|I_0'| = |I_0| - |I_0{}''|$ and $|I_c'| = |I_c| - |I_c{}''|$. When all the reduced costs of variables whose indices are in $I_0{}'$ and $I_c{}'$ are nonnegative, then so are the reduced costs of variables whose indices are in $I_0{}''$ and $I_c{}''$, due to our assumption about these two subsets.

Naturally, we usually do not know which indices will be in I_0 and I_c at optimality, however, to reduce the workload at each iteration we try to *guess* which indices are the most likely ones to end up in I_0 and I_c at optimality. We place such indices in $I_0{}''$ and $I_c{}''$ sets, respectively. If we guess well, then after all the reduced costs for $I_0{}'$ and $I_c{}'$ become nonnegative, hopefully, only a few reduced costs for $I_0{}''$ and $I_c{}''$ are negative. Here we see a trade-off: if we select $I_0{}''$ and $I_c{}''$ too small, then the computational saving is insignificant, and if we select $I_0{}''$ and $I_c{}''$ too large, then some of large negative reduced costs might be missed and the overall number of iterations might increase. Moreover, once all the dual variables with indices in $I_0{}'$ and $I_c{}'$ are priced, then we have to price all variables with indices in $I_0{}''$ and $I_c{}''$, which are large. So it is important to choose $I_0{}''$ and $I_c{}''$ in such a way that pricing the variables in $I_0{}''$ and $I_c{}''$ does not occur too many times.

We will describe two possible strategies for maintaining sets $I_0{}'$, $I_c{}'$, $I_0{}''$ and $I_c{}''$. One strategy is very simple and is called *shrinking* in SVM literature (Joachims, 1999). At each iteration an index is placed in $I_0{}''$ or $I_c{}''$ if its appropriate reduced cost remained nonnegative for a given number of consecutive iterations (say 100). According to this strategy the sets $I_0{}'$ and $I_c{}'$ are large during the earlier iterations and become gradually smaller during the course of the algorithm. This nicely correlates with the fact that the size of I_s is very small in the earlier iterations I_s gets gradually larger during the course of the algorithm. It is often the case that maximum of $|I_s|(|I_0'|+|I_c'|)$ over all iteration is 3 or 4 times smaller than $\max\{|I_s|\} \times \max\{(|I_0'|+|I_c'|)\}$. At the end one still has to price all the dual variables for $I_0{}''$ and $I_c{}''$, but only a few of such iterations are usually needed.

The second strategy is called *sprint* in Linear Programming literature and was introduced by Forrest (1989). Sprint (sometimes also called *sifting*) has been proven to be very effective in practice for problems that contain large number of inactive constraints (see Bixby et al., 1992). Following the sprint strategy we select a relatively small subset of dual variables with the smallest (including the most negative) reduced costs and we form $I_0{}'$ and $I_c{}'$ from the indices of those variables. Once the problems was solved for $I_0{}'$ and $I_c{}'$ the remaining constraints are priced again and the next relatively small sets of candidates are selected. Pricing all remaining variables and choosing the next small subset is called a *major iteration.* According to this strategy $I_0{}'$ and $I_c{}'$ are always kept small, but the sets $I_0{}''$ and $I_c{}''$ have to be considered regularly throughout the algorithm. As long as the ratio of major iterations to the number of "cheap" iterations is small, the implementation will be efficient.

Table 3 below shows that *sprint* outperforms *shrinking* in most cases, especially on larger, more difficult problems.

4.2 Memory Saving Version

We now discuss the memory saving version. SVM^{light} has an elegant scheme, where the kernel values are stored in cache according to their most recent usage. The size of the cache is dictated by

Name	n	k	n_s	n_c	SVM-QP$_{shr}$	SVM-QP
web_100_100	24692	300	980	453	537	65
web_40_10	24692	300	1037	568	281	68
web_40_100	24692	300	1214	313	416	84
web_100_10	24692	300	679	835	124	40
letter_100_100	20000	16	241	39	6	3
letter_40_1	20000	16	250	266	6	5
letter_40 _100	20000	16	346	8	7	4
letter_100_10	20000	16	193	146	6	4
letter_40_10	20000	16	320	57	7	4
letter_lin_100	20000	16	17	1056	20	35
ocr9_256_100	7291	256	378	0	7	5
ocr0_256_100	7291	256	309	0	6	4
abalone_4_100	4177	10	64	1863	4	5
abalone_p5_100	4177	10	304	1520	31	22
spam_300_100	4601	57	1417	181	80	64
spam_lin_100	4601	57	58	822	15	11
adult_100_1	16100	123	97	5996	89	81
adult_100_100	16100	123	871	4823	253	175
adult_200_1	16100	123	168	5785	95	85
adult_200_100	16100	123	483	5219	107	140
adult_50_10	16100	123	615	5143	120	120

Table 3: Sprint vs. Shrinking.

Name	n	k	n_s	n_c	SVM-QP	SVM-QP$_{mem}$	SVM$^{light}_{mem}$
web_100_100	24692	300	980	453	65	428	1097
web_40_10	24692	300	1037	568	68	447	553
web_40_100	24692	300	1214	313	84	524	1201
web_100_10	24692	300	679	835	40	230	348
letter_100_100	20000	16	241	39	3	14	26
letter_40_1	20000	16	250	266	5	17	9
letter_40 _100	20000	16	346	8	4	17	15
letter_100_10	20000	16	193	146	4	14	14
letter_40_10	20000	16	320	57	4	20	11
letter_lin_100	20000	16	17	1056	35	72	1052
ocr9_256_100	7291	256	378	0	5	24	13
ocr0_256_100	7291	256	309	0	4	13	9
abalone_4_100	4177	10	64	1863	5	13	78
abalone_p5_100	4177	10	304	1520	22	44	-
adult_100_1	16100	123	97	5996	81	171	228
adult_100_100	16100	123	871	4823	175	624	1541
adult_200_1	16100	123	168	5785	85	174	253
adult_200_100	16100	123	483	5219	140	173	1360
adult_50_10	16100	123	615	5143	120	355	593

Table 4: Comparison for memory saving mode.

the user. In the experiments discussed above we allowed the size of the cache to be 500MB, which is at least as much memory as was used by SVM-QP.

We did not implement such sophisticated memory handling mechanism in our code. Luckily *sprint* provides a natural setting for a memory saving mode. Instead of storing the whole Q_s we only store the elements of Q whose columns are in I_s and whose rows are in $I_s \cup I_0' \cup I_c'$. The size of $I_0' \cup I_c'$ can be regulated according to the available storage space. At each major iteration all the elements of Q_s whose row indices are in $I_0'' \cup I_c''$ have to be recomputed. This can be a costly step. To further try to reduce the computational cost of that step, we apply *shrinking* to $I_0'' \cup I_c''$. That is, if during a few consecutive major iterations a certain reduced cost remained nonnegative then the appropriate variable is removed from $I_0'' \cup I_c''$ and is ignored until the later stage of the algorithm. In Table 4 below we present our results. We chose the size of I_0' and I_c' to be 50 each, this way the total storage for the elements of Q_s (including Q_{ss}) did not exceed 20MB. We compare our CPU time to that of SVM^{light} with 20MB of cache limit. We also list the CPU times for the version of SVM-QP that stores the full Q_s, to demonstrate the trade-off between the CPU time and memory requirement.

4.3 Warm Start

One of the significant advantages of the active set methods over the interior point methods is that the former can benefit very well from warm starts. For instance, if some additional labeled training data become available, the old optimal solution is used as a starting point for the active set algorithm

and the new optimal solution is typically obtained within a few iterations. This will be explored in more detail in the subsection on the incremental mode of our algorithm.

Another situation where warm start arises, is when one wants to explore the path of optimal solutions for various values of penalty parameter C. In Hastie et al. (2004) the whole solution path is generated using an active set method similar to ours. There some differences between the two methods, however. The method in Hastie et al. (2004) is a parametric active set method, which in practice is usually slower than a purely primal or dual active set method, such as ours. Also their method requires that at each iteration an optimal solution of a parametric problem is available, hence there does not seem to be any possibility to use *sprint* or *shrinking*. It remains to be seen whether a good implementation of the algorithm in Hastie et al. (2004) can match the performance of our algorithm.

Our algorithm is not suitable directly for generating the entire parametric path, but using the warm starts one can easily use it to generate solutions for a selection of the values of parameter C.

The warm starts can also be used when one wants to explore different values of kernel parameters, but the efficiency of such application needs a separate computational study.

Here we investigate the use of warm start to increase the efficiency of the algorithm itself. It has been noticed (see, for example, Fine and Scheinberg, 2001) that for many SVM problems the matrix Q has eigenvalues decaying to zero. It was suggested in Fine and Scheinberg (2001) to use a low rank approximation of Q and solve the approximate problem with an interior point method using product form Cholesky factorizations, which benefit from the low rank of Q. Such approximations, however, are not always very accurate. The idea we explore here is to use the solution of the approximate problem to warm start the active set method.

If k is the rank of the approximation of Q, then per iteration complexity of the IPM is $O(nk^2)$. There is a trade-off in choosing the right value for k: if k is chosen to be too large, then the IPM will not be efficient and if k is too small then the solution produced by the IPM is too far from the optimal solution of the true problem. We chose $k = 50$, which is reasonably small to make the IPM part fast and sufficiently large to hope for a good warm start. The results in Table 5 are not as dramatic as one might hope. Often the active set method itself is so fast that it outperforms the IPM even for $k = 50$, for instance on **letter_x_x** problems. In other cases the approximation does not produce a good enough warm start. There also cases where Q itself has very low rank and, hence, the problem can be solved to optimality just by the IPM; see **letter_lin_100**, for instance. There are examples however, where the combined method achieves better timing results than either method, when used separately. This seem to happen for the problems with relatively large I_c sets, such as the **adult_x_x** problems. We have to note that we are using a rather crude implementation of the IPM for SVM. One might achieve better results with a more efficient implementation of an IPM.

4.4 Incremental Mode

Incremental mode is used when the training data is available one point (or a few points) at a time. Our algorithm applies naturally and almost without change to the incremental mode. Whenever more data points become available, their indices get placed in set I_0, then Step 2(i) is applied to price the corresponding variables, and if a negative reduced cost is found, then the algorithm proceeds in the usual manner. The only difference with the batch case (when all data is available at once) is that the pricing is Step 2(i) cannot be applied to the data that is not available yet. Hence, the constraints

Name	n	k	n_s	n_c	SVM-QP$_p$	SVM-QP
web_100_100	24692	300	980	453	113	65
web_40_10	24692	300	1037	568	112	68
web_40_100	24692	300	1214	313	142	84
web_100_10	24692	300	679	835	82	40
letter_100_100	20000	16	241	39	36	3
letter_40_1	20000	16	250	266	38	5
letter_40 _100	20000	16	346	8	49	4
letter_100_10	20000	16	193	146	33	4
letter_40_10	20000	16	320	57	44	4
letter_lin_100	20000	16	17	1056	7	35
ocr9_256_100	7291	256	378	0	21	5
ocr0_256_100	7291	256	309	0	15	4
abalone_4_100	4177	10	64	1863	5	5
abalone_p5_100	4177	10	304	1520	10	22
spam_300_100	4601	58	1417	181	40	64
spam_lin_100	4601	58	58	822	9	11
adult_100_1	16100	123	97	5996	66	81
adult_100_100	16100	123	871	4823	125	175
adult_200_1	16100	123	168	5785	68	85
adult_200_100	16100	123	483	5219	92	140
adult_50_10	16100	123	615	5143	95	120

Table 5: Warm starting SVM-QP by an IPM.

Name	n	k	n_s	n_c	SVM-QP	SVM-QP$_{inc}$
web_100_100	24692	300	980	453	65	1388
web_40_10	24692	300	1037	568	68	1017
web_40_100	24692	300	1214	313	84	1190
web_100_10	24692	300	679	835	40	1101
letter_100_100	20000	16	241	39	3	24
letter_40_1	20000	16	250	266	5	37
letter_40 _100	20000	16	346	8	4	34
letter_100_10	20000	16	193	146	4	24
letter_40_10	20000	16	320	57	4	39
letter_lin_100	20000	16	17	1056	35	43
ocr9_256_100	7291	256	378	0	5	27
ocr0_256_100	7291	256	309	0	4	16
abalone_4_100	4177	10	64	1863	5	13
abalone_p5_100	4177	10	304	1520	22	137
spam_300_100	4601	58	1417	181	64	414
spam_lin_100	4601	58	58	822	11	647
adult_100_1	16100	123	97	5996	81	372
adult_100_100	16100	123	871	4823	175	5882
adult_200_1	16100	123	168	5785	85	330
adult_200_100	16100	123	483	5219	140	1819
adult_50_10	16100	123	615	5143	120	2274

Table 6: Incremental mode.

with sufficiently negative reduced costs are not included, until their data points are added to the problem. As we show in Table 6, this results in a dramatic increase of CPU time.

Notice, that the sifting does not make sense in the incremental mode, since it selects the sets I_0' and I_c' based on the entire data set. However, shrinking can be easily applied, since its selection of I_0' and I_c' is only based on the past behavior of each individual constraint.

5. Concluding Remarks

Traditional active set methods for convex QPs were considered impractical for large-scale SVM problems. However, they have theoretical appeal for many reasons. In this paper we studied in details an active set method SVM and show that an efficient implementation can outperform other state-of-the-art SVM software.

Furture direction of this work lies in a more comprehensive theoretical analysis of the behavior and complexity of the method for SVM problems.

Acknowledgments

The author is grateful to Alexandre Belloni for pointing out the paper (Frangioni, 1996) for handling the singular reduced QP system as well as for many fruitful discussions. The author is also grateful to John Forrest for suggesting the *sprint* technique.

References

J. Balcazar, Y. Dai, and O Watanabe. Provably fast trainning algorithms for support vector machines. In *Proc. 1st IEEE International Conference on Data Mining*, pages 43–50. IEEE, 2001.

R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten, and D. F. Shanno. Very large-scale linear programming: A case study in combining interior point and simplex methods. *Operations Research*, 40(5):885–897, 1992.

C. L. Blake and C. J Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

B. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 409–415. MIT Press, 2001.

N. Cristianini and J. Shawe-Taylor. *An Introductin to Support Vector Macines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

M. C. Ferris and T. S. Munson. Interior point methods for massive support vector machines. Technical Report 00-05, Computer Sciences Department, University of Wisconsin, Madison, WI, 2000.

S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

R. Fletcher. A general quadratic programming algorithm. *Journal of the Institute of Mathematics and Its Applications*, pages 76–91, 1971.

J. J. Forrest. Mathematical programming with a library of optimization subroutines. Presented at the ORSA/TIMS Joint National Meeting, 1989.

A. Frangioni. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Computers in Operations Research*, 23(11):1099–1118, 1996.

D. Goldfarb. Extension of newton's method and simplex method for solving quadratic problems. In F. A. Lootsma, editor, *Numerical Methods for Nonlinear Optimization*, pages 239–254. Academic Press, London, 1972.

D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programms. *Mathematical Programming*, 27:1–33, 1983.

G. H. Golub and Ch. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

T. Joachims. Making large-scale support vector machine nlearning practicle. In B. Schölkopf, C. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods*, chapter 12, pages 169–184. MIT Press, 1999.

L. Kaufman. Solving the quadratic programming problem arising in support vector classification. In B. Schölkopf, C. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods Support Vector Learning*, pages 147–168. MIT Press, 1998.

K. C. Kiwiel. A dual method for certain positive semidefinite quadratic programming problems. *SIAM Journal on Scientific and Statistical Computing*, 10:175–186, 1989.

J Nocedal and S. Wright. *Numerical Optimization.* Springer-Verlag, 1999.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the IEEE Neural Networks for Signal Processing VII Workshop*, pages 276–285. IEEE, 1997.

J. C. Platt. Fast trining support vector machines using sequential mininal optimization. In B. Schölkopf, C. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods*, chapter 12, pages 185–208. MIT Press, 1999.

R. Vanderbei. *Linear Programming: Foundations and Extensions*. Sringer, New York, NY, 2nd edition, 2001.

Journal of Machine Learning Research 7 (2006) 2259-2301 Submitted 10/05; Revised 7/06; Published 11/06

Causal Graph Based Decomposition of Factored MDPs

Anders Jonsson ANDERS.JONSSON@UPF.EDU
Departament de Tecnologia
Universitat Pompeu Fabra
Passeig de Circumval·lació, 8
08003 Barcelona, Spain

Andrew Barto BARTO@CS.UMASS.EDU
Computer Science Department
University of Massachusetts
Amherst MA 01003, USA

Editor: Carlos Guestrin

Abstract

We present Variable Influence Structure Analysis, or VISA, an algorithm that performs hierarchical decomposition of factored Markov decision processes. VISA uses a dynamic Bayesian network model of actions, and constructs a causal graph that captures relationships between state variables. In tasks with sparse causal graphs VISA exploits structure by introducing activities that cause the values of state variables to change. The result is a hierarchy of activities that together represent a solution to the original task. VISA performs state abstraction for each activity by ignoring irrelevant state variables and lower-level activities. In addition, we describe an algorithm for constructing compact models of the activities introduced. State abstraction and compact activity models enable VISA to apply efficient algorithms to solve the stand-alone subtask associated with each activity. Experimental results show that the decomposition introduced by VISA can significantly accelerate construction of an optimal, or near-optimal, policy.

Keywords: Markov decision processes, hierarchical decomposition, state abstraction

1. Introduction

Markov decision processes, or MDPs, are widely used to model stochastic control tasks. Many researchers have developed algorithms that determine optimal or near-optimal decision policies for MDPs. However, most of these algorithms scale poorly as the size of a task grows. Much recent research on MDPs has focused on finding task structure that makes it possible to simplify construction of a useful policy. In this paper, we present Variable Influence Structure Analysis, or VISA, an algorithm that identifies task structure in factored MDPs and combines hierarchical decomposition and state abstraction to exploit task structure and simplify policy construction. VISA was first introduced in a conference paper (Jonsson and Barto, 2005); this paper provides more detail and additional insights as well as a new section on compact activity models.

Hierarchical decomposition exploits task structure by introducing stand-alone policies (also known as activities, macro-actions, temporally-extended actions, options, or skills) that can take multiple time steps to execute. We use the term activity (Harel, 1987) to denote such a stand-alone policy. Activities can exploit repeating structure by representing subroutines that are executed multiple times during solution of a task. If an activity has been learned in one task, it can be reused

in other tasks that require execution of the same subroutine. Activities also enable more efficient exploration by providing high-level behavior that enables a decision maker to look ahead to the completion of the associated subroutine. There exist three major models of activities in reinforcement learning: Hierarchical Abstract Machines, or HAMs (Parr and Russell, 1998), options (Sutton et al., 1999), and MAXQ (Dietterich, 2000a).

It may not be apparent to a system designer how to select subroutines that enable efficient hierarchical decomposition. To take full advantage of hierarchical decomposition, a system should be able to identify useful subroutines on its own. Several researchers have developed algorithms that use task-specific knowledge to identify useful subroutines. One approach is to identify useful subgoals and introduce activities that accomplish the subgoals (Digney, 1996; McGovern and Barto, 2001; Şimşek and Barto, 2004). Another approach is to solve several tasks and identify activities that are useful across tasks (Pickett and Barto, 2002; Thrun and Schwartz, 1996). There also exist algorithms that use graph theory to cluster states into regions and introduce activities for moving between regions (Menache et al., 2002; Mannor et al., 2004; Şimşek et al., 2005). Other algorithms introduce activities that cause the values of specific variables to change (Hengst, 2002; Singh et al., 2005).

Hierarchical decomposition is intimately related to state abstraction, that is, ignoring part of the available information to reduce the effective size of the state space. At each moment, only some of the information that is part of the state description may be relevant for selecting an optimal action. For example, the color of the wall is most likely irrelevant for the task of navigating to the front door of a building. State abstraction compresses the state space by grouping together states that only differ on irrelevant information. Each group of states can be treated as a single state, reducing the complexity of policy computation. Dean and Givan (1997) showed that under certain conditions, the optimal policy of an MDP is preserved under state abstraction.

Each activity can be viewed as a stand-alone subtask that can be solved independently. If each subtask is as difficult to solve as the original task, hierarchical decomposition actually increases the complexity of finding an optimal policy. However, if state abstraction is used to simplify the solution of each subtask, hierarchical decomposition can significantly accelerate policy computation. In particular, information that is relevant for one subtask may be irrelevant for another. In other words, it makes sense to perform state abstraction separately for each subtask (Dietterich, 2000b; Jonsson and Barto, 2001).

VISA uses a compact model of factored MDPs first suggested by Boutilier et al. (1995). When an action is executed, the resulting value of a state variable depends on the values of state variables prior to executing the action. In many cases, the resulting value is conditionally independent of a subset of the state variables at the previous time step. The compact model uses dynamic Bayesian networks, or DBNs (Dean and Kanazawa, 1989), to represent the effect of actions in factored MDPs. Since DBNs encode conditional independence, the model can represent the effect of actions using much less memory than the number of states. Several researchers have developed algorithms that take advantage of the DBN model to efficiently compute policies of factored MDPs (Boutilier et al., 1995; Feng et al., 2003; Guestrin et al., 2001; Hoey et al., 1999; Kearns and Koller, 1999).

2. Overview

In addition to being compact, the DBN model contains information about the preconditions necessary for an action to have the desired effect. For example, consider a task in which the objective is

to play music, described by two state variables: one representing my current location, and the other representing the current sound level. There are actions for changing my location, and an action to turn on the stereo. Being next to the stereo is a precondition for causing music to play when making a motion to turn on the stereo, a fact that is encoded in the transition probabilities of the DBN model. In other words, there is a causal relationship between the location variable and the sound level variable, conditional on the action of turning on the stereo.

To change the value of the sound level variable, it is first necessary to satisfy the precondition of being next to the stereo. Thus, a useful activity is one that causes my location to be next to the stereo. Given such an activity, it is straightforward to solve the task: first execute the activity that causes my location to be next to the stereo, and then turn on the stereo. The idea behind VISA is to use the DBN model to identify the preconditions necessary to change the value of each state variable and introduce activities for satisfying those preconditions. The result is a hierarchy of activities that can be used in a compact representation of the solution to the factored MDP. The HEX-Q algorithm (Hengst, 2002) is based on similar ideas, but does not use the DBN model to identify preconditions.

The goal of VISA is to introduce activities in such a way that their associated subtasks are easier to solve than the original task. Since the DBN model implicitly represents relationships between state variables, it is relatively easy to determine which state variables and activities are relevant for solving a particular subtask. This makes it possible to perform state abstraction for subtasks by ignoring irrelevant state variables and activities. For example, while causing my location to be next to the stereo, it is possible to ignore differences in sound level, since the sound level typically has no impact on location. If the subtasks are sufficiently easy to solve, hierarchical decomposition can lead to a significant reduction in computational complexity.

VISA, described in Section 4, uses the DBN model to construct a causal graph describing state variable relationships. If two state variables mutually influence each other, it is not possible to introduce activities that change the value of one without taking into account the value of the other. Consequently, it is not possible to perform state abstraction in a way that makes the associated subtasks easier to solve. State variables that mutually influence each other correspond to cycles in the causal graph, so VISA gets rid of cycles by identifying the strongly connected components of the graph and constructing a component graph with one node per component.

The algorithm then identifies exits (Hengst, 2002), that is, combinations of variable values and actions that cause the value of some state variable to change. For each exit, VISA introduces an activity that solves the subtask of changing the corresponding variable value. VISA uses the causal graph to identify state variables and activities that are relevant for solving the subtask, and performs state abstraction by ignoring irrelevant state variables and activities. At the top level, the algorithm introduces an activity that corresponds to the original MDP. Experimental results show that the decomposition generated by VISA can significantly accelerate construction of an optimal or near-optimal policy.

If VISA had access to compact models of activities, similar to the DBN model of primitive actions, it could apply more efficient algorithms to construct the stand-alone policies of activities. To fully model the stand-alone subtask associated with each activity, it is necessary to determine the transition probabilities of the lower-level activities used to solve the subtask. However, existing methods cannot determine transition probabilities of activities without enumerating the state space. Since the state space grows exponentially with the number of state variables, this seems like a bad idea. Instead, the implementation of VISA in Section 4 uses reinforcement learning (Sutton and

Barto, 1998), which does not require knowledge of transition probabilities, to learn the policy of each activity.

In Section 5, we describe an algorithm that constructs compact activity models without enumerating the state space. We decompose computation of an activity model by considering the conditional probabilities of one state variable at a time. The result is a DBN model for each activity identical to the DBN model of primitive actions. VISA can then apply the more efficient algorithms that take advantage of DBN models, accelerating construction of the stand-alone policies even further. Experimental results show that our algorithm can construct compact activity models without significantly decelerating solution of a task.

3. Background

In this section, we provide a background to the problem that our algorithm attempts to solve, and we introduce notation of concepts that we use throughout the paper.

3.1 Markov Decision Processes

A finite Markov decision process, or MDP (Bellman, 1957), is a tuple $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$, where S is a finite set of states, A is a finite set of actions, $\Psi \subseteq S \times A$ is a set of admissible state-action pairs, P is a transition probability function, and R is an expected reward function. Let $A_s \equiv \{a \in A \mid (s,a) \in \Psi\}$ be the set of admissible actions in state $s \in S$. Ψ is such that for each state $s \in S$, A_s is non-empty, that is, there is at least one admissible action for each state. As a result of executing action $a \in A_s$ in state $s \in S$, the process transitions to state $s' \in S$ with probability $P(s' \mid s,a)$ and provides the decision maker with an expected reward $R(s,a)$. P is such that for each admissible state-action pair $(s,a) \in \Psi$, $\sum_{s' \in S} P(s' \mid s,a) = 1$.

For each state $s \in S$ and each action $a \in A_s$, a stochastic policy π selects action a in state s with probability $\pi(s,a)$. π is such that for each state $s \in S$, $\sum_{a \in A_s} \pi(s,a) = 1$. In the discounted case, the optimal value function V^* associated with MDP $\mathcal{M}$ is defined by the Bellman optimality equation:

$$V^*(s) = \max_{a \in A_s} \left[R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V^*(s') \right], \tag{1}$$

where γ is a discount factor. An optimal policy π^* is any stochastic policy that, in each state $s \in S$, assigns positive probabilities only to actions in the set

$$A^*(s) \equiv \arg\max_{a \in A_s} \left[R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V^*(s') \right].$$

A factored MDP is described by a set of discrete state variables $\mathbf{S}$. Each state variable $S_i \in \mathbf{S}$ takes on values in its domain $\mathcal{D}(S_i)$. The set of states $S \subseteq \times_{S_i \in \mathbf{S}} \mathcal{D}(S_i)$ is a subset of the Cartesian product of the state variable domains. A state $\mathbf{s} \in S$ is an assignment of values to the set of state variables $\mathbf{S}$. Let $f_{\mathbf{C}}$, $\mathbf{C} \subseteq \mathbf{S}$, be a projection such that if $\mathbf{s}$ is an assignment to $\mathbf{S}$, $f_{\mathbf{C}}(\mathbf{s})$ is $\mathbf{s}$'s assignment to $\mathbf{C}$. We define a *context* $\mathbf{c}$ as an assignment of values to the subset of state variables $\mathbf{C} \subseteq \mathbf{S}$.

3.2 Coffee Task

We illustrate factored MDPs using the coffee task (Boutilier et al., 1995), in which a robot has to deliver coffee to its user. The coffee task is described by six binary state variables: S_{L}, the robot's

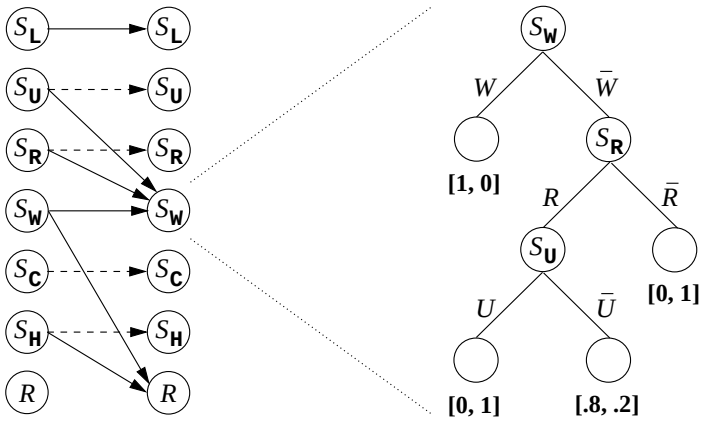


Figure 1: The DBN for action GO in the coffee task

location (office or coffee shop); $S_{\mathtt{U}}$, whether the robot has an umbrella; $S_{\mathtt{R}}$, whether it is raining; $S_{\mathtt{W}}$, whether the robot is wet; $S_{\mathtt{C}}$, whether the robot has coffee; and $S_{\mathtt{H}}$, whether the user has coffee. To distinguish between variable values we use the notation $\mathcal{D}(S_i) = \{i, \bar{i}\}$, which has obvious meaning for all state variables except $S_{\mathtt{L}}$, where we use L to denote the coffee shop and $\overline{L}$ to denote the office. The robot has four actions: GO, causing its location to change and the robot to get wet if it is raining and it does not have an umbrella; BC (buy coffee) causing it to hold coffee if it is in the coffee shop; GU (get umbrella) causing it to hold an umbrella if it is in the office; and DC (deliver coffee) causing the user to hold coffee if the robot has coffee and is in the office. All actions have a chance of failing. The robot gets a reward of 0.9 whenever the user has coffee plus a reward of 0.1 whenever it is dry.

3.3 DBN Model

Boutilier et al. (1995) developed a compact model of factored MDPs that uses dynamic Bayesian networks, or DBNs (Dean and Kanazawa, 1989), to represent the effect of actions. The DBN model contains one DBN for each action $a \in A$ of a factored MDP. Figure 1 illustrates the DBN for action GO in the coffee task. The DBN has two nodes for each state variable plus two nodes representing expected reward. Nodes on the left represent the values of variables prior to executing GO, and nodes on the right represent the values after executing GO. The value of a state variable S_i as a result of executing GO depends on the values of state variables that have edges to S_i in the DBN. Let $\mathbf{Pa}(S_i) \subseteq \mathbf{S}$ denote the subset of state variables with edges to S_i. A dashed line indicates that a state variable is unaffected by GO.

In the DBN for action a, each state variable S_i is associated with a conditional probability distribution P_i^a that determines the value of S_i after executing a. Like Boutilier et al. (1995), we assume that conditional probabilities are stored in trees. Figure 1 illustrates the conditional probability tree associated with state variable $S_{\mathtt{W}}$ and action GO. For example, if the robot is dry ($\overline{W}$), it is raining (R), and the robot does not have an umbrella ($\overline{U}$), the robot becomes wet with probability 0.8 after executing GO. We assume that there are no edges between state variables in the same layer of the DBN. Consequently, the DBN model cannot represent arbitrary transition probabilities. Instead, the

transition probabilities are approximated according to $P(\mathbf{s}' \mid \mathbf{s}, a) \approx \prod_{S_i \in \mathbf{S}} P_i^a(S_i = f_{\{S_i\}}(\mathbf{s}') \mid \mathbf{Pa}(S_i) = f_{\mathbf{Pa}(S_i)}(\mathbf{s}))$.

3.4 Options

We use *options* (Sutton et al., 1999) to model activities. Given an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$, an option is a tuple $o = \langle I, \pi, \beta \rangle$, where $I \subseteq S$ is an initiation set, π is a policy, and β is a termination function. Option o can be executed in any state $s \in I$, repeatedly selects actions $a \in A$ according to π, and terminates in state $s' \in S$ with probability $\beta(s')$. An action $a \in A$ can be viewed as an option with initiation set $I = \{s \in S \mid (s,a) \in \Psi\}$ whose policy always selects a and terminates in all states with probability 1. Adding options to the action set of an MDP forms a semi-Markov decision process, or SMDP (Puterman, 1994). It is possible to construct hierarchies of options in which the options on one level selects among options on a lower level.

Ravindran (2004) showed that an option o is associated with a stand-alone task given by the option SMDP $\mathcal{M}_o = \langle S, O_o, \Psi_o, P_o, R_o \rangle$, where O_o is a set of lower-level options. The set of admissible state-option pairs $\Psi_o \subseteq S \times O_o$ is determined by the initiation sets of options in O_o. The transition probability function P_o is determined by the transition probability function P of the underlying MDP and the policies and termination functions of the options in O_o. The expected reward function R_o is independent of the expected reward function R of the underlying MDP and can be selected to reflect the desired behavior of option o. The policy π of option o can be defined as any optimal policy of the option SMDP $\mathcal{M}_o$.

SMDP Q-learning (Bradtke and Duff, 1995) maintains estimates of the optimal option-value $Q(s,o)$, representing the return for executing option o in state s. Following execution of an option o in state s, the option-value is updated using the following update rule:

$$Q(s,o) \leftarrow Q(s,o) + \alpha \left[r + \gamma^k \max_{o' \in O_o} Q(s',o') - Q(s,o) \right],$$

where s' is the state in which o terminated, k is the number of time steps elapsed during the execution of o, r is the cumulative discounted reward during this time, and α is the learning rate. Parr (1998) showed that SMDP Q-learning eventually converges to an optimal policy when the learning rate α is appropriately decreased towards 0.

3.5 State Abstraction

We use partitions to represent state abstraction in MDPs. A partition Λ of the set of states S is a collection of disjoint subsets, or blocks, $\lambda \subseteq S$ such that $\bigcup_{\lambda \in \Lambda} \lambda = S$. $[s]_\Lambda \in \Lambda$ denotes the block to which state $s \in S$ belongs. A function $f : S \to X$ from S onto an arbitrary set X induces a partition Λ_f of S such that for each pair of states $(s_i, s_j) \in S^2$, $[s_i]_{\Lambda_f} = [s_j]_{\Lambda_f}$ if and only if $f(s_i) = f(s_j)$. Let Λ_1 and Λ_2 be two partitions of S. Partition Λ_1 refines Λ_2, denoted $\Lambda_1 \leq \Lambda_2$, if and only if, for each pair of states $(s_i, s_j) \in S^2$, $[s_i]_{\Lambda_1} = [s_j]_{\Lambda_1}$ implies that $[s_i]_{\Lambda_2} = [s_j]_{\Lambda_2}$. The relation $\leq$ is a partial ordering on the set of partitions of S.

Dean and Givan (1997) defined two properties of partitions of the set of states S. A partition Λ has the stochastic substitution property if, for each pair of states $(s_i, s_j) \in S^2$, each action $a \in A$ and each block $\lambda \in \Lambda$, $[s_i]_\Lambda = [s_j]_\Lambda$ implies that $\sum_{s_k \in \lambda} P(s_k \mid s_i, a) = \sum_{s_k \in \lambda} P(s_k \mid s_j, a)$. Λ is reward respecting if for each pair of states $(s_i, s_j) \in S^2$ and each action $a \in A$, $[s_i]_\Lambda = [s_j]_\Lambda$ implies that $R(s_i, a) = R(s_j, a)$. A partition Λ that has the stochastic substitution property and is reward

respecting induces a reduced MDP which has fewer states and preserves optimality (Dean and Givan, 1997). Ravindran (2004) developed a theory of MDP homomorphisms and extended the above definitions to partitions of the set Ψ of admissible state-action pairs.

4. VISA

Variable Influence Structure Analysis, or VISA (Jonsson and Barto, 2005), is an algorithm that analyzes causal relationships between state variables to perform hierarchical decomposition of factored MDPs. VISA uses the DBN model of factored MDPs to compactly represent transition probabilities and expected reward. However, VISA makes additional use of the DBN model. The conditional probability distributions of the DBN model specify which preconditions have to hold for the value of a state variable to change as a result of executing an action. The aim of VISA is to facilitate variable value changes by introducing activities that satisfy those preconditions. For example, if the task is to play music, a useful activity is one that causes my location to be next to the stereo, since being next to the stereo is a precondition for successfully making a motion to turn it on.

Algorithm 1 VISA

```
1: Input: DBN model of a factored MDP
2: construct the causal graph of the task
3: identify the strongly connected components of the causal graph
4: for each strongly connected component
5:     identify a set of exits that cause the values of state variables in the component to change
6:     for each exit
7:         construct the components of an option SMDP
8:         use the causal graph to perform state abstraction for the option SMDP
9: apply reinforcement learning techniques to learn the policy of each option SMDP
```

Algorithm 1 gives a high-level description of VISA. Before decomposing the task, VISA uses the DBN model to construct a causal graph that determines how state variables influence each other. A state variable influences another if it appears in the precondition of an action that changes the value of the latter. If two state variables mutually influence each other, changing the value of one variable depends on the value of the other. Thus, it is impossible to decompose the task by introducing activities that exclusively change the value of one of the variables. State variables that mutually influence each other correspond to cycles in the causal graph. VISA gets rid of cycles by identifying the strongly connected components of the causal graph. State variables in a strongly connected component are treated as a single variable for the purpose of decomposition.

For each strongly connected component, VISA searches the conditional probability distributions of the DBN model for *exits* (Hengst, 2002), that is, pairs of a precondition and an action that cause the value of a state variable in the component to change. For each exit, VISA introduces an exit option, that is, an activity that terminates when the precondition of the exit is met and then executes the exit action. In other words, the purpose of an exit option is to change the value of a state variable by first satisfying the necessary precondition and then executing the appropriate action. To determine the policy of each exit option, VISA constructs the components of an option SMDP and defines the policy as a solution to the resulting option SMDP.

To simplify learning the policy of an exit option, VISA performs state abstraction for the option SMDP. From the causal graph it is easy to identify a set of state variables that are irrelevant, that

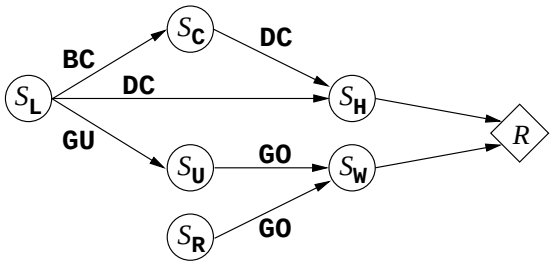


Figure 2: The causal graph of the coffee task

is, do not influence state variables whose values appear in the precondition. VISA performs state abstraction by ignoring differences in the values of irrelevant state variables. In addition, the option SMDP only needs to include actions and options that change the values of state variables that appear in the precondition. The resulting state abstraction significantly reduces the complexity of learning the exit option policies. The causal graph implicitly defines a hierarchy of options in which an exit option that changes the value of a state variable in a strongly connected component selects between options that change the values of state variables in strongly connected components with incoming edges.

4.1 Causal Graph

The first step of VISA is to construct a causal graph representing the causal relationships between state variables. The causal graph contains one node per state variable plus one node corresponding to expected reward. There is a directed edge between two state variables S_j and S_i if and only if there exists an action $a \in A$ such that there is an edge between S_j and S_i in the DBN for a. In other words, each edge in the causal graph represents a causal relationship between two state variables conditional on one or several actions. The algorithm labels each edge with the actions that give rise to the causal relationship.

Recall that Figure 1 shows the DBN for action GO in the coffee task. There are several interesting things to note. For each state variable S_i, the value of S_i as a result of executing GO depends on the value of S_i prior to executing GO. In other words, each node in the causal graph should have an associated reflexive edge. However, we are not interested in the causal relationship of a state variable onto itself, so we remove reflexive edges in the causal graph. Also, there are edges from state variable $S_{\mathtt{U}}$ to state variable $S_{\mathtt{W}}$ in the DBN, as well as from $S_{\mathtt{R}}$ to $S_{\mathtt{W}}$. Consequently, there should be an edge from $S_{\mathtt{U}}$ to $S_{\mathtt{W}}$ in the causal graph labeled GO, as well as an edge from $S_{\mathtt{R}}$ to $S_{\mathtt{W}}$ labeled GO.

The causal graph of the coffee task is shown in Figure 2. Note that the edges from the DBN for action GO have been incorporated, as well as edges from the DBNs for the other actions. Also note that there are no cycles in the causal graph. However, this is not true for arbitrary tasks, since it is possible for state variables to mutually influence each other. VISA gets rid of cycles in the causal graph by identifying the strongly connected components of the graph, each consisting of one or several state variables that are pairwise connected through directed paths. It is possible to construct a component graph in which each node is a strongly connected component, and which has an edge between two nodes if and only if there is an edge in the causal graph between a state variable of the

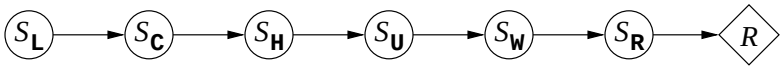


Figure 3: HEX-Q's state variable ordering in the coffee task

first component and a state variable of the second component. The component graph is guaranteed to contain no cycles. In the coffee task, each state variable in the causal graph is its own strongly connected component, so the component graph is identical to the causal graph.

The expected reward node deserves additional explanation. Just as for the other variables, there is an edge in the causal graph between a state variable S_i and the expected reward node if and only if there is an action $a \in A$ such that the expected reward as a result of executing a depends on the value of S_i. All edges to the expected reward node are incoming, since the value of a state variable never depends on the expected reward received at the previous time step. Thus, the expected reward as a result of executing any action only depends on the values of state variables with edges to the expected reward node in the causal graph. For the purpose of optimizing reward, it is only necessary to consider actions and options that change the values of those state variables.

4.2 Identifying Exits

VISA builds on ideas from the HEX-Q algorithm (Hengst, 2002), an algorithm that also performs hierarchical decomposition of factored MDPs. The HEX-Q algorithm first determines an ordering on the state variables by randomly executing actions and counting the frequency with which the value of each state variable changes. The state variable whose value changes the most frequently becomes the lowest variable in the ordering, and so on. For each state variable S_i in the ordering, the HEX-Q algorithm identifies exits $\langle k, a \rangle$, pairs of a state variable value $k \in \mathcal{D}(S_i)$ and an action $a \in A$, that cause the value of the next state variable in the ordering to change. The HEX-Q algorithm introduces an option for each exit, and the options on one level of the hierarchy become actions on the next level.

Even though the HEX-Q algorithm achieved some early success, the frequency of change heuristic may not be an accurate indicator of how state variables influence each other. In addition, the ordering does not capture the fact that the value of a state variable may depend on multiple other state variables. Figure 3 illustrates the state variable ordering that the HEX-Q algorithm comes up with in the coffee task. There are several differences between this ordering and the causal graph. The ordering wrongly concludes that state variable $S_{\mathtt{W}}$ influences $S_{\mathtt{R}}$, when it is really the other way around. The ordering also fails to capture the fact that the value of $S_{\mathtt{H}}$ depends on both $S_{\mathtt{L}}$ and $S_{\mathtt{C}}$.

VISA also searches for exits that cause the values of state variables to change. However, instead of the frequency of change heuristic, VISA uses the causal graph to determine how state variables influence each other. Since the causal graph more realistically describes the causal relationships between state variables, VISA is able to successfully decompose more general tasks than the HEX-Q algorithm. Also, since the value of a state variable may depend on several other state variables, an exit $\langle \mathbf{c}, a \rangle$ in VISA is composed of a context $\mathbf{c}$ and an action $a \in A$. Recall that a context $\mathbf{c}$ is an assignment of values to a subset $\mathbf{C} \subseteq \mathbf{S}$ of the state variables.

VISA searches for exits in the conditional probability trees of the DBN model. Consider the conditional probability tree associated with state variable $S_{\mathtt{W}}$ and action `GO` in Figure 1. The third

EXIT	VARIABLE	CHANGE
$\langle(), \texttt{GO}\rangle$	$S_{\texttt{L}}$	$\overline{L} \rightarrow L$, $L \rightarrow \overline{L}$
$\langle(S_{\texttt{L}} = L), \texttt{BC}\rangle$	$S_{\texttt{C}}$	$\overline{C} \rightarrow C$
$\langle(S_{\texttt{L}} = \overline{L}), \texttt{DC}\rangle$	$S_{\texttt{C}}$	$C \rightarrow \overline{C}$
$\langle(S_{\texttt{L}} = \overline{L}, S_{\texttt{C}} = C), \texttt{DC}\rangle$	$S_{\texttt{H}}$	$\overline{H} \rightarrow H$
$\langle(S_{\texttt{L}} = \overline{L}), \texttt{GU}\rangle$	$S_{\texttt{U}}$	$\overline{U} \rightarrow U$
$\langle(S_{\texttt{U}} = \overline{U}, S_{\texttt{R}} = R), \texttt{GO}\rangle$	$S_{\texttt{W}}$	$\overline{W} \rightarrow W$

Table 1: Exits identified in the coffee task

leaf from the left is associated with states that assign $\overline{W}$ to $S_{\texttt{W}}$, R to $S_{\texttt{R}}$, and $\overline{U}$ to $S_{\texttt{U}}$. As a result of executing action GO in such states, the value of $S_{\texttt{W}}$ becomes W with probability 0.8. Since the value of state variable $S_{\texttt{W}}$ changes from $\overline{W}$ to W with non-zero probability, VISA generates an exit $\langle(S_{\texttt{U}} = \overline{U}, S_{\texttt{R}} = R), \texttt{GO}\rangle$ that causes the value of $S_{\texttt{W}}$ to change. The context of the exit is determined by the values of state variables on the path from the root to the leaf. Note that the value of $S_{\texttt{W}}$ does not appear in the exit since that is the state variable whose value the exit changes. Also note that the exit $\langle(S_{\texttt{U}} = \overline{U}, S_{\texttt{R}} = R), \texttt{GO}\rangle$ does not cause the value of $S_{\texttt{W}}$ to change with probability 1, so to effectuate the change the robot may have to execute GO multiple times in the context $(S_{\texttt{U}} = \overline{U}, S_{\texttt{R}} = R)$.

Table 1 shows a complete list of exits identified by VISA in the coffee task. The table shows which state variable is affected by each exit together with the resulting change. To generate these exits, VISA had to search through each leaf of each conditional probability tree of the DBN model. At each leaf, the algorithm examined whether the value of state variable S_i changes, where S_i is the state variable whose conditional probabilities the current tree represents. In other words, the complexity of this part of the algorithm is proportional to the number of leaves of the conditional probability trees.

4.3 Exit Transformations

Sometimes it is possible to transform exits in order to take further advantage of causality. Consider the two exits $\langle(S_{\texttt{L}} = \overline{L}), \texttt{DC}\rangle$ and $\langle(S_{\texttt{L}} = \overline{L}, S_{\texttt{C}} = C), \texttt{DC}\rangle$ in the coffee task. These are almost identical: their associated exit options both terminate in states that assign the value $\overline{L}$ to state variable $S_{\texttt{L}}$ and execute action DC following successful termination. Recall that $C \rightarrow \overline{C}$ is the exit option associated with the exit $\langle(S_{\texttt{L}} = \overline{L}), \texttt{DC}\rangle$, causing the value of $S_{\texttt{C}}$ to change from C to $\overline{C}$. The effect of the exit $\langle(S_{\texttt{L}} = \overline{L}, S_{\texttt{C}} = C), \texttt{DC}\rangle$ is equivalent to the effect of a transformed exit $\langle(S_{\texttt{C}} = C), C \rightarrow \overline{C}\rangle$, that is, reach a state that assigns C to $S_{\texttt{C}}$ and execute option $C \rightarrow \overline{C}$ following termination. The benefit of this transformation is that the exit option $\overline{H} \rightarrow H$ associated with the exit $\langle(S_{\texttt{L}} = \overline{L}, S_{\texttt{C}} = C), \texttt{DC}\rangle$ no longer has to care about the value of $S_{\texttt{L}}$, effectively removing an edge in the component graph of the task. After identifying an exit, VISA compares it to exits identified for ancestor strongly connected components, and performs exit transformations when possible.

4.4 Introducing Exit Options

For each exit $\langle \mathbf{c}, a\rangle$ with a non-empty context $\mathbf{c}$, VISA introduces an option $o = \langle I, \pi, \beta\rangle$. Option o terminates in any state $\mathbf{s} \in S$ whose projection $f_{\mathbf{C}}(\mathbf{s})$ onto $\mathbf{C}$ equals $\mathbf{c}$. We refer to an option

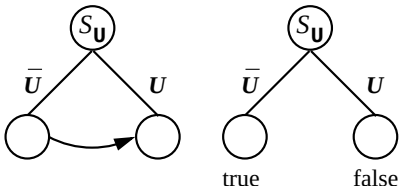


Figure 4: The transition graph (left) and reachability tree (right) of the component $S_{\mathtt{U}}$

introduced by VISA as an *exit option*. Unlike regular options, an exit option associated with an exit $\langle \mathbf{c}, a \rangle$ executes action a following termination. Note that it is not necessary to introduce options for exits with empty contexts, since these options are in fact equivalent to primitive actions. For example, VISA identifies an exit $\langle (), \mathtt{GO} \rangle$ in the coffee task. Executing action GO in any state causes the location of the robot to change, so the exit option associated with this exit is equivalent to the primitive action GO. As we shall see, it is still useful to detect exits with empty contexts.

In the coffee task example, we adopt the convention of referring to an exit option using the change that it causes, since this is an unambiguous and simple notation. For example, option $\overline{W} \rightarrow W$ is the exit option associated with the exit $\langle (S_{\mathtt{U}} = \overline{U}, S_{\mathtt{R}} = R), \mathtt{GO} \rangle$ that causes the value of $S_{\mathtt{W}}$ to change from $\overline{W}$ to W with non-zero probability. In general, several exits may cause the same change in the value of a variable, and VISA would introduce an exit option for each of these exits, so this notation is not always unambiguous.

4.4.1 INITIATION SET

The initiation set I of exit option o determines when o is admissible, that is, the subset of states in which it is possible to execute o. Two factors influence the initiation set. Option o should only be admissible in states from which it is possible to reach the associated context $\mathbf{c}$. For example, option $\overline{W} \rightarrow W$ should only be admissible in states that assign $\overline{U}$ to $\mathbf{S}_{\mathtt{U}}$ and R to $\mathbf{S}_{\mathtt{R}}$. The robot has no action for getting rid of an umbrella, and it cannot affect whether it is raining, so it can only get wet if it does not have an umbrella and it is raining. Option o should also only be admissible if it causes the value of at least one state variable to change. In our example, option $\overline{W} \rightarrow W$ should only be admissible in states that assign $\overline{W}$ to $\mathbf{S}_{\mathtt{W}}$, since otherwise the option cannot cause the value of $\mathbf{S}_{\mathtt{W}}$ to change from $\overline{W}$ to W.

VISA includes a method for constructing the initiation set of each exit option. For each strongly connected component, the algorithm constructs a transition graph that represents possible transitions between contexts in the joint value set of its state variables. Each transition graph is in the form of a tree in which possible transitions are represented as directed edges between the leaves. Possible transitions are determined using the conditional probability trees of the DBN model. Figure 4 (left) shows the transition graph of the strongly connected component containing the state variable $S_{\mathtt{U}}$ in the coffee task. The robot can acquire an umbrella by executing the exit option $\overline{U} \rightarrow U$, so there is a corresponding edge in the transition graph between the leaf associated with states that assign $\overline{U}$ to $S_{\mathtt{U}}$ and the leaf associated with states that assign U to $S_{\mathtt{U}}$. However, the robot has no action for getting rid of an umbrella, so there is no edge going the other way.

VISA uses the transition graphs to construct a set of trees that represent the initiation set I. For each transition graph, VISA constructs a *reachability tree* that classifies states based on whether

(true) or not (false) the associated context is reachable. Figure 4 (right) shows the reachability tree for S_U associated with the context $(S_U = \overline{U}, S_R = R)$ of the exit associated with $\overline{W} \rightarrow W$. Similarly, VISA constructs a reachability tree for S_R. VISA also constructs a tree that classifies states based on whether or not the associated exit changes the value of at least one state variable in the corresponding strongly connected component. In our example, states that assign $\overline{W}$ to S_W map to a leaf labeled true, and states that assign W to S_W map to a leaf labeled false. The initiation set I of option o is implicitly defined by the trees constructed by VISA. A state $\mathbf{s} \in S$ is an element in I if and only if $\mathbf{s}$ maps to a leaf labeled true in each tree. Algorithm 2 summarizes the method used by VISA to construct the initiation set of an exit option.

Algorithm 2 Initiation set

1: Input: DBN model, component graph, exit $\langle \mathbf{c}, a \rangle$
2: identify the set of components that contain state variables whose values appear in $\mathbf{c}$
3: **for each** component in this set
4: use the DBN model to construct a transition graph of the component
5: perform search in the transition graph to construct a reachability tree
6: construct a tree that determines whether $\langle \mathbf{c}, a \rangle$ changes the value of at least one variable
7: define the initiation set as the set of states that map to true in each tree

Alternatively, reachability could be computed directly using operations on trees or algebraic decision diagrams (ADDs). Feng and Hansen (2002) showed how to compute forward reachability in factored MDPs using ADDs. Since an exit represents termination of an exit option, here we are interested in computing backward reachability: from which states is it possible to reach the exit? Algorithmically, this is similar to forward reachability. However, VISA makes further use of transition graphs, and once the transition graphs have been constructed, reachability is easily computed using depth-first search on the reverse edges. For this reason we chose to stick with the above approach.

4.4.2 TERMINATION CONDITION

An exit option terminates as soon as it reaches the context $\mathbf{c}$ of its associated exit $\langle \mathbf{c}, a \rangle$, or as soon as it can no longer reach $\mathbf{c}$. Even though an exit option executes action a following termination, we can still represent termination of the option using the standard termination condition function β. For an exit option, $\beta(\mathbf{s})$ is 1 for states in the set $\{\mathbf{s} \in S \mid f_{\mathbf{C}}(\mathbf{s}) = \mathbf{c}\}$, where $\mathbf{c}$ is the associated context. $\beta(\mathbf{s})$ is also 1 for states $\mathbf{s} \notin I$, that is, when the process can no longer reach the associated context $\mathbf{c}$. In all other cases, $\beta(\mathbf{s}) = \mathbf{0}$.

4.4.3 POLICY

VISA cannot directly define the policy of an exit option since it does not know the best strategy for reaching the associated context $\mathbf{c}$. Instead, the algorithm constructs an option SMDP $\mathcal{M}_o = \langle S_o, O_o, \Psi_o, P_o, R_o \rangle$ for option o that implicitly defines its policy π. First, the algorithm defines $S_o = S$. Next, the algorithm finds all strongly connected components that contain at least one state variable whose value appears in the context $\mathbf{c}$ associated with option o. The algorithm defines O_o as the set of options that cause the values of state variables in those strongly connected components to change. For example, consider the exit option $\overline{W} \rightarrow W$ and its associated context $(S_U = \overline{U}, S_R = R)$. Two strongly connected components contain state variables whose values appear in the context: the

strongly connected component containing $S_{\mathtt{U}}$, and the strongly connected component containing $S_{\mathtt{R}}$. A single option, $\overline{U} \rightarrow U$, causes the values of state variables to change in the former component, while no option causes the values of state variables to change in the latter. In other words, the option set O_o of $\overline{W} \rightarrow W$ only needs to include the exit option $\overline{U} \rightarrow U$. Note that primitive actions may change the values of state variables in strongly connected components for which there are no options; for example, action `GO` changes the value of state variable $S_{\mathtt{L}}$.

If there are lower-level options that cause the process to leave the initiation set of an option in O_o, VISA includes these options in O_o as well. For example, the exit option $\overline{U} \rightarrow U$ causes the process to leave the initiation set of the exit option $\overline{W} \rightarrow W$. If the robot does not have an umbrella and it is raining, the exit option $\overline{W} \rightarrow W$ will no longer be admissible as a result of executing the exit option $\overline{U} \rightarrow U$ causing the robot to hold an umbrella. In other words, an option whose option set O_o includes the exit option $\overline{W} \rightarrow W$ should include the exit option $\overline{U} \rightarrow U$ as well.

VISA defines the expected reward function R_o as -1 everywhere except when option o terminates unsuccessfully, in which case the algorithm administers a large negative reward. This ensures that the policy of option o attempts to reach the context $\mathbf{c}$ as quickly as possible. Note that this may not be optimal in terms of the expected reward of the original task; we address this issue at a later point. The set of admissible state-option pairs, Ψ_o, is determined by the initiation sets of the options in O_o. VISA does not represent the transition probability function P_o explicitly. It is possible to construct a DBN model for each option similar to the DBN model for the primitive actions. However, there is currently no technique that constructs DBN models of options without enumerating all states. Since a goal of VISA is to alleviate the curse of dimensionality, we want to avoid enumerating the states. Instead, VISA uses reinforcement learning (Sutton and Barto, 1998), which does not require explicit knowledge of the transition probabilities, to learn the policy of each option. In the next section, we develop an algorithm that constructs DBN models of options identified by VISA without enumerating all states, as an alternative to reinforcement learning. The transition graphs of strongly connected components play a part in constructing DBN models of options, but nothing prevents options from changing the values of state variables that do not appear in the associated context, which makes the issue slightly more complicated.

4.5 State Abstraction

VISA simplifies learning in the option SMDPs by performing state abstraction separately for each exit option. This is where causality really matters. Let us consider all strongly connected components that contain at least one state variable whose value appears in the context $\mathbf{c}$ associated with an option. Let $\mathbf{Z} \subseteq \mathbf{S}$ denote the subset of state variables contained in those strongly connected components. Let $\mathbf{Y} \subseteq \mathbf{S}$ denote the subset of state variables S_i such that either $S_i \in \mathbf{Z}$ or there is a directed path in the causal graph from S_i to a state variable in $\mathbf{Z}$. For example, in the case of exit option $\overline{W} \rightarrow W$, $\mathbf{Z} = \{S_{\mathtt{U}}, S_{\mathtt{R}}\}$ and $\mathbf{Y} = \{S_{\mathtt{L}}, S_{\mathtt{U}}, S_{\mathtt{R}}\}$, since there is a directed path from $S_{\mathtt{L}}$ to $S_{\mathtt{U}}$ in the causal graph of the coffee task.

Recall that the goal of an exit option o is to reach the associated context $\mathbf{c}$. We know that $\mathbf{C} \subseteq \mathbf{Z} \subseteq \mathbf{Y}$, that is, that the state variables whose values appear in the context $\mathbf{c}$ are contained in $\mathbf{Y}$. We also know that there are no edges from any state variable $S_j \notin \mathbf{Y}$ to any state variable $S_i \in \mathbf{Y}$; if there were, state variable S_j would have been included in $\mathbf{Y}$. It follows that the option SMDP $\mathcal{M}_o$ can ignore the values of state variables not in $\mathbf{Y}$ since they have no influence on the variables in $\mathbf{C}$, whose values we want to set to $\mathbf{c}$.

More formally, we can define a partition that satisfies the stochastic substitution property and is reward respecting (cf. Section 3.5), and thus guaranteed to preserve an optimal solution to the option SMDP $\mathcal{M}_o$.

Theorem 1 *The projection function $f_{\mathbf{Y}}$ induces a partition $\Lambda_{\mathbf{Y}}$ of S that has the stochastic substitution property and is reward respecting.*

Proof The projection function $f_{\mathbf{Y}}$ induces a partition $\Lambda_{\mathbf{Y}}$ of S such that two states $\mathbf{s}_1$ and $\mathbf{s}_2$ belong to the same block if and only if $f_{\mathbf{Y}}(\mathbf{s}_1) = f_{\mathbf{Y}}(\mathbf{s}_2)$, that is, if $\mathbf{s}_1$ and $\mathbf{s}_2$ assign exactly the same values to state variables in $\mathbf{Y}$. Let $\mathbf{y}_\lambda$ denote the assignment to $\mathbf{Y}$ of states in block λ of the induced partition, that is, for each state $\mathbf{s} \in \lambda$, we have that $f_{\mathbf{Y}}(\mathbf{s}) = \mathbf{y}_\lambda$. Then for each pair of states $(\mathbf{s}_1, \mathbf{s}_2) \in S^2$, each action $a \in A$, and each block $\lambda \in \Lambda_{\mathbf{Y}}$, $[\mathbf{s}_1]_{\Lambda_{\mathbf{Y}}} = [\mathbf{s}_2]_{\Lambda_{\mathbf{Y}}}$ implies that

$$\begin{aligned}
\sum_{\mathbf{s}\in\lambda} P(\mathbf{s} \mid \mathbf{s}_1, a) &= \sum_{\mathbf{s}\in\lambda} P(f_{\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_1, a) P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_1, a) = \\
&= \sum_{\mathbf{s}\in\lambda} P(\mathbf{y}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_1), a) P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_1, a) = \\
&= P(\mathbf{y}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_1), a) \sum_{\mathbf{s}\in\lambda} P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_1, a) = \\
&= P(\mathbf{y}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_2), a) \sum_{\mathbf{s}\in\lambda} P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_2, a) = \\
&= \sum_{\mathbf{s}\in\lambda} P(\mathbf{y}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_2), a) P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_2, a) = \\
&= \sum_{\mathbf{s}\in\lambda} P(f_{\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_2, a) P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_2, a) = \sum_{\mathbf{s}\in\lambda} P(\mathbf{s} \mid \mathbf{s}_2, a).
\end{aligned}$$

The equality $\sum_{\mathbf{s}\in\lambda} P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_1, a) = \sum_{\mathbf{s}\in\lambda} P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}_2, a)$ follows from the fact that as we sum over states in λ, we go through every possible assignment of values to state variables in the set $\mathbf{S} - \mathbf{Y}$, so in fact, $\sum_{\mathbf{s}\in\lambda} P(f_{\mathbf{S}-\mathbf{Y}}(\mathbf{s}) \mid \mathbf{s}', a) = 1$ for each state $\mathbf{s}' \in S$. It follows that the partition $\Lambda_{\mathbf{Y}}$ induced by $f_{\mathbf{Y}}$ has the stochastic substitution property.

In general, the partition $\Lambda_{\mathbf{Y}}$ induced by $f_{\mathbf{Y}}$ is not reward respecting with respect to the expected reward function R of the original MDP. However, recall that the expected reward function R_o of option o is independent of the expected reward function R of the original MDP. To form a reduced option SMDP it is sufficient that the partition $\Lambda_{\mathbf{Y}}$ is reward respecting with respect to R_o. R_o is defined as -1 everywhere except when the process leaves the initiation set of option o. The initiation set of option o is determined by the state variables in $\mathbf{Z} \subseteq \mathbf{Y}$, so whether or not the process leaves the initiation set depends only on those state variables. It follows that $\Lambda_{\mathbf{Y}}$ is reward respecting with respect to R_o. ■

VISA goes a step further and forms the partition $\Lambda_{\mathbf{Z}}$ induced by the projection $f_{\mathbf{Z}}$. In other words, the option SMDP of option o ignores all state variables not in strongly connected components for which the value of at least one state variable appears in the context $\mathbf{c}$ associated with option o.

Theorem 2 *The projection function $f_{\mathbf{Z}}$ induces a partition $\Lambda_{\mathbf{Z}}$ of S that is reward respecting and has the stochastic substitution property if and only if for each pair of states $\mathbf{s}_1, \mathbf{s}_2 \in S^2$, each option $o' \in O_o$, and each block $\lambda \in \Lambda_{\mathbf{Z}}$, $[\mathbf{s}_1]_{\Lambda_{\mathbf{Z}}} = [\mathbf{s}_2]_{\Lambda_{\mathbf{Z}}}$ implies that $P_o(\mathbf{z}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_1), o') = P_o(\mathbf{z}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_2), o')$, where $\mathbf{z}_\lambda$ is the assignment of values to $\mathbf{Z}$ of states in block λ.*

Proof $\Lambda_{\mathbf{Z}}$ is still reward respecting with respect to R_o. However, a state variable in $\mathbf{Y}-\mathbf{Z}$ may influence the state variables in $\mathbf{Z}$, so $\Lambda_{\mathbf{Z}}$ does not always have the stochastic substitution property. We can write the sum $\sum_{\mathbf{s}\in\lambda} P_o(\mathbf{s} \mid \mathbf{s}_1, o')$ as

$$\begin{aligned}
\sum_{\mathbf{s}\in\lambda} P_o(\mathbf{s} \mid \mathbf{s}_1, o') &= \sum_{\mathbf{s}\in\lambda} P_o(f_{\mathbf{Z}}(\mathbf{s}) \mid \mathbf{s}_1, o') P_o(f_{\mathbf{S}-\mathbf{Z}}(\mathbf{s}) \mid \mathbf{s}_1, o') = \\
&= \sum_{\mathbf{s}\in\lambda} P_o(\mathbf{z}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_1), o') P_o(f_{\mathbf{S}-\mathbf{Z}}(\mathbf{s}) \mid \mathbf{s}_1, o') = \\
&= P_o(\mathbf{z}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_1), o') \sum_{\mathbf{s}\in\lambda} P_o(f_{\mathbf{S}-\mathbf{Z}}(\mathbf{s}) \mid \mathbf{s}_1, o') = \\
&= P_o(\mathbf{z}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_1), o').
\end{aligned}$$

Using the same calculations, we obtain $\sum_{\mathbf{s}\in\lambda} P_o(\mathbf{s} \mid \mathbf{s}_2, o') = P_o(\mathbf{z}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_2), o')$. $\Lambda_{\mathbf{Z}}$ has the stochastic substitution property if and only if for each pair of states $(\mathbf{s}_1, \mathbf{s}_2) \in S^2$, each option $o' \in O_o$, and each block $\lambda \in \Lambda_{\mathbf{Z}}$, $[\mathbf{s}_1]_{\Lambda_{\mathbf{Z}}} = [\mathbf{s}_2]_{\Lambda_{\mathbf{Z}}}$ implies that $P_o(\mathbf{z}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_1), o') = P_o(\mathbf{z}_\lambda \mid f_{\mathbf{Y}}(\mathbf{s}_2), o')$, where $f_{\mathbf{Y}}(\mathbf{s}_1) = f_{\mathbf{Y}}(\mathbf{s}_2)$ does not hold in general. ■

From the work of Dean and Givan (1997) and Theorem 1 it follows that the partition $\Lambda_{\mathbf{Y}}$ induces a reduced SMDP that preserves optimality. Since the reduced SMDP can have far fewer state-action pairs than the original option SMDP, it can be significantly easier to solve, resulting in an important reduction in complexity. In addition, it follows from Theorem 2 that the partition $\Lambda_{\mathbf{Z}}$ induces a reduced SMDP that preserves optimality if and only if for each exit option $o' \in O_o$, state variables in $\mathbf{Y}-\mathbf{Z}$ do not influence the state variables in $\mathbf{Z}$ as a result of executing o'. Instead of solving the option SMDP directly, VISA solves the reduced SMDP induced by the partition $\Lambda_{\mathbf{Z}}$, which can have even fewer state-action pairs than the reduced SMDP induced by $\Lambda_{\mathbf{Y}}$.

Because of the way exits are defined, the exit options discovered by VISA often satisfy Theorem 2. For example, consider the exit option $\overline{H} \rightarrow H$ in the coffee task. After exit transformations, $\mathbf{Z} = \{S_{\mathtt{C}}\}$ and $\mathbf{Y} = \{S_{\mathtt{L}}, S_{\mathtt{C}}\}$, so $\mathbf{Y}-\mathbf{Z} = \{S_{\mathtt{L}}\}$. The options in the set O_o are $\overline{C} \rightarrow C$ and $C \rightarrow \overline{C}$, with associated exits $\langle (S_{\mathtt{L}} = L), \mathtt{BC} \rangle$ and $\langle (S_{\mathtt{L}} = \overline{L}), \mathtt{DC} \rangle$, respectively. As a result of executing action BC, the resulting value of state variable $S_{\mathtt{C}}$ depends on the previous value of state variable $S_{\mathtt{L}}$. However, as a result of executing the exit option $\overline{C} \rightarrow C$, the resulting value of $S_{\mathtt{C}}$ does not depend on the previous value of $S_{\mathtt{L}}$. Regardless of the previous value of $S_{\mathtt{L}}$, option $\overline{C} \rightarrow C$ always reaches the context $(S_{\mathtt{L}} = L)$ prior to executing BC, which causes the robot to buy coffee with non-zero probability. The same is true for exit option $C \rightarrow \overline{C}$, so it follows from Theorem 2 that the partition $\Lambda_{\mathbf{Z}}$ induced by $f_{\mathbf{Z}}$ has the stochastic substitution property.

If there exists a state variable in $\mathbf{Y}-\mathbf{Z}$ that influences a state variable in $\mathbf{Z}$, the partition $\Lambda_{\mathbf{Z}}$ does not have the stochastic substitution property. In other words, an optimal solution to the option SMDP $\mathcal{M}_o$ is not preserved in the reduced SMDP induced by the partition $\Lambda_{\mathbf{Z}}$. A solution to the reduced SMDP only corresponds to an approximate solution to $\mathcal{M}_o$. However, we believe that there is still a reason to perform state abstraction this way. The size of the partition $\Lambda_{\mathbf{Y}}$ may be exponentially larger than the size of $\Lambda_{\mathbf{Z}}$, so the difference in learning complexity may be significant in the two cases. We argue that the reduction in learning complexity often outweighs the loss of exact optimality.

To take even further advantage of structure, VISA stores the policies of options in the form of policy trees. The benefit of using a policy tree is that the number of leaves in the tree may be smaller

than the actual number of states. At each leaf of the policy tree, VISA stores action-values or, more accurately described, option-values, which indicate the utility of executing different options in states that map to that leaf. Recall that VISA maintains a transition graph, in the form of a tree, for each strongly connected component in the causal graph. The policy tree structure of an option can be constructed by merging the transition graph trees of strongly connected components that contain state variables whose values appear in the associated context. The policy tree structure induces a partition Λ_π such that $\Lambda_Z \leq \Lambda_\pi$, that is, Λ_π is guaranteed to have at most as many blocks as Λ_Z.

Another part of abstraction is reducing the number of options in the option set O_o of the option SMDP $\mathcal{M}_o$. If there are fewer options to select from, an autonomous agent can discover more quickly which option or options result in an optimal value for each block of the state partition. As we explained above, VISA finds strongly connected components that contain at least one state variable whose value appears in the context $\mathbf{c}$ associated with option o. The algorithm fills the option set O_o with options that change the values of state variables in those strongly connected components. The algorithm also includes options that leave the initiation sets of options in O_o. It is not necessary to include other options in O_o since they do not have any impact on reaching the context $\mathbf{c}$ associated with option o. Thus, VISA can reduce the number of options of each option SMDP, further reducing the complexity of learning.

4.6 Task Option

VISA also introduces an option, which we call the *task option*, associated with the reward node in the component graph of the task. The purpose of the task option is to approximate a solution to the original MDP. However, instead of being a policy that selects among primitive actions, the policy of the task option selects among the exit options introduced by VISA. Thus, the policy of the task option represents a hierarchical solution to the task that takes advantage of the exit options to set the values of relevant state variables in such a way as to maximize expected reward.

The task option is admissible everywhere, that is, its initiation set equals S. If the task is finite-horizon, the termination condition function β is defined such that the task option terminates whenever the task is completed. If the task is infinite-horizon, β is defined such that the task option never terminates. To learn the task option policy, VISA constructs the option SMDP corresponding to the task option using the same strategy it uses for the exit options. However, the expected reward function of the task option SMDP is equal to the expected reward function of the original MDP. For determining the policy of the task option, the expected reward for executing an exit option o is defined as the sum of discounted reward of the primitive actions selected during the execution of o.

VISA also performs state abstraction for the task option SMDP in the same way it does for exit options. First, VISA finds the set of state variables $\mathbf{Z} \subseteq \mathbf{V}$ in strongly connected components with edges to the expected reward node in the component graph. VISA performs state abstraction by ignoring the values of state variables not in $\mathbf{Z}$, and it constructs a policy tree structure to further reduce the number of states. In addition, the option set of the task option SMDP only includes exit options that change the values of state variables in $\mathbf{Z}$.

The task option is the only reward-dependent component of VISA. If several tasks share the same set of state variables and actions, the same set of exit options apply to all of these tasks. For example, this would apply to a workshop environment with a fixed number of objects where a robot may be instructed to perform several tasks, such as moving objects. VISA can construct a causal graph that is common to all tasks by excluding the expected reward node, and use the graph to

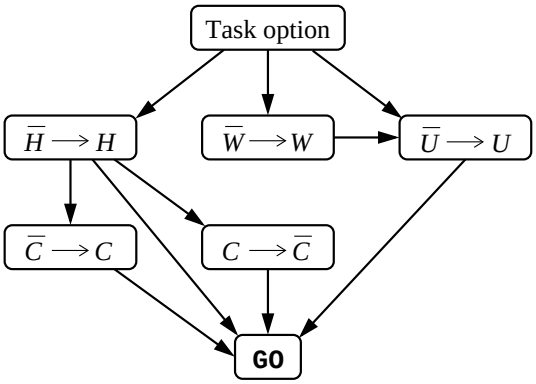


Figure 5: The hierarchy of options discovered by VISA in the coffee task

introduce exit options as before. When provided with the expected reward function of a specific task, VISA can construct the task option by overlaying the expected reward node onto the existing causal graph. This way, VISA just needs to learn the policies of the exit options once and can reuse them throughout all tasks. This facilitates transfer of knowledge between tasks in the same environment.

4.7 Option Hierarchy

The task option together with the exit options introduced by VISA implicitly define a hierarchy of options in which the options on one level selects options on the next lower level. Recall that for an option associated with a node in the component graph, the option SMDP only includes options that change state variables in components that have edges to that node. In other words, the component graph determines the structure of the option hierarchy. Since the component graph is guaranteed to contain no cycles, the option hierarchy is well-defined, and it is not possible for an option to execute itself, either directly or indirectly.

Figure 5 shows the hierarchy of options that VISA comes up with in the coffee task. The option hierarchy is determined by the component graph of the coffee task, illustrated in Figure 2. The task option always sits at the top level of the hierarchy. There are two components with edges to the expected reward node, namely $S_{\mathtt{H}}$ and $S_{\mathtt{W}}$. Option $\overline{H} \to H$ changes the value of $S_{\mathtt{H}}$, and option $\overline{W} \to W$ changes the value of $S_{\mathtt{W}}$. In addition, option $\overline{U} \to U$ causes the process to leave the initiation set of $\overline{W} \to W$. In other words, the task option selects among the three options $\overline{H} \to H$, $\overline{W} \to W$, and $\overline{U} \to U$.

In turn, there are two components with edges to the component $S_{\mathtt{H}}$, namely $S_{\mathtt{L}}$ and $S_{\mathtt{C}}$. The primitive action `GO` changes the value of $S_{\mathtt{L}}$, while options $\overline{C} \to C$ and $C \to \overline{C}$ change the value of $S_{\mathtt{L}}$. Consequently, option $\overline{H} \to H$ selects among `GO`, $\overline{C} \to C$, and $C \to \overline{C}$. There are also two components with edges to $S_{\mathtt{W}}$, namely $S_{\mathtt{U}}$ and $S_{\mathtt{R}}$. Option $\overline{U} \to U$ changes the value of $S_{\mathtt{U}}$, while no option changes the value of $S_{\mathtt{R}}$. Thus, $\overline{W} \to W$ can only select $\overline{U} \to U$. Finally, there are edges between $S_{\mathtt{L}}$ and $S_{\mathtt{C}}$ as well as between $S_{\mathtt{L}}$ and $S_{\mathtt{U}}$, so options $\overline{C} \to C$, $C \to \overline{C}$, and $\overline{U} \to U$ all select among the primitive action `GO` that changes the value of $S_{\mathtt{L}}$.

4.8 Merging Strongly Connected Components

If there are many context-action pairs that cause changes, it is not particularly useful to introduce an option for each of them. Instead, VISA merges two strongly connected components that are linked by too many exits. After VISA identifies exits for a strongly connected component, the algorithm counts the number of exits identified. If the number of exits is larger than a threshold, VISA merges the strongly connected component with one or several of its parents. The merge operation places all state variables in the strongly connected components into a single component and recomputes the exits of the new component. As a result, the complexity of solving an associated subtask increases because there are more state variables in the set $\mathbf{Z}$. However, the number of subtasks decreases since there are fewer exits as a result.

4.9 Summary of the Algorithm

In summary, VISA first constructs the causal graph to determine how state variables are related. If there are cycles in the causal graph, it is not possible to decompose the task, so VISA gets rid of cycles by identifying the strongly connected components. For each strongly connected component, VISA uses the DBN model to identify exits, that is, pairs of variable values and actions that cause the value of some state variable in the component to change. For each exit, VISA constructs the components of an exit option, whose purpose it is to bring about the corresponding variable value change using a minimum number of options. At the top level, VISA constructs a task option that uses the exit options to approximate a solution to the original MDP. VISA uses reinforcement learning techniques to learn a policy of each option introduced. Algorithm 3 provides pseudo-code for VISA.

4.10 Limitations of the Algorithm

VISA only decomposes a task if there are two or more strongly connected components in the causal graph of the task. Otherwise, VISA cannot exploit conditional independence between state variables to identify options. Since the option SMDPs are stand-alone, the hierarchy discovered by VISA enables recursive optimality at best, as opposed to hierarchical optimality (Dietterich, 2000a). In addition, VISA works best when there are relatively few exits that cause the values of state variables in a strongly connected component to change.

Furthermore, the option-specific state abstraction performed by VISA is independent of the way options are formed. Given access to the causal graph, VISA makes it possible to efficiently perform state abstraction for any option whose goal is to reach a context specified by an assignment of values to a subset of the state variables. For the purpose of state abstraction, it does not matter how an autonomous agent determines that it is useful to reach that specific context. In other words, the state abstraction part of VISA could be combined with other techniques for discovering useful activities, as long as they are of the required form.

State abstraction for the task option is particularly efficient in tasks for which the expected reward depends on only a few state variables. If most state variables influence reward, learning the task option policy requires almost the same effort as learning a policy over primitive actions. Also note that exit options attempt to change the value of a state variable using as few primitive actions as possible. In terms of expected reward, such a behavior may not be optimal, since each action does not necessarily incur the same expected reward. In some tasks, it would be necessary to choose a

Algorithm 3 VISA

Input: DBN model of a factored MDP $\mathcal{M}$ with set of state variables **S**
construct the causal graph of the task
compute the strongly connected components of the causal graph
perform a topological sort of the strongly connected components
for each strongly connected component **SC** $\subseteq$ **S** in topological order
    identify exits that cause the values of state variables in **SC** to change
    while the number of exits exceeds a threshold
        merge **SC** with a parent strongly connected component
        label the new strongly connected component **SC** and recompute the exits
    for each exit $\langle \mathbf{c}, a \rangle$ of the strongly connected component **SC**
        perform any possible exit transformations
        compute the set **Z** of influencing state variables
        construct an initiation set I
        construct a termination function β using the context **c**
        construct a policy tree by merging transition graphs of parent components
        let S_o be the leaves of the policy tree
        let O_o be the set of options that changes values of state variables in **Z**
        let Ψ_o be defined by the initiation sets of options in O_o
        define R_o as -1 everywhere except when the context **c** is unreachable
        let P_o be undefined
        construct the option SMDP $\mathcal{M}_o = \langle S_o, O_o, \Psi_o, P_o, R_o \rangle$
        construct an exit option $o = \langle I, \pi, \beta \rangle$, where $\pi =$ optimal policy of $\mathcal{M}_o$
    construct the transition graph of the strongly connected component **SC**
construct a task option corresponding to the original task
use reinforcement learning techniques to learn the policy of each option

different expected reward function for exit option SMDPs to avoid large negative rewards. However, VISA is most efficient in tasks for which few state variables influence reward. In such tasks, lower-level variables do not influence reward, so the optimal behavior is to achieve the precondition of an exit as quickly as possible. For example, in the coffee task, the option hierarchy discovered by VISA enables optimality, since none of the exit options choose suboptimal actions.

4.11 Experiments

We ran several experiments to test the performance of VISA. Since VISA uses the DBN model of factored MDPs, it would be unfair to compare it to algorithms that begin with less prior knowledge. Instead, we compared VISA to two algorithms that also assume knowledge of the DBN model: SPUDD (Hoey et al., 1999) and symbolic Real-Time Dynamic Programming, or sRTDP (Feng et al., 2003). SPUDD is a more efficient version of policy iteration that takes advantage of the compactness of the DBN model to compute the value function in the form of an algebraic decision diagram, or ADD.

sRTDP is an online planning algorithm that, at each step, constructs a set of states that are similar to the current state according to one of two heuristics, called value and reach. The algorithm uses the DBN model to determine the set of possible next states, and performs a masked backup of the value function restricted to the set of current and next states. The algorithm then selects for execution one of the actions whose current action-value estimate is highest. sRTDP stores the value function in the form of ADDs, and uses SPUDD to perform the masked value backup at each step. SPUDD includes a mechanism that limits the size of the ADDs, divides the state variables into subsets, and decomposes the value backup into several smaller computations. In our implementation, we did not allow the size of the ADDs to exceed 10,000 nodes.

We performed experiments with each algorithm in four tasks: the coffee task, the Taxi task, the Factory task (Hoey et al., 1999), and a simplified version of the autonomous guided vehicle (AGV) task of Ghavamzadeh and Mahadevan (2001). In the Taxi task (Dietterich, 2000a), a taxi has to pick up a passenger from their location and deliver them to their destination. The Taxi task has 600 states and 6 actions. In the Factory task (Hoey et al., 1999), a robot has to assemble a component made of two objects. Before assembly is possible, the robot has to perform various operations on each of the two objects, such as shaping, smoothing, polishing and painting. The objects can then be connected either by drilling and bolting or by gluing. The task is described by 17 binary variables, for a total of approximately 130,000 states, and the robot has 14 actions.

The Factory task was designed as a infinite-horizon task whose reward function assigns partial reward in many states. When the component has been assembled, the optimal policy repeatedly selects the same action in the same state for maximal reward. However, when using reinforcement learning to learn a policy, it is necessary to reset the state once in a while to ensure that a policy is learned for all states. When the state is reset, the positive reward as a result of assembling the component is not large enough to prevent the learning agent from exploiting the partial reward in other states. For this reason, we redefined the reward function of the Factory task to only assign positive reward when the component has been assembled. This neither affects the optimal policy nor the set of state variables that influence reward.

In the AGV task (Ghavamzadeh and Mahadevan, 2001), an autonomous guided vehicle (AGV) has to transport parts between machines in a manufacturing workshop. We simplified the task by reducing the number of machines from 4 to 2 and setting the processing time of machines to 0 to

make the task fully observable. The resulting task is illustrated in Figure 6 and has approximately 75,000 states. The goal of the AGV is to proceed to the load station, pick up a random part i, transport it to the drop-off location D_i of machine M_i, drop it off, then proceed to the pick-up location P_i of machine M_i, pick up the processed part, transport it to the warehouse, and finally drop it off. The AGV is restricted to move unidirectionally along the arrows in the figure, and has to ensure that at least one part of each type is stored in the warehouse. The set of state variables describing the task is $\mathbf{S} = \{S_x, S_y, S_f, S_h, S_{d1}, S_{p1}, S_{d2}, S_{p2}, S_{a1}, S_{a2}\}$, where S_x and S_y represent the location of the AGV, S_f the direction it is facing, S_h the part it is holding, S_{di} the number of parts at the drop-off location D_i of machine M_i, S_{pi} the number of parts at the pick-up location P_i, and S_{ai} whether a part of type i is present in the warehouse. The AGV has 6 actions: move in the direction it is facing, turn left or right, drop off a part, pick up a part, and idle. Even though we simplified the AGV task, its size still presents a challenge for algorithms that discover activities.

Each graph in the results illustrates the average reward over 100 learning runs with each algorithm. Since the algorithms are fundamentally different, we compared the actual running time in milliseconds. The graphs for VISA include the time it takes to decompose the factored MDP. We used SMDP Q-learning to learn the option policies, which reduces to regular Q-learning for policies that select among primitive actions. We set the discount factor to $\gamma = 0.9$ and initially used a step-size parameter $\alpha = 0.05$, which we decayed at regular time intervals by multiplying the current step-size parameter by 0.9. The policies of all options, including the task option, were learned in parallel. Prior to executing, sRTDP computes action ADDs; the graphs include the time it takes to do this. We report results of both heuristics (value and reach) used by sRTDP to construct the set of similar states. All algorithms were coded in Java, except that the CUDD library (written in C) was used to manipulate ADDs through the Java Native Interface.

SPUDD is conceptually different from VISA and sRTDP in that it does not require actual experience in the domain. Instead, it uses the transition probabilities and expected reward of the DBN model to repeatedly update the policy off-line. Consequently, it is not possible to measure the reward received as a result of executing actions in the environment. To evaluate the running time of SPUDD, we first recorded the time elapsed between each iteration of the algorithm. After each iteration, we recorded the current policy and stored it in memory. When policy iteration converged, we retrieved each stored policy from memory. For each policy, we ran experiments in the domain and selected actions according the policy. The average reward of each experiment appears at the time at which the policy was recorded. Just as for VISA and sRTDP, we ran 100 trials and plotted the average reward across trials.

In the results, we also present a comparison of the size of the state partitions produced by SPUDD and VISA. The size of the state partition produced by SPUDD equals the number of leaves in the ADD used to represent the value function. In contrast, the size of the state partition produced by VISA equals the total number of leaves in the policy trees of exit options, including the task option. Since the state partition produced by VISA does not necessarily preserve optimality, it is often smaller than that of SPUDD.

4.12 Results

Figure 7 illustrates the results of the experiments in the coffee task. Since the coffee task is very small, all algorithms converge quickly to an optimal policy, although SPUDD has a slight edge over the others. The state partition produced by SPUDD contains 48 aggregated states, as compared to

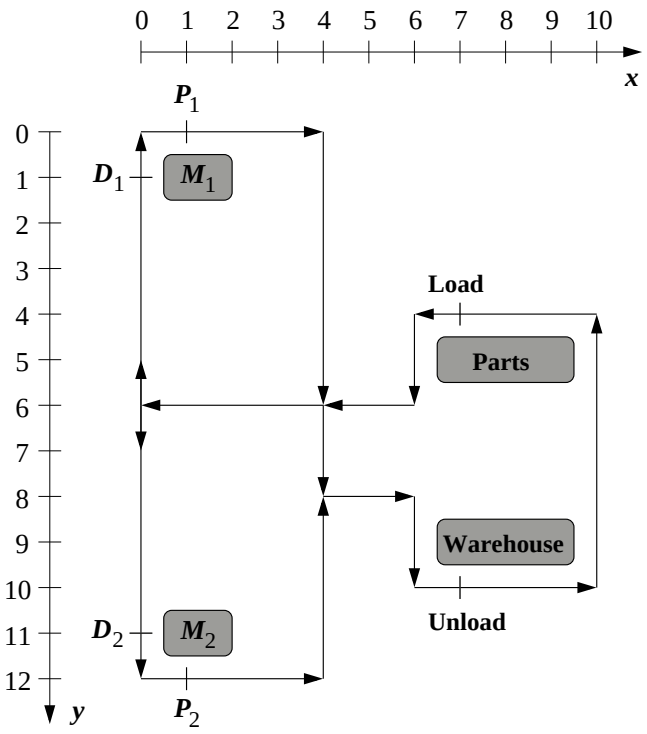


Figure 6: Illustration of the AGV task

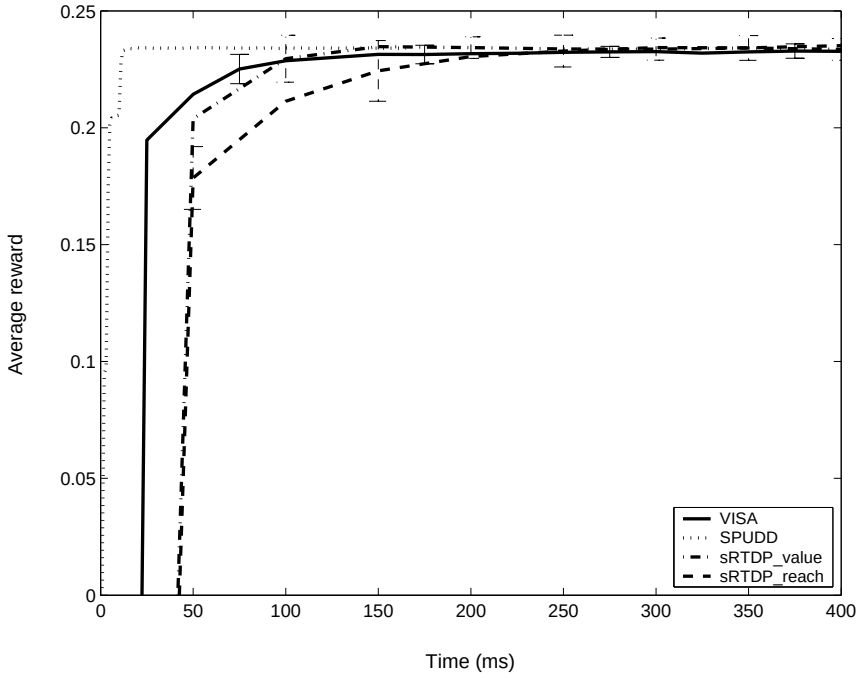


Figure 7: Results of learning in the coffee task

the $2^6 = 64$ total states of the task. In contrast, the state partition produced by VISA contains a total of 20 aggregated states.

Figure 8 illustrates the results of the experiments in the Taxi task. In this task, VISA and SPUDD perform significantly better than sRTDP, with SPUDD slightly faster than VISA. The state partition of SPUDD contains 525 aggregated states, almost as many as the 600 states of the original task. In comparison, the state partition produced by VISA contains a total of 106 aggregated states.

Figure 9 illustrates the results of the experiments in the Factory task. Again, SPUDD and VISA have a similar convergence times, although it appears as if VISA converges to a slightly suboptimal policy. The Factory task poses a significant challenge to online learning algorithms since a lot of actions undo the effect of other actions, making it difficult to achieve the objective. We believe this is the reason that sRTDP is struggling to converge quickly. The reason VISA does so well is that the hierarchical decomposition restricts the policy to select between options that achieve relevant subgoals, guiding the process towards the ultimate objective. The state partition of SPUDD contains 4,550 states, significantly less than the $2^{17} \approx 130{,}000$ states of the task. The state partition produced by VISA is even smaller, containing 2,620 aggregated states.

Figure 10 illustrates the results of the experiments in the AGV task of VISA and sRTDP using the reach heuristic. VISA decomposes the task in roughly 6 seconds and learning converges after 20 seconds. In comparison, it took SPUDD more than 4 minutes to converge to an optimal policy, and its performance is not shown in Figure 10. sRTDP using the reach heuristic completes the task a few times within the first minute of running time but convergence is much slower than for VISA. During our experiments, sRTDP using the value heuristic failed to complete the task even once within the

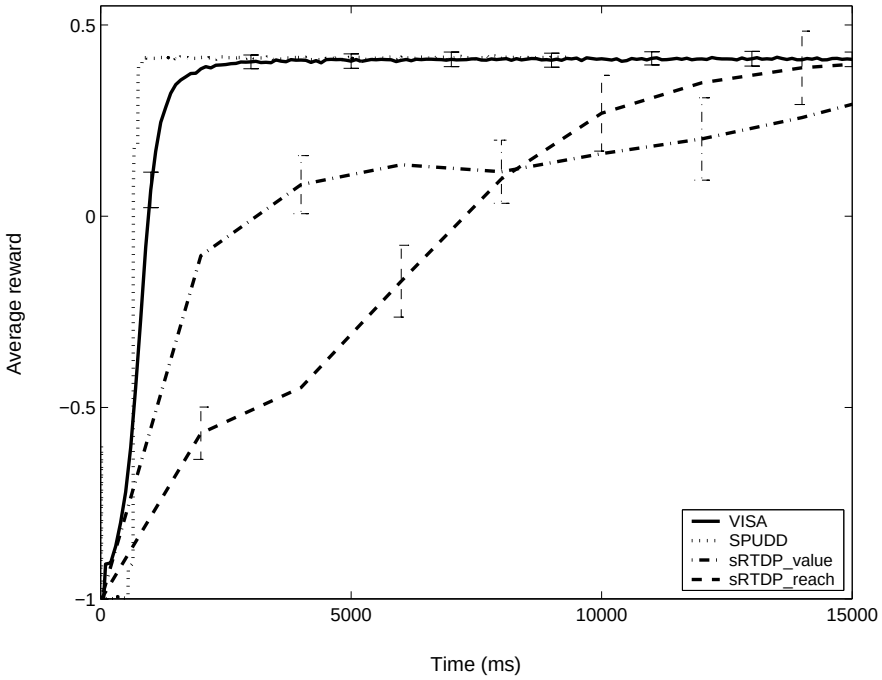


Figure 8: Results of learning in the Taxi task

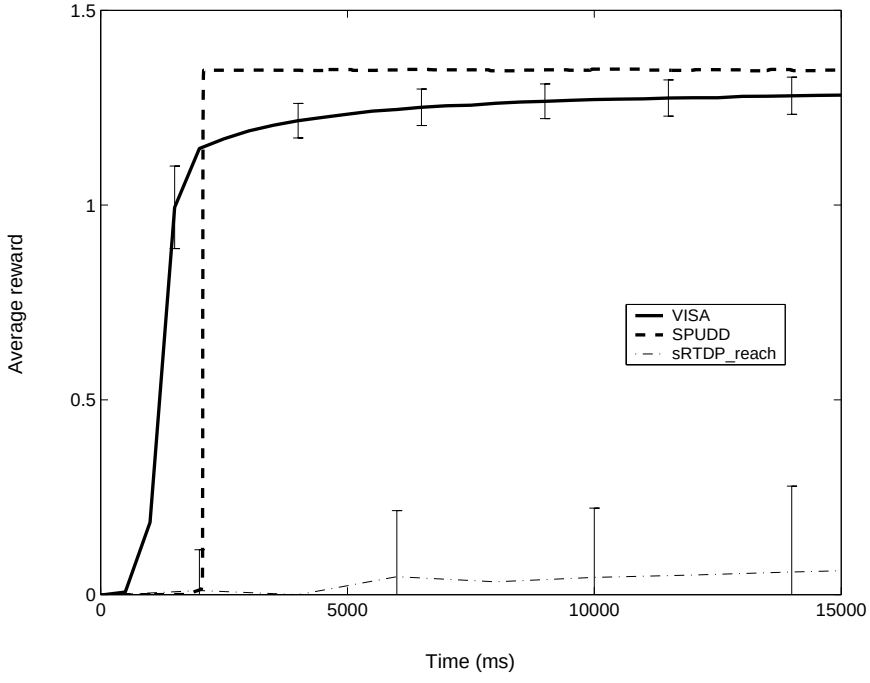


Figure 9: Results of learning in the Factory task

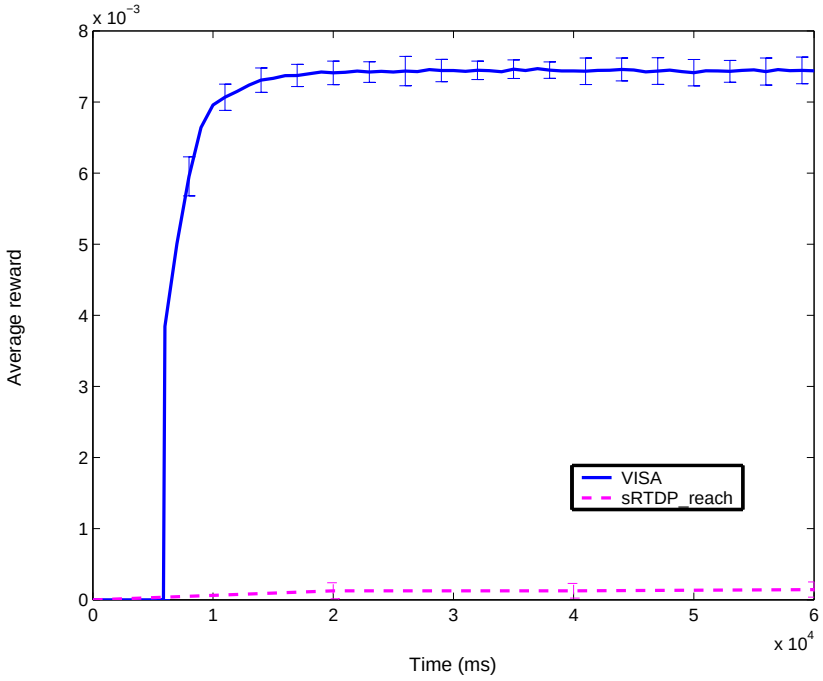


Figure 10: Results of learning in the AGV task

first 15 minutes. The state partition of SPUDD contains 11,096 states, compared to the 75,000 states of the task, while the state partition of VISA contains 5,996 aggregated states.

The results of the experiments illustrate the power of hierarchical decomposition when combined with option-specific abstraction. Even though SPUDD and sRTDP take advantage of task structure and are empirically faster than regular reinforcement learning algorithms, they still suffer from the curse of dimensionality as the size of the state space grows. On the other hand, VISA decomposes the original tasks into smaller, stand-alone tasks that are easier to solve without ever enumerating the state space. Instead, the complexity of the decomposition is polynomial in the size of the conditional probability trees of the DBN model. Each stand-alone task only distinguishes among values of a subset of the state variables, which means that the complexity of learning does not necessarily increase with the number of state variables. Evidently, the advantage offered by VISA varies between tasks and is dependent on the causal graph structure.

5. Constructing Compact Option Models

VISA computes option policies by first constructing the option SMDP, $\mathcal{M}_o$, of each exit option o. Since VISA does not have access to an estimate of the transition probability function, P_o, it cannot use a planning algorithm to solve the option SMDP. Instead, VISA uses SMDP Q-learning to learn the option policies, which does not require knowledge of transition probabilities as long as the algorithm has access to a real system and enough time. If VISA had access to DBN models that compactly describe the transition probabilities as a result of executing options, it would be possible to apply existing planning algorithms that exploit the DBN models to efficiently solve the

option SMDPs. Access to DBN models of options would open up new possibilities for learning and planning with options.

In this section, we develop ideas for constructing compact models of the exit options introduced by VISA. Current techniques for constructing option models require the state space to be enumerated. Since the goal of VISA is to reduce the complexity of learning by ignoring a subset of state variables, we want to avoid enumerating the state space. Instead, we define theoretical properties of partitions that preserve the transition probabilities and expected reward of options. We then discuss how to construct representations of transition probabilities and expected reward using partitions with these properties. In many cases, the partitions contain far fewer blocks than the number of states, resulting in a compact representation.

5.1 Multi-Time Option Models

Sutton et al. (1999) defined the multi-time model of an option $o = \langle I, \pi, \beta \rangle$ as

$$P(s' \mid s, o) = \sum_{t=1}^{\infty} \gamma^t P(s', t \mid s, o), \tag{2}$$

$$R(s, o) = E\{ \sum_{k=1}^{t} \gamma^{k-1} R(s_k, a_k) \mid s_1 = s \}, \tag{3}$$

where t is the random duration until o terminates and $P(s', t \mid s, o)$ is the probability that o terminates in state $s' \in S$ after t time steps when executed in state $s \in S$. The expectation in Equation 3 is taken over the distribution of state-action pairs (s_k, a_k), $k \in [1, t]$. This distribution is determined by the functions $P(s_{k+1} \mid s_k, a_k)$, $\pi(s_k, a_k)$, and $\beta(s_k)$. We refer to the terms $P(s' \mid s, o)$ as *discounted probabilities* since they do not sum to 1 for $\gamma < 1$. However, the multi-time model enables learning and planning with options as single units, which Sutton et al. (1999) call SMDP value learning and SMDP planning, respectively.

It is possible to use dynamic programming to compute the multi-time model in Equations 2 and 3. We can set up the Bellman form of the equations in which each term is a function of the terms at the next time step:

$$P(s' \mid s, o) = \gamma \sum_{a \in A} \pi(s, a) \left[P(s' \mid s, a) \beta(s') + \sum_{s'' \in S} P(s'' \mid s, a)(1 - \beta(s'')) P(s' \mid s'', o) \right], \tag{4}$$

$$R(s, o) = \sum_{a \in A} \pi(s, a) \left[R(s, a) + \gamma \sum_{s' \in S} P(s' \mid s, a)(1 - \beta(s')) R(s', o) \right]. \tag{5}$$

Let us label each state with a unique subscript $i \in \{1, \ldots, |S|\}$. Let P^a, $a \in A$, be the transition matrix for action a whose entry (i, j) is $P(s_j \mid s_i, a)$, and let P^o be the corresponding matrix for option o. Let Π^a, $a \in A$, be the diagonal matrix whose entry (i, i) is $\pi(s_i, a)$, and let B be the diagonal matrix whose entry (i, i) is $\beta(s_i)$. Let R^a, $a \in A$, be the vector whose ith entry is $R(s_i, a)$, and let R^o be the corresponding vector for option o. To avoid confusion with the option initiation set I, we use E to denote the identity matrix. Then we can write Equations 4 and 5 respectively in the following forms:

$$P^o = \gamma \sum_{a \in A} \Pi^a P^a (B + (E - B) P^o), \tag{6}$$

$$R^o = \sum_{a \in A} \Pi^a (R^a + \gamma P^a (E - B) R^o). \tag{7}$$

5.2 Multi-Time Models for Exit Options

Recall that an exit option o is associated with an exit $\langle \mathbf{c}, a \rangle$, composed of a context $\mathbf{c}$ and an action a. Unlike regular options, the exit option executes action a following termination in context $\mathbf{c}$. In addition, if o is executed in a state that already satisfies context $\mathbf{c}$, o immediately terminates and executes action a. As a consequence, it is necessary to modify the multi-time model to suit exit options. The multi-time model of an exit option o has the following form:

$$P^o = \gamma(BP^a + (E - B) \sum_{a' \in A} \Pi^{a'} P^{a'} P^o), \tag{8}$$

$$R^o = BR^a + (E - B) \sum_{a' \in A} \Pi^{a'} (R^{a'} + \gamma P^{a'} R^o), \tag{9}$$

where a is the action of the exit associated with o. The above definition assumes that o always terminates in a state that satisfies context $\mathbf{c}$, so that a is always executed following termination.

Because we modified the multi-time model to handle the case of exit options, we need to show that the discounted probabilities, P^o, and expected reward, R^o, associated with an exit option o are well-defined under the condition that o is guaranteed to eventually terminate.

Definition 3 *An option o is proper if for each state $s_i \in I$, o eventually terminates with probability 1 when executed in s_i.*

Definition 3 imposes a restriction on the policy π and termination condition function β of an option o.

Theorem 4 *For a proper option o, the systems of linear equations in Equations 8 and 9 are consistent and have unique solutions.*

The unknown quantities that we want to solve for are P^o and R^o. If we move the unknowns to the left-hand side of Equations 8 and 9 we obtain the following systems of equations:

$$\left[E - \gamma(E - B) \sum_{a' \in A} \Pi^{a'} P^{a'} \right] P^o = \gamma BP^a, \tag{10}$$

$$\left[E - \gamma(E - B) \sum_{a' \in A} \Pi^{a'} P^{a'} \right] R^o = BR^a + (E - B) \sum_{a' \in A} \Pi^{a'} R^{a'}. \tag{11}$$

Note that the unknown quantities P^o and R^o are multiplied by the same matrix $M = E - \gamma(E - B) \sum_{a' \in A} \Pi^{a'} P^{a'}$. The systems of linear equations in Equations 10 and 11 are consistent and have unique solutions if and only if matrix M is invertible, that is, if and only if the determinant of M is non-zero. The proof of Theorem 4 appears in Appendix A.

Since Equations 6 and 7 resemble the Bellman optimality equation in Equation 1, it is possible to use algorithms similar to value iteration and policy iteration to solve for P^o and R^o. Note, however, that we do not want to represent the matrices explicitly, since their size is proportional to the number of states. Instead, we can use decision trees or ADDs to compactly represent the matrices. The policy of an exit option is already in the form of a tree, and it is easy to construct a tree that represents the termination condition function β. SPUDD (Hoey et al., 1999) contains an efficient subroutine

that uses the DBN model to perform multiplication of a matrix with the transition probability matrix P^a without explicitly representing P^a. For a proper option o, iteratively performing the calculations on the right-hand side of Equations 8 and 9 will eventually converge to a compact representation of P^o and R^o.

5.3 Decomposition of the Option Model

In Section 4, we compared VISA with several algorithms that take advantage of the DBN model to compactly represent transition probabilities and expected reward. Even though these algorithms construct compact representations of the value function, VISA outperformed these algorithms in several tasks. The reason for this is that VISA introduces a set of subtasks and performs state abstraction for each subtask by ignoring irrelevant state variables, making each subtask easier to solve than the original task. It is possible to decompose computation of the multi-time option model in a similar way.

Recall that we approximate the transition probabilities of primitive actions as products of the conditional probabilities of each state variable $S_d \in \mathbf{S}$:

$$P(\mathbf{s}' \mid \mathbf{s}, a) \approx \prod_{S_d \in \mathbf{S}} P_d(f_{\{S_d\}}(\mathbf{s}') \mid f_{\mathbf{Pa}(S_d)}(\mathbf{s}), a).$$

The expression is an approximation for tasks in which there are dependencies between state variables at a same time step because our formalism does not account for such synchronous dependencies.

It is possible to approximate the terms of the multi-time model in a similar way. However, the multi-time model of an option o has two distributions that resemble transition probabilities: $P(\mathbf{s}' \mid \mathbf{s}, o)$, the discounted probability of transitioning from $\mathbf{s}$ to $\mathbf{s}'$ as a result of executing o; and $P(\mathbf{s}', t \mid \mathbf{s}, o)$, the exact probability of transitioning from $\mathbf{s}$ to $\mathbf{s}'$ in t time steps as a result of executing o. We can choose which of the two distributions to approximate.

If we choose to approximate $P(\mathbf{s}' \mid \mathbf{s}, o)$, we obtain the following approximation:

$$P(\mathbf{s}' \mid \mathbf{s}, o) \approx \prod_{S_d \in \mathbf{S}} P_d(f_{\{S_d\}}(\mathbf{s}') \mid f_{\mathbf{Pa}(S_d)}(\mathbf{s}), o).$$

Since we do not (yet) have access to a DBN model of option o, we assume that all state variables are parents of S_d, so $f_{\mathbf{Pa}(S_d)}(\mathbf{s}) = f_{\mathbf{S}}(\mathbf{s}) = \mathbf{s}$. We can compute the terms $P_d(f_{\{S_d\}}(\mathbf{s}') \mid \mathbf{s}, o)$ in the same way as the multi-time model:

$$P_d(f_{\{S_d\}}(\mathbf{s}') \mid \mathbf{s}, o) = \sum_{t=1}^{\infty} \gamma^t P_d(f_{\{S_d\}}(\mathbf{s}'), t \mid \mathbf{s}, o).$$

As a result, we obtain the following final approximation of Equation 2:

$$P(\mathbf{s}' \mid \mathbf{s}, o) \approx \prod_{S_d \in \mathbf{S}} \sum_{t=1}^{\infty} \gamma^t P_d(f_{\{S_d\}}(\mathbf{s}'), t \mid \mathbf{s}, o). \tag{12}$$

Equation 12 enables us to compute the conditional probabilities $P_d(v_d \mid \mathbf{s}, o)$ of the multi-time model separately for each state variable S_d. Here, $v_d \in \mathcal{D}(S_d)$ denotes one of the values of state variable S_d.

If we instead choose to approximate $P(\mathbf{s}', t \mid \mathbf{s}, o)$, we obtain the following alternative approximation of Equation (2):

$$P(\mathbf{s}' \mid \mathbf{s}, o) = \sum_{t=1}^{\infty} \gamma^t P(\mathbf{s}', t \mid \mathbf{s}, o) \approx \sum_{t=1}^{\infty} \prod_{S_d \in \mathbf{S}} \gamma^t P_d(f_{\{S_d\}}(\mathbf{s}'), t \mid \mathbf{s}, o). \tag{13}$$

Note that the difference between Equations 12 and 13 is the order of the summation and the product. As a result, Equation 12 assigns non-zero probability to events that could never occur, such as "the value of state variable $S_{\mathtt{L}}$ becomes L in 2 time steps, and the value of state variable $S_{\mathtt{W}}$ becomes W in 3 time steps." This event could never occur because an option cannot simultaneously terminate after 2 time steps and 3 time steps. In this sense, Equation 13 is a better approximation of $P(\mathbf{s}' \mid \mathbf{s}, o)$, but on the other hand, it does not enable us to compute a multi-time model of option o separately for each state variable. As we shall see, the ability to decompose the computation significantly reduces the complexity of computing the multi-time model. We believe that the reduction in complexity justifies the loss of accuracy, although we currently have no bounds on the approximation error. For this reason, we use Equation 12 as our approximation of Equation 2.

For each state variable S_d, each state $\mathbf{s} \in S$, and each value $v_d \in \mathcal{D}(S_d)$, we seek the term $P_d(v_d \mid \mathbf{s}, o)$ representing the probability of transitioning into a state that assigns v_d to S_d when o is executed in state $\mathbf{s}$. $P_d(v_d \mid \mathbf{s}, o)$ is given by the following equation:

$$P_d(v_d \mid \mathbf{s}, o) = \gamma(\beta(\mathbf{s}) P_d(v_d \mid \mathbf{s}, a) + (1 - \beta(\mathbf{s})) \sum_{a' \in A} \pi(\mathbf{s}, a') \sum_{\mathbf{s}' \in S} P(\mathbf{s}' \mid \mathbf{s}, a') P_d(v_d \mid \mathbf{s}', o)). \tag{14}$$

Let P_d^o be a $|S| \times |\mathcal{D}(S_d)|$ matrix whose entry (i, j) equals $P_d(j \mid \mathbf{s}_i, o)$. We can solve for P_d^o using the following system of equations:

$$P_d^o = \gamma(BP_d^a + (E - B) \sum_{a' \in A} \Pi^{a'} P^{a'} P_d^o), \tag{15}$$

where P_d^a is the equivalent of P_d^o for exit action a.

Lemma 5 *For a proper option o, the system of linear equations in Equation 15 is consistent and has a unique solution.*

Let us again move all unknowns to the left side of the equation to obtain

$$\left[E - \gamma(E - B) \sum_{a' \in A} \Pi^{a'} P^{a'} \right] P_d^o = \gamma B P_d^a. \tag{16}$$

The proof of Lemma 5 follows directly from the proof of Theorem 4 since the matrix $M = \left[E - \gamma(E - B) \sum_{a' \in A} \Pi^{a'} P^{a'} \right]$ that we need to invert to solve Equation 16 is the same as the matrix in Equations 10 and 11.

Instead of a single system of equations (8), we now have to solve one system of equations per state variable (15) to approximate the discounted probabilities P^o. Since the system of equations are similar, it appears as if little is gained, but as it turns out, the complexity of the computation may be dramatically lower. The matrix P_d^o only has one column per value in $\mathcal{D}(S_d)$, which is considerably less than the number of columns of P^o, even for a very compact state representation. This means that matrix multiplications can be carried out more efficiently. In addition, some state variables may be irrelevant for computing P_d^o, making the representation even more compact throughout the computation. Specifically, decomposing computation of the option model makes it possible to construct a different compact representation for each state variable.

5.4 Partitions

We formalize the ability to construct compact representations for each state variable using partitions. Recall that a partition Λ of the state set S that has the stochastic substitution property and is reward respecting induces a reduced MDP that preserves optimality. We define three more properties of partitions of S with respect to a factored MDP $\mathcal{M}$ and an option $o = \langle I, \pi, \beta \rangle$:

Definition 6 *A partition Λ of S is policy respecting if for each pair of states $(\mathbf{s}_i, \mathbf{s}_j) \in S^2$ and each action $a \in A$, $[\mathbf{s}_i]_\Lambda = [\mathbf{s}_j]_\Lambda$ implies that $\pi(\mathbf{s}_i, a) = \pi(\mathbf{s}_j, a)$.*

Definition 7 *A partition Λ of S is termination respecting if for each pair of states $(\mathbf{s}_i, \mathbf{s}_j) \in S^2$, $[\mathbf{s}_i]_\Lambda = [\mathbf{s}_j]_\Lambda$ implies that $\beta(\mathbf{s}_i) = \beta(\mathbf{s}_j)$.*

Definition 8 *A partition Λ of S is probability respecting of state variable S_d if for each pair of states $(\mathbf{s}_i, \mathbf{s}_j) \in S^2$, each action $a \in A$, and each value $v_d \in \mathcal{D}(S_d)$, $[\mathbf{s}_i]_\Lambda = [\mathbf{s}_j]_\Lambda$ implies that $P_d(v_d \mid \mathbf{s}_i, a) = P_d(v_d \mid \mathbf{s}_j, a)$.*

Using these definitions, it is possible to define partitions of S that preserve the multi-time model of an option o, which we prove in the following two theorems:

Theorem 9 *Let o be a proper option and let Λ_d be a partition of S that has the stochastic substitution property, is policy respecting, termination respecting, and probability respecting of S_d. Then for each pair of states $(\mathbf{s}_i, \mathbf{s}_j) \in S^2$ and each value $v_d \in \mathcal{D}(S_d)$, $[\mathbf{s}_i]_{\Lambda_d} = [\mathbf{s}_j]_{\Lambda_d}$ implies that $P_d(v_d \mid \mathbf{s}_i, o) = P_d(v_d \mid \mathbf{s}_j, o)$.*

The proof of Theorem 9 appears in Appendix B. As a consequence of Theorem 9, it is possible to ignore some state variables while computing the discounted probability model of an exit option. Take the example of computing the discounted probability $P^o_{\mathtt{W}}$ associated with state variable $S_{\mathtt{W}}$ and exit option $\overline{C} \to C$ in the coffee task. The policy and termination condition function of $\overline{C} \to C$ only distinguish between values of state variable $S_{\mathtt{L}}$, that is, any partition of S that distinguishes between values of $S_{\mathtt{L}}$ is policy respecting and termination respecting. The value of $S_{\mathtt{W}}$ as a result of executing any action is determined by the previous values of $S_{\mathtt{U}}$, $S_{\mathtt{R}}$, and $S_{\mathtt{W}}$, that is, any partition of S that distinguishes between values of $S_{\mathtt{U}}$, $S_{\mathtt{R}}$, and $S_{\mathtt{W}}$ is probability respecting of $S_{\mathtt{W}}$. From Theorem 1 we know that a partition of S has the stochastic substitution property and is reward respecting if it distinguishes between values of all state variables that influence relevant state variables. It follows from Theorem 9 that a partition of S that distinguishes between values of $S_{\mathtt{L}}$, $S_{\mathtt{U}}$, $S_{\mathtt{R}}$, and $S_{\mathtt{W}}$ preserves the discounted probability $P^o_{\mathtt{W}}$, that is, state variables $S_{\mathtt{C}}$ and $S_{\mathtt{H}}$ are irrelevant for computing $P^o_{\mathtt{W}}$.

Theorem 10 *Let o be a proper option and let Λ_R be a partition of S that has the stochastic substitution property, is reward respecting, policy respecting, and termination respecting. Then for each pair of states $(\mathbf{s}_i, \mathbf{s}_j) \in S^2$, $[\mathbf{s}_i]_{\Lambda_R} = [\mathbf{s}_j]_{\Lambda_R}$ implies that $R(\mathbf{s}_i, o) = R(\mathbf{s}_j, o)$.*

The proof of Theorem 10 appears in Appendix C. Usually, all state variables indirectly influence reward, so normally it is not possible to ignore the values of any state variables while computing the expected reward model R^o associated with exit option o.

5.5 Distribution Irrelevance

Dietterich (2000b) defined a condition that he calls result distribution irrelevance: a subset of the state variables may be irrelevant for the resulting distribution of a temporally-extended activity. This condition only exists in the undiscounted case, that is, for $\gamma = 1$. Otherwise, the time it takes the activity to terminate influences subsequent reward. We can take advantage of distribution irrelevance to compute the multi-time model of an exit option when $\gamma = 1$. Let o be the exit option associated with the exit $\langle \mathbf{c}, a \rangle$. Since o terminates in the context $\mathbf{c}$, we know the value of each state variable in the set $\mathbf{C} \subseteq \mathbf{S}$ immediately before action a is executed. In other words, the values of state variables in the set $\mathbf{C}$ prior to executing o are irrelevant for the resulting distribution of o.

Because of distribution irrelevance, we do not need to solve Equation 15 for state variables in the set $\mathbf{C}$. Instead, the conditional probabilities associated with state variable $S_d \in \mathbf{C}$ and option o are given by the conditional probabilities associated with S_d and the exit action a, restricted to states $s \in S$ such that $f_{\mathbf{C}}(s) = \mathbf{c}$. For example, as a result of executing the exit option associated with the exit $\langle (S_{\mathtt{L}} = L), \mathtt{BC} \rangle$ in the coffee task, the value of state variable $S_{\mathtt{L}}$ is L immediately before executing BC. Executing the exit action BC has no influence on the value of $S_{\mathtt{L}}$. As a result of executing the option that acquires coffee, the location of the robot is always the coffee shop, regardless of its previous location.

We can also simplify computation of conditional probabilities for state variables that are unaffected by actions that the policy selects. Let $\mathbf{U}^o \subseteq \mathbf{S}$ denote the subset of state variables whose values do not change as a result of executing any action selected by the policy π of exit option o. For the exit option o associated with exit $\langle (S_{\mathtt{L}} = L), \mathtt{BC} \rangle$ in the coffee task, $\mathbf{U}^o = \{S_{\mathtt{U}}, S_{\mathtt{R}}, S_{\mathtt{C}}, S_{\mathtt{H}}\}$, since the values of these state variables do not change as a result of executing GO, the only action selected by the policy of o. Thus, the conditional probabilities associated with state variables in the set $\mathbf{U}^o$ can be computed without solving Equation 15.

5.6 Summary of the Algorithm

In summary, to compute the multi-time model of an exit option one should first solve Equation 9 to compute the expected reward. In addition, for each state variable, one should solve Equation 15 to compute the discounted probability model associated with that state variable. If $\gamma = 1$ and it is possible to take advantage of distribution irrelevance, it is not necessary to solve Equation 15 for that state variable. The computation is most efficient if matrices are represented as trees or ADDs; in that case, the resulting models are also trees.

When we have computed the conditional probability tree associated with each state variable for an exit option, as well as a tree representing expected reward, we can construct a DBN for the option in the same way that we can for primitive actions. Figure 11 shows the DBN for the exit option associated with the exit $\langle (S_{\mathtt{L}} = L), \mathtt{BC} \rangle$ in the coffee task when $\gamma = 1$, taking advantage of distribution irrelevance. Note that there is no edge to state variable $S_{\mathtt{L}}$, which indicates that the resulting location does not depend on any of the state variables.

Since the DBN model of an option is in the same form as the DBN models of primitive actions, we can treat the option as a single unit and apply any of the algorithms that take advantage of compact representations. In addition, the DBN model makes it possible to apply our technique to nested options, that is, options selecting between other options. Once the policy of an option has been learned, we can construct its DBN model and use that model both to learn the policy of a higher-level option and later to construct a DBN model of the higher-level option.

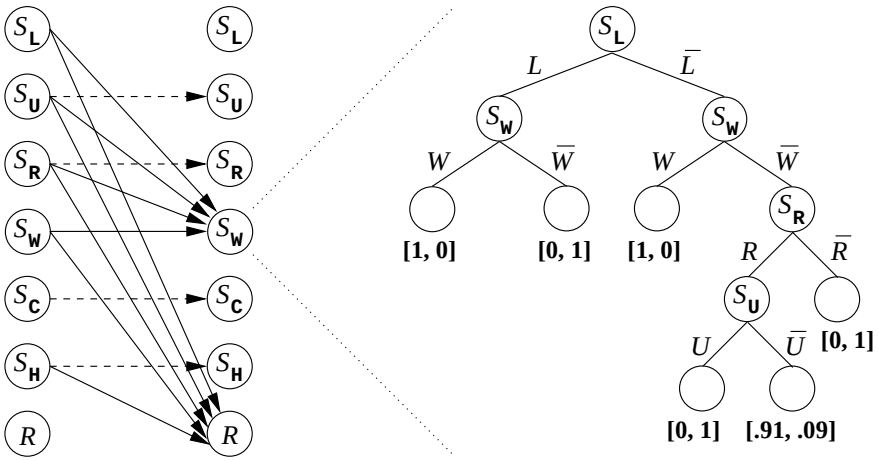


Figure 11: DBN for the option associated with $\langle (S_L = L), \texttt{BC} \rangle$

TASK	SPUDD	VISA	VISA-D
Coffee	75 ± 3	100 ± 33	18 ± 14
Taxi	$4{,}965 \pm 20$	$2{,}220{,}102 \pm 1{,}861$	$7{,}465 \pm 139$

Table 2: Comparison of time (ms) to convergence in two tasks

5.7 Experimental Results

We conducted a set of experiments to test the complexity of computing multi-time models for exit options. First, we ran VISA on the coffee task and used Equations 8 and 9 to compute the multi-time model of each exit option introduced. For each exit option, including the task option at the top level, we used SPUDD to compute an optimal policy. In a second experiment, we ran VISA again, but this time used Equation 15 to compute a separate transition probability model for each state variable. We set $\gamma = 1$ and used distribution irrelevance whenever possible to simplify computation of the transition probability model. We repeated these experiments in the Taxi task. For comparison, we also computed the time it takes SPUDD to converge in these two tasks.

Table 2 presents the results of the experiments, averaged over 100 trials. VISA-D denotes the VISA algorithm with the decomposed transition probability model. In all cases, the algorithms converged to an optimal policy for the task. Since the coffee task is very small, all algorithms converged relatively quickly. However, note that in the Taxi task, the convergence time of VISA is orders of magnitudes larger than that of SPUDD and VISA-D, while the convergence time of VISA-D is almost on par with that of SPUDD, even though it includes the time it took to compute the compact option models. Evidently, distribution irrelevance and simplified computation of the transition probability models can have a huge impact on the complexity of computing multi-time models for exit options.

6. Related Work

There exist several algorithms that decompose tasks into a hierarchy of activities. We have already mentioned the HEX-Q algorithm (Hengst, 2002) and its relation to our work. Dean and Lin (1995) used a fixed partition of the state space to decompose a factored MDP into regions. The authors developed an algorithm for solving the decomposed task by constructing activities for moving between regions. At the top level, the algorithm forms an abstract MDP with the regions as states and the activities as actions to approximate a global solution. Hauskrecht et al. (1998) extended this idea by suggesting several ways of constructing the set of activities given the decomposition. Most of their techniques rely on partial knowledge of the value function at different states to decide which activities to introduce. These techniques rely on prior knowledge of a useful partition, while our algorithm relies on the DBN model to decompose a task.

Nested Q-learning (Digney, 1996) introduces an activity for each value of each state variable. The goal of each activity is to reach the context described by the single state variable value. McGovern and Barto (2001) use diverse density to locate bottlenecks in successful solution paths, and introduce activities that reach these bottlenecks. Şimşek and Barto (2004) measure the relative novelty of each visited state, and introduce activities that reach states whose relative novelty exceeds a threshold value. Recent work on intrinsic motivation (Singh et al., 2005) tracks salient changes in variable values and introduces activities that cause salient changes to occur.

Other researchers use graph-theoretic approaches to decompose tasks. Menache et al. (2002) construct a state transition graph and introduce activities that reach states on the border of strongly connected regions of the graph. The authors use a max-flow/min-cut algorithm to identify border states in the transition graph. Mannor et al. (2004) use a clustering algorithm to partition the state space into different regions and introduce activities for moving between regions. Şimşek et al. (2005) identify subgoals by partitioning local state transition graphs that represent only the most recently recorded experience.

Another approach is to track learning in several related tasks and identify activities that are useful across tasks. SKILLS (Thrun and Schwartz, 1996) identifies activities that minimize a function of the performance loss induced by the resulting hierarchy and the total description length of all actions. PolicyBlocks (Pickett and Barto, 2002) identifies regions in the state space for which the policy is identical across tasks, and introduces activities that represent the policy of each region. Each activity is only admissible within its region of the state space.

Helmert (2004) developed an algorithm that constructs a causal graph similar to that of VISA and uses the graph to decompose deterministic planning tasks. The algorithm assumes a STRIPS formulation of actions (Fikes and Nilsson, 1971), which is similar to the DBN model of factored MDPs. Just like the DBN model, the STRIPS formulation expresses actions in terms of causes and effects on the state variables, except that the causes and effects are deterministic. Helmert (2004) uses the STRIPS action formulation to construct a causal graph in a special class of deterministic tasks in which the causal graph has one absorbing state variable with edges from each of the other state variables. The author shows that his algorithm efficiently solves a set of standard planning tasks using activities to represent the stand-alone tasks of the resulting decomposition.

There are several efficient algorithms for solving factored MDPs that use the DBN model to compactly describe transition probabilities and expected reward. Structured Policy Iteration, or SPI (Boutilier et al., 1995), stores the policy and value function in the form of trees. The algorithm performs policy iteration by intermittently updating the policy and value function, possibly changing

the structure of the trees in the process. Hoey et al. (1999) modified SPI to include algebraic decision diagrams, or ADDs, which store conditional probabilities more compactly than trees. Symbolic Real-Time Dynamic Programming, or sRTDP (Feng et al., 2003), also assumes that the conditional probabilities of the DBN model are stored using ADDs. sRTDP is an extension of Real-Time Dynamic Programming, or RTDP (Barto et al., 1995), that clusters states into abstract states based on two criteria, and performs an efficient backup of the value of the current abstract state following each execution of an action in the environment.

The DBN-E^3 algorithm (Kearns and Koller, 1999) is based on the assumption that there exists an approximate planning algorithm for the task, and that the structure of the DBN model is given. Using the planning procedure as a subroutine, the algorithm explores the state space and fills in the parameters of the DBN model. The running time of the algorithm is polynomial in the number of parameters of the DBN model, generally much smaller than the number of states. Guestrin et al. (2001) developed an algorithm based on linear programming that combines the DBN model with max-norm projections to solve factored MDPs. The algorithm is based on the assumption that there is a set of basis functions for representing the value function. It is guaranteed to converge to an approximately optimal solution.

Sutton et al. (1999) developed the multi-time model of options that we used to represent the effect of activities. The multi-time model includes an estimate of the transition probabilities and expected reward of options. Using the multi-time model of an option, it is possible to treat the option as a single unit during learning and planning. SMDP value learning (Sutton et al., 1999) uses the multi-time model to learn values or action-values in an SMDP. SMDP planning (Sutton et al., 1999) uses the multi-time model to perform planning in an SMDP, similar to policy iteration.

7. Conclusion

We presented Variable Influence Structure Analysis, or VISA, an algorithm that decomposes factored MDPs into hierarchies of options. VISA uses a DBN model of the factored MDP to construct a causal graph describing how state variables are related. The algorithm then searches in the conditional probability trees of the DBN model for exits, that is, combinations of state variable values and actions that cause the values of other state variables to change. VISA introduces an option for each exit and uses sophisticated techniques to construct the components of each option. The result is a hierarchy of options in which the policy of an option selects among options at a lower level in the hierarchy. Experimental results in a series of tasks show that the performance of VISA is comparable to that of state-of-the-art algorithms that exploit the DBN model, and in one task (AGV) VISA significantly outperforms the other algorithms.

VISA is based on the assumption that the values of key state variables change relatively infrequently. This is the same assumption made by Hengst (2002), Helmert (2004), and Singh et al. (2005). Just like the HEX-Q algorithm (Hengst, 2002), VISA decomposes a task into activities by detecting the combinations of state variable values and actions that cause key variable value changes. However, as we already discussed, VISA uses the causal graph to represent how state variables are related, which is a more realistic model than that used by HEX-Q. Unlike the work of Helmert (2004), VISA can handle any configuration of the causal graph.

Many existing algorithms need to accumulate extensive experience in the environment to decompose a task into activities, and they usually store quantities for each state. Assuming that the DBN model is given, VISA does not need to accumulate experience in the environment to perform

the decomposition. In addition, VISA only stores quantities proportional to the size of the conditional probability trees of the DBN model. Although we do not provide any comparisons, it is likely that VISA uses less memory and performs decomposition of a task in less time than these other algorithms.

Our second algorithm is a method for computing compact models of the options discovered by VISA. Existing methods for computing compact option models do not scale well to large tasks. For this reason, the first implementation of VISA uses reinforcement learning to approximate an optimal policy of each option. If VISA had access to compact option models, it could use dynamic programming techniques to compute the option policies without interacting with the environment. Our algorithm constructs partitions with certain properties to reduce the complexity of computing compact option models. The algorithm computes a DBN model for each option identical to the DBN model for primitive actions. This makes it possible to apply existing algorithms that use the DBN model to efficiently approximate option policies.

For VISA to successfully decompose a task, the causal graph needs to contain at least two separate strongly connected components. In tasks for which each state variable indirectly influences each other state variable, decomposition using this strategy is not possible. In other words, VISA is limited to function well in tasks with relatively sparse relationships between state variables. We believe that a non-trivial number of realistic tasks fall within this category. For example, in most navigation tasks, location influences the value of variables representing stationary objects, which in turn have no impact on location. Moreover, constructing the causal graph is polynomial in the size of the DBNs, so it is relatively inexpensive to test whether or not VISA can successfully decompose a task.

7.1 Future Work

Hoey et al. (1999) pioneered the use of algebraic decision diagrams, or ADDs, to store the conditional probability distributions of the DBN model. Since ADDs are a more compact representation than trees, they require less memory. More importantly, several operations can be executed faster on ADDs than on trees. Although VISA uses trees to represent the conditional probability distributions, it would be relatively straightforward to change the representation to ADDs. Possibly, this modification could speed up decomposition and construction of compact option models.

It is also possible to combine VISA with other techniques that facilitate scaling. For example, once VISA has decomposed a task into options, we can apply reinforcement learning with function approximation to learn the option policies. Another possibility is to use existing algorithms to detect bottlenecks in the transition graph of a strongly connected component in the causal graph. This would enable further decomposition of the option SMDPs into even smaller subtasks.

Recall that VISA performs state abstraction for an option SMDP by constructing the partition $\Lambda_{\mathbf{Z}}$, where $\mathbf{Z} \subseteq \mathbf{S}$ is the set of state variables in strongly connected components whose variable values appear in the context of the associated exit. As a result of state abstraction, the option policy may be suboptimal. The problem occurs when an option selected by the option policy changes the value of a state variable not in $\mathbf{Z}$ that indirectly influences state variables in $\mathbf{Z}$. This problem would be alleviated if we merge strongly connected components whose state variables are affected by the same actions. The resulting decomposition would be less efficient in terms of learning complexity but would guarantee recursive optimality.

Our formal analysis of constructing compact option models requires partitions with a set of established properties. The requirement that the partitions should have all of these properties is quite strong. A possible line of future research is to relax or approximate the required properties of partitions, which could lead to even more efficient computation of option models, albeit with some loss of accuracy. An analysis of the resulting approximation could help determine a tradeoff between the complexity of computing compact option models and the accuracy of the resulting model.

We also made a strong independence assumption in order to reduce the complexity of computing a compact option model. Our algorithm assumes that the value of a state variable that results from executing an option is independent of the resulting values of other state variables. Since an option takes variable time to execute, the option passes through many states during execution. The independence assumption only holds if the resulting values of state variables are independent regardless of which state the option is currently in. In many cases, our independence assumption induces an approximation error. If possible, we would like to establish bounds on this approximation error to analyze the accuracy of our algorithm.

It is unrealistic to assume that a DBN model is always given prior to learning. If no DBN model is available, it is necessary to learn a DBN model from experience prior to executing VISA. There exist algorithms for active learning of Bayesian networks that can be applied to factored MDPs (Murphy, 2001; Steck and Jaakkola, 2002; Tong and Koller, 2001). However, these algorithms assume that it is possible to sample the MDP at arbitrary states. If we assume that it is only possible to sample the MDP along trajectories, it becomes necessary to develop novel algorithms for active learning of DBN models. Such algorithms would select actions with the goal of learning a DBN model describing the effect of actions on the state variables as quickly as possible.

Acknowledgments

The authors would like to thank Alicia "Pippin" Wolfe and Mohammad Ghavamzadeh for useful comments on this paper. This work was partially funded by NSF grants ECS-0218125 and CCF-0432143.

Appendix A. Proof of Theorem 4

In this appendix we prove Theorem 4 from Section 5. Equations 10 and 11 are consistent and have unique solutions if and only if the matrix $M = E - \gamma(E - B)\sum_{a\in A}\Pi^a P^a$ is invertible, that is, if $\det(M) \neq 0$. Each element of P^a is in the range $[0,1]$, and each row of P^a sums to 1. Because of the properties of π, it follows that $\sum_{a\in A}\Pi^a = E$ and that $\sum_{a\in A}\Pi^a P^a$ has the same properties as P^a. $(E - B)$ is a diagonal matrix whose elements are in the range $[0,1]$. Then $\gamma(E - B)\sum_{a\in A}\Pi^a P^a$ is a matrix such that each element is in the range $[0,1]$ and such that the sum of each row is in the range $[0,1]$. In other words, M has the following properties, where $n = |S|$:

1. for each $i = 1,\ldots,n$: $0 \leq m_{ii} \leq 1$,
2. for each $i = 1,\ldots,n$, $j \neq i$: $-m_{ii} \leq m_{ij} \leq 0$,
3. for each $i = 1,\ldots,n$: $0 \leq \sum_{j=1}^{n} m_{ij} \leq m_{ii}$.

Lemma 11 *An element m_{ii} on the diagonal of M equals 0 if and only if*

1. $\gamma = 1$,
2. $\beta(s_i) = 0$,
3. *for each action $a \in A$ such that $\pi(s_i, a) > 0$, $P(s_i \mid s_i, a) = 1$.*

Proof $m_{ii} = 1 - \gamma(1 - \beta(s_i)) \sum_{a \in A} \pi(s_i, a) P(s_i \mid s_i, a)$. The only solution to $m_{ii} = 0$ is $\gamma = 1$, $\beta(s_i) = 0$, and $P(s_i \mid s_i, a) = 1$ for each action $a \in A$ such that $\pi(s_i, a) > 0$. ■

An option is proper if and only if there is no set of absorbing states S' such that $\beta(s) = 0$ for each state $s \in S'$. A set of states S' is absorbing if and only if the probability of transitioning from any state in S' to any state outside S' is 0. A special case occurs when S' contains a single state s_i such that $\beta(s_i) = 0$ and such that $P(s_i \mid s_i, a)$ for each action $a \in A$ such that $\pi(s_i, a) > 0$. From Lemma 11 it follows that an element m_{ii} on the diagonal of M equals 0 if and only if s_i is an absorbing state such that $\beta(s_i) = 0$. Since no such state exists for a proper option o, we conclude that all elements on the diagonal of M are larger than 0 for a proper option o. Then it is possible to multiply each row of M by its diagonal element $1/m_{ii}$ to obtain a matrix A with the following properties:

1. for each $i = 1, \ldots, n$: $a_{ii} = 1$,
2. for each $i = 1, \ldots, n$, $j \neq i$: $-1 \leq a_{ij} \leq 0$,
3. for each $i = 1, \ldots, n$: $0 \leq \sum_{j=1}^{n} a_{ij} \leq 1$.

Since matrix A is obtained by multiplying each row of M by a scalar, the determinant of M equals 0 if and only if the determinant of A equals 0. We can write A as

$$A = \begin{pmatrix} 1 & a_{12} & \cdots & a_{1n} \\ a_{21} & 1 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 1 \end{pmatrix} = \begin{pmatrix} - & \mathbf{r}_1 & - \\ - & \mathbf{r}_2 & - \\ & \vdots & \\ - & \mathbf{r}_n & - \end{pmatrix},$$

where $\mathbf{r}_i$ is the ith row of A. It is possible to eliminate an element a_{ij}, $j < i$, by subtracting $a_{ij}\mathbf{r}_j$ from row $\mathbf{r}_i$:

$$\mathbf{r}_i - a_{ij}\mathbf{r}_j = \begin{pmatrix} a_{i1} - a_{ij}a_{j1} & \cdots & a_{ij} - a_{ij} \cdot 1 & \cdots & 1 - a_{ij}a_{ji} & \cdots & a_{in} - a_{ij}a_{jn} \end{pmatrix}.$$

Lemma 12 $0 \leq 1 - a_{ij}a_{ji} \leq 1$, *and* $1 - a_{ij}a_{ji} = 0$ *if and only if* $a_{ij} = a_{ji} = -1$.

Proof Follows immediately from the properties of A. ■

Lemma 13 *If* $1 - a_{ij}a_{ji} > 0$, *elimination of* a_{ij} *preserves the properties of* A.

Proof Since $1 - a_{ij}a_{ji} > 0$, we can multiply $\mathbf{r}_i - a_{ij}\mathbf{r}_j$ by $1/(1 - a_{ij}a_{ji})$:

$$\bar{\mathbf{r}}_i = \frac{1}{1 - a_{ij}a_{ji}} [\mathbf{r}_i - a_{ij}\mathbf{r}_j] = \begin{pmatrix} \frac{a_{i1} - a_{ij}a_{j1}}{1 - a_{ij}a_{ji}} & \cdots & 0 & \cdots & 1 & \cdots & \frac{a_{in} - a_{ij}a_{jn}}{1 - a_{ij}a_{ji}} \end{pmatrix}.$$

It follows immediately that element i of row $\bar{\mathbf{r}}_i$ equals $(1-a_{ij}a_{ji})/(1-a_{ij}a_{ji}) = 1$ and that element j equals $(a_{ij}-a_{ij}\cdot 1)/(1-a_{ij}a_{ji}) = 0$. For each $k = 1,\ldots,n$, $k \neq i, j$, compute bounds on element k of $\bar{\mathbf{r}}_i$:

$$\begin{aligned}
\frac{a_{ik}-a_{ij}a_{jk}}{1-a_{ij}a_{ji}} &\leq \frac{a_{ik}-0}{1-a_{ij}a_{ji}} \leq \frac{0-0}{1-a_{ij}a_{ji}} = 0, \\
\frac{a_{ik}-a_{ij}a_{jk}}{1-a_{ij}a_{ji}} &= \frac{1-a_{ij}a_{ji}+a_{ik}-a_{ij}a_{jk}-(1-a_{ij}a_{ji})}{1-a_{ij}a_{ji}} = \\
&= \frac{1+a_{ik}-a_{ij}(a_{ji}+a_{jk})}{1-a_{ij}a_{ji}} - 1 \geq \\
&\geq \frac{1+a_{ik}+a_{ij}}{1-a_{ij}a_{ji}} - 1 \geq \frac{1-1}{1-a_{ij}a_{ji}} - 1 = -1.
\end{aligned}$$

Also compute bounds on the sum of the elements of $\bar{\mathbf{r}}_i$:

$$\begin{aligned}
\sum_{k=1}^{n} \frac{a_{ik}-a_{ij}a_{jk}}{1-a_{ij}a_{ji}} &= \sum_{k=1}^{j-1} \frac{a_{ik}-a_{ij}a_{jk}}{1-a_{ij}a_{ji}} + \frac{a_{ij}-a_{ij}\cdot 1}{1-a_{ij}a_{ji}} + \sum_{k=j+1}^{i-1} \frac{a_{ik}-a_{ij}a_{jk}}{1-a_{ij}a_{ji}} + \\
&+ \frac{1-a_{ij}a_{ji}}{1-a_{ij}a_{ji}} + \sum_{k=i+1}^{n} \frac{a_{ik}-a_{ij}a_{jk}}{1-a_{ij}a_{ji}} \leq \\
&\leq 0+0+0+1+0 = 1, \\
\sum_{k=1}^{n} \frac{a_{ik}-a_{ij}a_{jk}}{1-a_{ij}a_{ji}} &= \frac{1}{1-a_{ij}a_{ji}} \left[\sum_{k=1}^{n} a_{ik} - a_{ij} \sum_{k=1}^{n} a_{jk} \right] \geq \frac{0+0}{1-a_{ij}a_{ji}} = 0.
\end{aligned}$$

It follows that row $\bar{\mathbf{r}}_i$ satisfies the properties of A. ■

From Lemma 12 and Lemma 13 it follows that the properties of A are preserved under elimination unless the element on the diagonal equals 0. We can compute the determinant of A by repeatedly performing elimination until A is an upper triangular matrix. If any element on the diagonal becomes 0 during elimination, $\det(A) = 0$. Otherwise, the determinant of A equals the inverse of the product of the coefficients by which we multiplied rows during elimination. Since each coefficient is larger than 0, it follows that $\det(A) > 0$.

Lemma 14 *Let $C = \{c_1,\ldots,c_m\}$ be a set of m indices, and let $S(C,\mathbf{r}_i) = \sum_{k=1}^{m} a_{ic_k}$ be the sum of elements of row $\mathbf{r}_i$ whose column indices are elements of C. Assume that $i \in C$ and that $S(C,\bar{\mathbf{r}}_i) = 0$ after elimination of an element a_{ij}, $j < i$. Then $S(C\cup\{j\},\mathbf{r}_i) = 0$ and $S(C\cup\{j\},\mathbf{r}_j) = 0$ prior to elimination of a_{ij}.*

Proof When we eliminate an element a_{ij}, $j < i$, the sum of elements of row $\bar{\mathbf{r}}_i$ whose column indices are elements of C is

$$\begin{aligned}
S(C,\bar{\mathbf{r}}_i) &= S(C,\bar{\mathbf{r}}_i) + 0 = \\
&= \sum_{k=1}^{m} \frac{a_{ic_k}-a_{ij}a_{jc_k}}{1-a_{ij}a_{ji}} + \frac{a_{ij}-a_{ij}\cdot 1}{1-a_{ij}a_{ji}} = \\
&= \frac{1}{1-a_{ij}a_{ji}} \left[\left(\sum_{k=1}^{m} a_{ic_k} + a_{ij} \right) - a_{ij} \left(\sum_{k=1}^{m} a_{jc_k} + 1 \right) \right].
\end{aligned}$$

Since i is one of the indices in C, it follows from the properties of A that $S(C,\bar{\mathbf{r}}_i) = 0$ if and only if $\sum_{k=1}^{m} a_{ic_k} + a_{ij} = 0$ and either $a_{ij} = 0$ or $\sum_{k=1}^{m} a_{jc_k} + 1 = 0$. If $a_{ij} = 0$, there was no reason to perform elimination, so it follows that $S(C \cup \{j\}, \mathbf{r}_i) = \sum_{k=1}^{m} a_{ic_k} + a_{ij} = 0$ and that $S(C \cup \{j\}, \mathbf{r}_j) = \sum_{k=1}^{m} a_{jc_k} + 1 = 0$. ■

Lemma 15 *If $S(C,\mathbf{r}_k) = 0$ for each row $\mathbf{r}_k$, $k \in C$, after elimination of an element a_{ij}, $j \notin C$, in row $\mathbf{r}_i$, $i \in C$, it follows that $S(C \cup \{j\}, \mathbf{r}_k) = 0$ for each row $\mathbf{r}_k$, $k \in C \cup \{j\}$ prior to elimination of a_{ij}.*

Proof If $S(C,\mathbf{r}_i) = 0$ following elimination of a_{ij}, it follows from Lemma 14 that $S(C \cup \{j\}, \mathbf{r}_i) = 0$ and that $S(C \cup \{j\}, \mathbf{r}_j) = 0$ prior to elimination of a_{ij}. For $k \in C - \{i\}$, $S(C \cup \{j\}, \mathbf{r}_k) = S(C,\mathbf{r}_k) + a_{kj} = a_{kj}$. Since $a_{kj} \leq 0$ and $S(C \cup \{j\}, \mathbf{r}_k) \geq 0$, it follows that $S(C \cup \{j\}, \mathbf{r}_k) = a_{kj} = 0$. ■

Lemma 16 *If* $\det(A) = 0$, *it is possible to rearrange the rows and columns of A to obtain*

$$\begin{pmatrix} X & 0 \\ Y & Z \end{pmatrix},$$

where X is a $k \times k$ matrix such that for each $i = 1,\ldots,k$, $\sum_{j=1}^{k} x_{ij} = 0$.

Proof If $\det(A) = 0$, there exists $i, j < i$ such that a_{ii} becomes 0 during elimination of a_{ij}. From Lemma 12 it follows that $a_{ij} = a_{ji} = -1$ prior to elimination of a_{ij}, so $S(\{i,j\}, \mathbf{r}_i) = a_{ij} + a_{ii} = -1 + 1 = 0$ and $S(\{i,j\}, \mathbf{r}_j) = a_{jj} + a_{ji} = 1 - 1 = 0$. Let $C = \{i,j\}$. Recursively find each index l such that elimination of element a_{kl} occurred prior to this round in row $\mathbf{r}_k$, $k \in C$. Then it follows from Lemma 15 that $S(C \cup \{l\}, \mathbf{r}_k) = 0$ for each $k \in C \cup \{l\}$ prior to elimination of a_{kl}. Add each such index l to C. Prior to elimination of any element, it is possible to rearrange the rows and columns of A to obtain

$$\begin{pmatrix} a'_{11} & \cdots & a'_{1m} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a'_{m1} & \cdots & a'_{mm} & 0 & \cdots & 0 \\ a'_{(m+1)1} & \cdots & a'_{(m+1)m} & a'_{(m+1)(m+1)} & \cdots & a'_{(m+1)n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a'_{n1} & \cdots & a'_{nm} & a'_{n(m+1)} & \cdots & a'_{nn} \end{pmatrix},$$

where the first m rows and columns are those whose indices are elements of C. Since $S(C,\mathbf{r}_k) = 0$ for each row $\mathbf{r}_k$, $k \in C$, it follows that the sum of row $\mathbf{r}_k$ equals 0 and that for each $l \notin C$, element a_{kl} equals 0. ■

From the definition of M it follows that it is only possible to rearrange the rows and columns to obtain

$$\begin{pmatrix} X & 0 \\ Y & Z \end{pmatrix},$$

such that the sum of each row of X equals 0, if there is an absorbing set of states S' such that $\beta(s) = 0$ for each state $s \in S'$ and if $\gamma = 1$. For a proper option o, it is not possible to rearrange M that way. Since the sum of one row of M equals 0 if and only if the sum of the same row of A equals 0, it is not possible to rearrange A that way either. It follows from the contrapositive of Lemma 16 that $\det(A) \neq 0$, which also means that $\det(M) \neq 0$. This concludes the proof of Theorem 4.

Appendix B. Proof of Theorem 9

Assume that for each block λ and each value $v_d \in \mathcal{D}(S_d)$, the probability $P_d(v_d \mid \mathbf{s}_k, o)$ is identical for each state $\mathbf{s}_k \in \lambda$. Let P^o_{λ,v_d} denote that probability. We will show that $P_d(v_d \mid \mathbf{s}_i, o) = P_d(v_d \mid \mathbf{s}_j, o)$ checks under this assumption if $[\mathbf{s}_i]_{\Lambda_d} = [\mathbf{s}_j]_{\Lambda_d}$.

From Equation 14, the expression for $P_d(v_d \mid \mathbf{s}_i, o)$ is given by

$$\gamma \left[\beta(\mathbf{s}_i) P_d(v_d \mid \mathbf{s}_i, a) + (1 - \beta(\mathbf{s}_i)) \sum_{a' \in A} \pi(\mathbf{s}_i, a') \sum_{\mathbf{s}' \in S} P(\mathbf{s}' \mid \mathbf{s}_i, a') P_d(v_d \mid \mathbf{s}', o) \right].$$

We can expand the sum $\sum_{\mathbf{s}' \in S}$ by first summing over blocks λ of partition Λ_d and then over states $\mathbf{s}_k$ in block λ, replacing $P_d(v_d \mid \mathbf{s}_k, o)$ with P^o_{λ,v_d}:

$$\gamma \left[\beta(\mathbf{s}_i) P_d(v_d \mid \mathbf{s}_i, a) + (1 - \beta(\mathbf{s}_i)) \sum_{a' \in A} \pi(\mathbf{s}_i, a') \sum_{\lambda \in \Lambda_d} \sum_{\mathbf{s}_k \in \lambda} P(\mathbf{s}_k \mid \mathbf{s}_i, a') P^o_{\lambda,v_d} \right].$$

Since P^o_{λ,v_d} does not depend on $\mathbf{s}_k$, we can move it outside the summation to obtain

$$\gamma \left[\beta(\mathbf{s}_i) P_d(v_d \mid \mathbf{s}_i, a) + (1 - \beta(\mathbf{s}_i)) \sum_{a' \in A} \pi(\mathbf{s}_i, a') \sum_{\lambda \in \Lambda_d} P^o_{\lambda,v_d} \sum_{\mathbf{s}_k \in \lambda} P(\mathbf{s}_k \mid \mathbf{s}_i, a') \right].$$

We can expand the expression for $P_d(v_d \mid \mathbf{s}_j, o)$ in the same way to obtain

$$\gamma \left[\beta(\mathbf{s}_j) P_d(v_d \mid \mathbf{s}_j, a) + (1 - \beta(\mathbf{s}_j)) \sum_{a' \in A} \pi(\mathbf{s}_j, a') \sum_{\lambda \in \Lambda_d} P^o_{\lambda,v_d} \sum_{\mathbf{s}_k \in \lambda} P(\mathbf{s}_k \mid \mathbf{s}_j, a') \right].$$

If $[\mathbf{s}_i]_{\Lambda_d} = [\mathbf{s}_j]_{\Lambda_d}$, it follows immediately from the definitions of stochastic substitution property, policy respecting, termination respecting, and probability respecting of S_d that $P_d(v_d \mid \mathbf{s}_i, o) = P_d(v_d \mid \mathbf{s}_j, o)$. Lemma 5 states that the solution to the equations in Equation 15 is unique. Since we know that $P_d(v_d \mid \mathbf{s}_i, o) = P_d(v_d \mid s_j, o)$ is a solution, it follows from Lemma 5 that it is the only solution. This concludes the proof.

Appendix C. Proof of Theorem 10

Assume that for each block $\lambda \in \Lambda_R$ and each state $\mathbf{s}_k \in \lambda$, the expected reward $R(\mathbf{s}_k, o)$ as a result of executing option o is equal, and let R^o_λ denote that expected reward. We will show that $R(\mathbf{s}_i, o) = R(\mathbf{s}_j, o)$ checks under this assumption if $[\mathbf{s}_i]_{\Lambda_R} = [\mathbf{s}_j]_{\Lambda_R}$.

From Equation 9, the expression for $R(\mathbf{s}_i, o)$ is given by

$$R(\mathbf{s}_i, o) = \beta(\mathbf{s}_i) R(\mathbf{s}_i, a) + (1 - \beta(\mathbf{s}_i)) \sum_{a' \in A} \pi(\mathbf{s}_i, a') \left[R(\mathbf{s}_i, a') + \sum_{\mathbf{s}' \in S} P(\mathbf{s}' \mid \mathbf{s}_i, a') R(\mathbf{s}', o) \right].$$

We can expand the sum $\sum_{s' \in S}$ by first summing over blocks λ of partition Λ_R and then over states $\mathbf{s}_k$ in block λ, replacing $R(\mathbf{s}_k, o)$ with R^o_λ:

$$R(\mathbf{s}_i, o) = \beta(\mathbf{s}_i) R(\mathbf{s}_i, a) + (1 - \beta(\mathbf{s}_i)) \sum_{a' \in A} \pi(\mathbf{s}_i, a') \left[R(\mathbf{s}_i, a') + \sum_{\lambda \in \Lambda_R} \sum_{\mathbf{s}_k \in \lambda} P(\mathbf{s}_k \mid \mathbf{s}_i, a') R^o_\lambda \right].$$

Since R^o_λ does not depend on $\mathbf{s}_k$, we move it outside the summation to obtain

$$R(\mathbf{s}_i, o) = \beta(\mathbf{s}_i) R(\mathbf{s}_i, a) + (1 - \beta(\mathbf{s}_i)) \sum_{a' \in A} \pi(\mathbf{s}_i, a') \left[R(\mathbf{s}_i, a') + \sum_{\lambda \in \Lambda_R} R^o_\lambda \sum_{\mathbf{s}_k \in \lambda} P(\mathbf{s}_k \mid \mathbf{s}_i, a') \right].$$

We expand the expression for $R(\mathbf{s}_j, o)$ in the same way to obtain

$$R(\mathbf{s}_j, o) = \beta(\mathbf{s}_j) R(\mathbf{s}_j, a) + (1 - \beta(\mathbf{s}_j)) \sum_{a' \in A} \pi(\mathbf{s}_j, a') \left[R(\mathbf{s}_j, a') + \sum_{\lambda \in \Lambda_R} R^o_\lambda \sum_{\mathbf{s}_k \in \lambda} P(\mathbf{s}_k \mid \mathbf{s}_j, a') \right].$$

If $[\mathbf{s}_i]_{\Lambda_d} = [\mathbf{s}_j]_{\Lambda_d}$, it follows immediately from the definitions of stochastic substitution property, reward respecting, policy respecting, and termination respecting that $R(\mathbf{s}_i, o) = R(\mathbf{s}_j, o)$. Theorem 4 states that the solution to the equations in Equation 9 is unique. Since we know that $R(\mathbf{s}_i, o) = R(\mathbf{s}_j, o)$ is a solution, it follows from Theorem 4 that it is the only solution. This concludes the proof.

References

A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence, Special Volume on Computational Research on Interaction and Agency*, 72(1):81–138, 1995.

R. Bellman. A Markov decision process. *Journal of Mathematical Mechanics*, 6:679–684, 1957.

C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. *Proceedings of the International Joint Conference on Artificial Intelligence*, 14:1104–1113, 1995.

S. Bradtke and M. Duff. Reinforcement learning methods for continuous-time Markov decision problems. *Advances in Neural Information Processing Systems*, 7:393–400, 1995.

Ö. Şimşek and A. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. *Proceedings of the International Conference on Machine Learning*, 21:751–758, 2004.

Ö. Şimşek, A. Wolfe, and A. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. *Proceedings of the International Conference on Machine Learning*, 22, 2005.

T. Dean and R. Givan. Model minimization in Markov decision processes. *Proceedings of the National Conference on Artificial Intelligence*, 14:106–111, 1997.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

T. Dean and S. Lin. Decomposition techniques for planning in stochastic domains. *Proceedings of the International Joint Conference on Artificial Intelligence*, 14:1121–1129, 1995.

T. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000a.

T. Dietterich. State Abstraction in MAXQ Hierarchical Reinforcement Learning. *Advances in Neural Information Processing Systems*, 12:994–1000, 2000b.

B. Digney. Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments. *From animals to animats*, 4:363–372, 1996.

Z. Feng and E. Hansen. Symbolic Heuristic Search for Factored Markov Decision Processes. *Proceedings of the National Conference on Artificial Intelligence*, 18:455–460, 2002.

Z. Feng, E. Hansen, and S. Zilberstein. Symbolic generalization for on-line planning. *Proceedings of Uncertainty in Artificial Intelligence*, 19:209–216, 2003.

R. Fikes and N. Nilsson. Strips: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:189–208, 1971.

M. Ghavamzadeh and S. Mahadevan. Continuous-Time Hierarchical Reinforcement Learning. *Proceedings of the International Conference on Machine Learning*, 18:186–193, 2001.

C. Guestrin, D. Koller, and R. Parr. Max-norm Projections for Factored MDPs. *International Joint Conference on Artificial Intelligence*, 17:673–680, 2001.

D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

M. Hauskrecht, N. Meuleau, L. Kaelbling, T. Dean, and C. Boutilier. Hierarchical Solution of Markov Decision Processes using Macro-actions. *Uncertainty in Artificial Intelligence*, 14:220–229, 1998.

M. Helmert. A planning heuristic based on causal graph analysis. *Proceedings of the International Conference on Automated Planning and Scheduling*, 14:161–170, 2004.

B. Hengst. Discovering Hierarchy in Reinforcement Learning with HEXQ. *Proceedings of the International Conference on Machine Learning*, 19:243–250, 2002.

J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. Spudd: Stochastic Planning using Decision Diagrams. *Proceedings of Uncertainty in Artificial Intelligence*, 15:279–288, 1999.

A. Jonsson and A. Barto. Automated State Abstractions for Options Using the U-Tree Algorithm. *Advances in Neural Information Processing Systems*, 13:1054–1060, 2001.

A. Jonsson and A. Barto. A Causal Approach to Hierarchical Decomposition of Factored MDPs. *Proceedings of the International Conference on Machine Learning*, 22:401–408, 2005.

M. Kearns and D. Koller. Efficient Reinforcement Learning in Factored MDPs. *Proceedings of the International Joint Conference on Artificial Intelligence*, 16:740–747, 1999.

S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. *Proceedings of the International Conference on Machine Learning*, 21:560–567, 2004.

A. McGovern and A. Barto. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. *Proceedings of the International Conference on Machine Learning*, 18:361–368, 2001.

I. Menache, S. Mannor, and N. Shimkin. Q-Cut – Dynamic Discovery of Sub-Goals in Reinforcement Learning. *Proceedings of the European Conference on Machine Learning*, 13:295–306, 2002.

K. Murphy. *Active Learning of Causal Bayes Net Structure*. Technical Report, University of California, Berkeley, USA, 2001.

R. Parr. *Hierarchical Control and Learning for Markov Decision Processes*. Ph.D. Thesis, University of California at Berkeley, 1998.

R. Parr and S. Russell. Reinforcement Learning with Hierarchies of Machines. *Advances in Neural Information Processing Systems*, 10:1043–1049, 1998.

M. Pickett and A. Barto. Policyblocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning. *Proceedings of the International Conference on Machine Learning*, 19: 506–513, 2002.

M. Puterman. *Markov Decision Processes*. Wiley Interscience, New York, USA, 1994.

B. Ravindran. *An Algebraic Approach to Abstraction in Reinforcement Learning*. Ph.D. Thesis, Department of Computer Science, University of Massachusetts, Amherst, USA, 2004.

S. Singh, A. Barto, and N. Chentanez. Intrinsically Motivated Reinforcement Learning. *Advances in Neural Information Processing Systems*, 18:1281–1288, 2005.

H. Steck and T. Jaakkola. Unsupervised Active Learning in Large Domains. *Proceedings of Uncertainty in Artificial Intelligence*, 18:469–476, 2002.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, USA, 1998.

R. Sutton, D. Precup, and S. Singh. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

S. Thrun and A. Schwartz. Finding structure in reinforcement learning. *Advances in Neural Information Processing Systems*, 8:385–392, 1996.

S. Tong and D. Koller. Active learning for parameter estimation in Bayesian networks. *Advances in Neural Information Processing Systems*, 13:647–653, 2001.

Journal of Machine Learning Research 7 (2006) 2303-2328 Submitted 7/05; Revised 7/06; Published 11/06

Accurate Error Bounds for the Eigenvalues of the Kernel Matrix

Mikio L. Braun MIKIO.BRAUN@FIRST.FRAUNHOFER.DE
Fraunhofer FIRST.IDA
Kekuléstr. 7
12489 Berlin, Germany

Editor: John Shawe-Taylor

Abstract

The eigenvalues of the kernel matrix play an important role in a number of kernel methods, in particular, in kernel principal component analysis. It is well known that the eigenvalues of the kernel matrix converge as the number of samples tends to infinity. We derive probabilistic finite sample size bounds on the approximation error of individual eigenvalues which have the important property that the bounds scale with the eigenvalue under consideration, reflecting the actual behavior of the approximation errors as predicted by asymptotic results and observed in numerical simulations. Such scaling bounds have so far only been known for tail sums of eigenvalues. Asymptotically, the bounds presented here have a slower than stochastic rate, but the number of sample points necessary to make this disadvantage noticeable is often unrealistically large. Therefore, under practical conditions, and for all but the largest few eigenvalues, the bounds presented here form a significant improvement over existing non-scaling bounds.

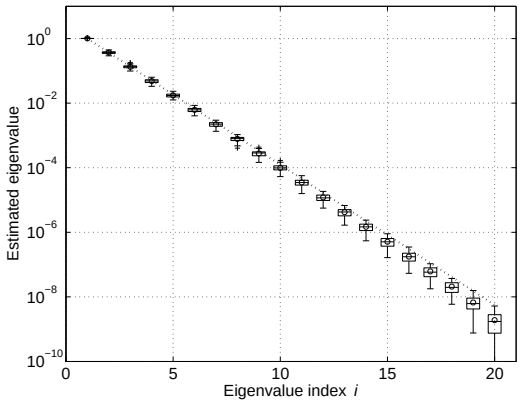
Keywords: kernel matrix, eigenvalues, relative perturbation bounds

1. Introduction

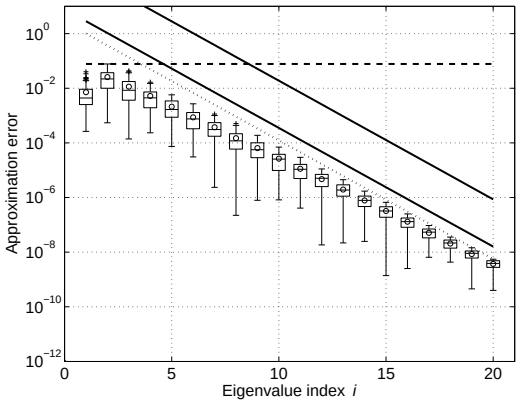
In the theoretical analysis of kernel principal component analysis (Schölkopf et al., 1998), the approximation error between the eigenvalues of the kernel matrix and their asymptotic counterparts plays a crucial role, as the eigenvalues compute the principal component variances in kernel feature space, and these are related to the reconstruction error of projecting to leading kernel principal component directions.

In order to obtain accurate bounds on the approximation error of eigenvalues, it has proven to be of prime importance to derive bounds which scale with the eigenvalue under consideration. The reason is that the approximation error scales with the eigenvalue such that the error is typically much smaller for small eigenvalues. Therefore, non-scaling bounds tend to overestimate the error for small eigenvalues as they are dominated by the largest occurring errors. Now, since smooth kernels usually display rapidly decaying eigenvalues, and such kernels are typically used in machine learning, obtaining accurate bounds in particular for small eigenvalues is highly relevant.

In an asymptotic setting, the effect that the approximation errors scale with the corresponding eigenvalues is well understood. In a paper by Koltchinskii and Giné (2000), a central limit theorem for the distribution of the approximation errors is derived. Considering only a single eigenvalue with multiplicity one, the asymptotic distribution of the properly scaled difference between approximate and true eigenvalue asymptotically approaches a normal distribution with mean zero and variance $\lambda_i^2 \operatorname{Var}_\mu(\psi_i^2)$. Thus, we would expect that the approximation error is of order $O(\lambda_i \operatorname{Std}_\mu(\psi_i^2) n^{-1/2})$,



(a) Approximate eigenvalues (box plots) and the true eigenvalues (dotted line). Note that although the box plots appear to become larger visually, due to the logarithmic scale the approximation error actually becomes small quickly.



(b) Approximation errors (box plots). For orientation, the true eigenvalues (dotted line) have also be included in the plot. The dashed line plots the smallest possible non-scaling bound on the approximation error. The solid lines plot two bounds derived in this paper, the smaller one requiring the knowledge of the true eigenfunctions.

Figure 1: Approximated eigenvalues for kernel matrices with rapidly decaying eigenvalues have an approximation error which scales with the true eigenvalue.

leading to a much smaller approximation error for small eigenvalues than for large eigenvalues (neglecting the variance of ψ_i^2 for the moment).

We are interested in deriving a probabilistic finite sample size bound to show that this effect not only occurs asymptotically, but can already be observed for small sample sizes. The following numerical example illustrates this effect: In Figure 1 we have plotted the approximate eigenvalues and the approximation errors for a kernel function with exponentially decaying eigenvalues constructed from Legendre polynomials (see Section 7.1 for details). The approximation errors scale with the true eigenvalue, and the smallest possible non-scaling bound (dashed line) overestimates the error severely for all but the first four eigenvalues. On the other hand, our bounds (solid lines) scale with the eigenvalues resulting in a bound which matches the true approximation error significantly better.

Such scaling bounds have recently been derived for tail sums of eigenvalues by Blanchard et al. (2006). There, the square root of the considered tail sum occurs in the bound, leading to bounds which correctly predict that the error for tail sums of small eigenvalues is smaller than that for tail sums starting with larger eigenvalues.

However, scaling bounds for the approximation error between individual eigenvalues, as are derived in this work, were not known so far. Note that these two settings are not interchangeable: although bounds on tail sums can be combined (more concretely, subtracted) to obtain bounds for single eigenvalues, the scaling still depends on tail sums, not single eigenvalues.

Note that the error bounds presented in this paper depend on the true eigenvalue. At first, this seems to be an undesirable feature, as this limits the practical applicability of these bounds. However, we have adopted a more theoretical approach in this work with the goal to understand

the underlying principles which permit the derivation of scaling bounds for individual eigenvalues first. In a second step, one could then use these results to construct statistical tests to estimate, for example, the overall decay rate of the eigenvalues based on these bounds. We will briefly discuss the question of constructing confidence bounds again in Section 9.

Overview

This paper is structured as follows: Section 2 contains the statements of the main results and explains the involved quantities. The actual proofs of the results can be found in Sections 3–6. Several numerical examples are discussed in Section 7. The results are compared to existing results in Section 8. Finally, Section 9 summarizes the results and suggests some directions for future work. Supplementary material can be found in the Appendix. References to the Appendix are prefixed by an "A.".

2. The Main Results

The main result consists of three parts: a basic bound, and specialized estimates for two classes of kernel functions. The basic perturbation bound deals with the approximation error based on the norms of certain error matrices. The norms of these error matrices are estimated for kernels with uniformly bounded eigenfunctions, and for kernels with bounded diagonal. Note that the scaling property is already present in the basic perturbation bound, and not a consequence of the estimates of the norms of the error matrices.

2.1 Preliminaries

We consider the following setting: Let k be a Mercer kernel on a probability space $\mathcal{X}$ with probability measure μ. This means that k can be written as

$$k(x,y) = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(y),$$

where $(\lambda_i)_{i\in\mathbb{N}}$ is a sequence of summable non-negative, non-increasing numbers, and $(\psi_i)_{i\in\mathbb{N}}$ is a family of mutually orthogonal unit norm functions with respect to the scalar product $(f,g) \mapsto \int_{\mathcal{X}} fg\,d\mu$. The λ_i are the eigenvalues and the ψ_i the eigenfunctions of the integral operator T_k which maps f to the function $x \mapsto \int_{\mathcal{X}} k(x,y) f(y) \mu(dy)$. Slightly abbreviating the true relationships, we will call λ_i the eigenvalues and ψ_i the eigenfunctions of k.

Let $X_1, \ldots, X_n$ be an i.i.d. sample from μ. The (normalized) kernel matrix is the $n \times n$ matrix $\mathbf{K}_n$ with entries

$$[\mathbf{K}_n]_{ij} := \frac{1}{n} k(X_i, X_j).$$

Denote the (random) eigenvalues of $\mathbf{K}_n$ by $l_1 \geq \ldots \geq l_n \geq 0$. These eigenvalues of $\mathbf{K}_n$ converge to their asymptotic counterparts $(\lambda_i)_{i\in\mathbb{N}}$ (see, for example, the papers by Koltchinskii and Giné, 2000, and Dauxois et al., 1982, or more recently, the Ph.D. thesis of von Luxburg, 2004).

For kernels with an infinite number of non-zero eigenvalues, k can be decomposed into a degenerate kernel $k^{[r]}$ and an error function e^r given a *truncation point* r:

$$\begin{aligned} k^{[r]}(x,y) &:= \sum_{i=1}^{r} \lambda_i \psi_i(x)\psi_i(y), \\ e^r(x,y) &:= k(x,y) - k^{[r]}(x,y). \end{aligned} \tag{1}$$

Note that $k^{[r]}$ and e^r are both Mercer kernels as well. The kernel matrices induced by $k^{[r]}$ and e^r will be denoted by $\mathbf{K}_n^{[r]}$ and $\mathbf{E}_n^r$, respectively, such that $\mathbf{E}_n^r = \mathbf{K}_n - \mathbf{K}^{[r]}$.

Furthermore, let Ψ_n^r be the $n \times r$ matrix with entries

$$[\Psi_n^r]_{i\ell} = \frac{1}{\sqrt{n}} \psi_\ell(X_i).$$

The ℓth column of Ψ_n^r is thus the sample vector of the eigenfunction ψ_ℓ. Therefore, Ψ_n^r is called the *eigenfunction sample matrix*. Using Ψ_n^r, we can write $\mathbf{K}_n^{[r]} = \Psi_n^r \operatorname{diag}(\lambda_1, \ldots, \lambda_r) \Psi_n^{r\top}$ (compare Equation (3)).

The norm of a matrix $\|\mathbf{A}\|$ will always be the operator norm $\max_{\|x\|=1} \|\mathbf{A}x\|$. The ith eigenvalue of a matrix $\mathbf{A}$ in decreasing order will be denoted by $\lambda_i(\mathbf{A})$.

2.2 The Basic Perturbation Bound

The following theorem forms the basis for the finite sample size bounds which we will present. It is a deterministic bound which also holds for non-random choices of points $x_1, \ldots, x_n$.

Theorem 1 (Basic Perturbation Bound) *For $1 \le r \le n$, $1 \le i \le n$,*

$$|l_i - \lambda_i| \le \lambda_i \|\mathbf{C}_n^r\| + \lambda_r + \|\mathbf{E}_n^r\|,$$

with $\mathbf{C}_n^r = \Psi_n^{r\top} \Psi_n^r - \mathbf{I}_r$.

The bound consists of two competing terms. Let us introduce the following symbols and names for the error terms:

$$\begin{aligned} C(r,n) &= \|\mathbf{C}_n^r\|, && \text{(relative error term)} \\ E(r,n) &= \lambda_r + \|\mathbf{E}_n^r\|. && \text{(absolute error term)} \end{aligned}$$

These two terms will be bounded under different assumptions on the kernel matrix.

The relative error term $C(r,n)$ measures the amount of non-orthogonality of the sample vectors of the first r eigenfunctions of k. As $n \to \infty$, $C(r,n) \to 0$ almost surely because the scalar products between the sample vectors converge to the scalar product with respect to μ and the ψ_i form an orthogonal family of unit norm functions with respect to that scalar product. The absolute error term $E(r,n)$ measures the effect of the truncation of the kernel function. Consequently, as $r \to \infty$, $E(r,n) \to 0$. On the other hand, both terms compete against each other, because for $r \to \infty$, $C(r,n) \to \infty$, and $E(r,n)$ does in general not converge to zero as $n \to \infty$. Depending on the choice of r (see below), the bound will have a characteristic shape which first scales with λ_i while the first term dominates, until, for large i (and small eigenvalues), the bound stagnates at a certain level. Also note that if the kernel is degenerate (has only a finite number of non-zero eigenvalues), the bound will be fully relative.

We see that r has to be chosen to balance these two terms. Trivially, the best bound is obtained by minimizing with respect to r, which gives the following corollary.

Corollary 2 *For all* $1 \leq i \leq n$,

$$|l_i - \lambda_i| \leq \min_{1 \leq r \leq n} (\lambda_i C(r,n) + E(r,n)).$$

Note that the optimal choice of r can not be easily computed in general since the choice depends on the true eigenvalues and, as we will see below, the form of the bounds on C and E might not allow to write down the minimizer in closed form.

However, even suboptimal choices of r can lead to meaningful bounds and insights. For the two classes of kernel functions considered below, we will discuss three alternatives with increasing dependency on i, the index of the eigenvalue considered, and the sample size n: (i) *Keep r fixed.* This choice will typically lead to good bounds when $i < r$. However, the bound does not converge to zero as $n \to \infty$. (ii) *Choose r according to i, for example $r = i$.* This choice can be used to show that the bounds decay quickly as i increases. Again, the bound does not converge to zero. (iii) *Choose r according to n.* The goal is to let r grow slowly with n to ensure that the overall bound converges to zero, showing the asymptotic rate of the bound. This case will be discussed in more depth in Section 6.

2.3 Estimates I: Bounded Eigenfunctions

The first class of kernel functions which we consider are Mercer kernels whose eigenfunctions ψ_i are uniformly bounded. An example for this case is given by ψ_i being a sine basis on $\mathcal{X} = [0, 2\pi]$. In the following, let $\Lambda_{>r} = \sum_{i=r+1}^{\infty} \lambda_i$. Convergence of this series follows from the requirement that $(\lambda_i) \in \ell^1$, and $\lambda_i \geq 0$.

Theorem 3 (Bounded Eigenfunctions) *Let k be a Mercer kernel with bounded eigenfunctions, $|\psi_i(x)| \leq M < \infty$ for all $i \in \mathbb{N}$, $x \in \mathcal{X}$. Then, for $1 \leq r \leq n$, with probability larger than $1 - \delta$,*

$$C(r,n) < M^2 r \sqrt{\frac{2}{n} \log \frac{r(r+1)}{\delta}}, \qquad E(r,n) < \lambda_r + M^2 \Lambda_{>r}$$

Consequently, Theorem 1 implies that

$$|l_i - \lambda_i| = O(\lambda_i r \sqrt{\log r} n^{-\frac{1}{2}} + \Lambda_{>r}).$$

Since the eigenfunctions are uniformly bounded, the estimation errors involved in $C(r,n)$ can be bounded conveniently using the Hoeffding inequality uniformly over all $r(r+1)/2$ entries of $\mathbf{C}_n^r$. In particular, in contrast to the bound derived in the next section, $C(r,n)$ does not depend on the eigenvalues. Moreover, $E(r,n)$ can be bounded in a deterministic fashion in this case.

Next we discuss different choices of r as explained at the end of the previous section. For any fixed r, the bound converges to $\lambda_r + M^2 \Lambda_{>r}$ with the usual stochastic convergence rate of $O(n^{-\frac{1}{2}})$. Unless $\Lambda_{>r} = 0$, the bound will not converge to zero.

Setting $r = i$, we see that the bound converges to $\lambda_i + M^2 \Lambda_{>i} = O(\Lambda_{>i})$. This term decays quickly as $i \to \infty$. For example, if $\lambda_i = O(i^{-\alpha})$ for some $\alpha > 1$, then $\Lambda_{>i} = O(i^{1-\alpha})$, and if $\lambda_i = O(e^{-\beta i})$ for some $\beta > 0$, then $\Lambda_{>i} = O(e^{-\beta i})$ (see Theorem A.4 in the Appendix). From these considerations we see that although the bound does not vanish as $n \to \infty$ for this choice of r, the bound scales with the true eigenvalue at a rate which is only slightly slower. This error is still much smaller than that given by non-scaling error bounds, unless the sample size is very large.

Eigenvalues	rate for $r(n)$	error rate
$\lambda_i = O(i^{-\alpha})$, $\alpha > 1$	$r(n) = n^{\frac{1}{2\alpha}}$	$n^{\frac{1-\alpha}{2\alpha}}\sqrt{\log n}$
$\lambda_i = O(e^{-\beta i})$, $\beta > 0$	$r(n) = \log n^{\frac{1}{2\beta}}$	$n^{-\frac{1}{2}}(\log n)^{\frac{3}{2}}$

Table 1: Optimal rates for $r(n)$ and the resulting rates for the upper bound on the approximation error for the case of kernels with bounded eigenfunctions.

Finally, choosing r to grow with n will ensure that the bound vanishes as $n \to \infty$, but this choice will also lower the rate of $C(r,n) \to 0$ such that the resulting overall rate will be sub-stochastic. In Table 2, the optimal rates for $r(n)$ and the resulting rates for the bound are shown for the cases of polynomial and exponential decay of the true eigenvalues (see Section 6 for the proofs). We also see that in the best case (for $\alpha \to \infty$ in the polynomial case, and also for the exponential case), we obtain a rate which is slower than $O(n^{-1/2})$ only by a log-factor, which is almost negligible.

2.4 Estimates II: Bounded Kernel Function

Since the restriction of uniformly bounded eigenfunctions is rather severe, we next consider the case where the kernel function is bounded. More specifically, we will require that the *diagonal* $x \mapsto k(x,x)$ is bounded. A prominent example for such kernel functions are radial-basis kernel functions. Typically, these are kernel functions on normed spaces which are written as

$$k(x,y) = g(\|x-y\|),$$

where g is a bounded function. The choice $g(a) = \exp(-a^2/2\sigma)$ is often simply called the *rbf-kernel with kernel width* σ.

In the following theorem, two independent estimates of the error terms are presented, one based on the Bernstein inequality, and the other based on the Chebychev inequality. The reason for presenting two bounds is that while the bound based on the Bernstein inequality is asymptotically faster, the bound based on the Chebychev inequality usually gives much smaller estimates for small sample sizes since the Bernstein bound contains an $O(n^{-1})$ term which can have a prohibitively large constant if one considers small eigenvalues.

Theorem 4 (Bounded Kernel Function) *Let k be a Mercer kernel with $k(x,x) \le K < \infty$ for all $x \in \mathcal{X}$. Then, for $1 \le r \le n$, with probability larger than $1-\delta$,*

$$C(r,n) < r\sqrt{\frac{2K}{n\lambda_r}\log\frac{2r(r+1)}{\delta}} + \frac{4Kr}{3n\lambda_r}\log\frac{2r(r+1)}{\delta},$$
$$E(r,n) < \lambda_r + \Lambda_{>r} + \sqrt{\frac{2K\Lambda_{>r}}{n}\log\frac{2}{\delta}} + \frac{2K}{3n}\log\frac{2}{\delta}.$$

Consequently, by Theorem 1,

$$|l_i - \lambda_i| = O(\lambda_i\lambda_r^{-\frac{1}{2}} r\sqrt{\log r}\, n^{-\frac{1}{2}} + \Lambda_{>r} + \sqrt{\Lambda_{>r}}\, n^{-\frac{1}{2}} + \lambda_i\lambda_r^{-1} n^{-1} r\log r + n^{-1}).$$

Eigenvalues	rate for $r(n)$	error rate
$\lambda_i = O(i^{-\alpha})$, $\alpha > 1$	$r(n) = n^{\frac{1}{2+3\alpha}}$	$n^{\frac{1-\alpha}{2+3\alpha}}$
$\lambda_i = O(e^{-\beta i})$, $\beta > 0$	$r(n) = \log n^{\frac{1}{3\beta}}$	$n^{-\frac{1}{3}}(\log n)^2$

Table 2: Optimal rates for $r(n)$ and the resulting rates for the upper bound on the approximation error for bounded kernels.

The $\lambda_i\lambda_r^{-1}n^{-1}r\log r$ term in this bound can become prohibitively large for small n and small λ_r. In this case, an alternative bound gives more realistic estimates for moderately small δ:

$$C(r,n) < r\sqrt{\frac{2r(r+1)K}{2\lambda_r n\delta}}, \qquad E(r,n) < \lambda_r + \Lambda_{>r} + \sqrt{\frac{2K\Lambda_{>r}}{n\delta}}. \tag{2}$$

For these bounds,

$$|l_i - \lambda_i| = O(\lambda_i\lambda_r^{-\frac{1}{2}}r^2 n^{-\frac{1}{2}} + \Lambda_{>r} + \sqrt{\Lambda_{>r}}n^{-\frac{1}{2}}).$$

These bounds give a similar picture as those for bounded eigenfunctions. The most significant difference is the occurrence of $\lambda_r^{-1/2}$ and λ_r^{-1} in $C(r,n)$. These terms appear because the eigenfunctions of bounded kernels may have values as large as $\sqrt{K/\lambda_r}$, leading to large second moments of the eigenfunctions and large error terms in $\|\mathbf{C}_n^r\|$.

These observations are also mirrored by the asymptotic rates for the more realistic bound (2) which are summarized in Table 2 (and proved in Section 6). At most, we obtain a rate of $n^{-1/3}$. However, as we will see in Section 7.3, for small sample sizes, the resulting bounds are still much tighter than those for non-scaling bounds.

Overview of Sections 3–6

In the next four sections we will prove the main results. We have tried to make the proofs as self-contained as possible. The derivation of the basic perturbation result relies on several results from the perturbation theory of symmetric matrices which are collected in the Appendix, while the estimates of the norm of the error matrices in Section 4 and 5 rely on standard large deviation bounds. Those two sections could be informative for improving the error estimates in the presence of additional *a priori* information. Readers not interested in the technical details can safely skip to page 2318 where examples are presented and the discussion of the results is continued. That discussion does not refer to details of the proofs.

3. The Basic Perturbation Bound

In this section, we prove the basic perturbation bound (Theorem 1) which derives a bound on the perturbation in terms of the norms of certain error matrices. The proof uses two classic results on the perturbation of symmetric matrices attributed to Weyl and Ostrowski.

Recall that the kernel function k is decomposed into a degenerate kernel $k^{[r]}$ obtained by truncation, and the error term e^r (see Equation (1)). From these functions, we form the $n \times n$ matrices $\mathbf{K}_n^{[r]}$

and $\mathbf{E}_n^r$ with entries

$$[\mathbf{K}_n^{[r]}]_{ij} = \frac{1}{n} k^{[r]}(X_i, X_j), \qquad [\mathbf{E}_n^r]_{ij} = \frac{1}{n} e^r(X_i, X_j).$$

Therefore, $\mathbf{K}_n = \mathbf{K}_n^{[r]} + \mathbf{E}_n^r$, such that $\mathbf{K}_n$ is an additive perturbation of $\mathbf{K}_n^{[r]}$ by $\mathbf{E}_n^r$. The effect on individual eigenvalues of such perturbations is addressed by Weyl's theorem (Theorem A.1).

Lemma 5 *For* $1 \leq i \leq n$, $r \in \mathbb{N}$,

$$|\lambda_i(\mathbf{K}_n^{[r]}) - l_i| \leq \|\mathbf{E}_n^r\|.$$

Proof By Weyl's theorem,

$$|\lambda_i(\mathbf{K}_n^{[r]}) - \lambda_i(\mathbf{K}_n^{[r]} + \mathbf{E}_n^r)| \leq \|\mathbf{E}_n^r\|,$$

and $\mathbf{K}_n^{[r]} + \mathbf{E}_n^r = \mathbf{K}_n$. ■

For the degenerate kernel matrix $\mathbf{K}_n^{[r]}$, we will derive a multiplicative bound on the approximation error of the eigenvalues. The main step is to realize that the kernel matrix of the truncated kernel can be written as the multiplicative perturbation of the diagonal matrix containing the true eigenvalues: Recall that $[\Psi_n^r]_{i\ell} = \psi_\ell(X_i)/\sqrt{n}$ (see Section 2.1) and let $\Lambda^r = \mathrm{diag}(\lambda_1, \ldots, \lambda_r)$. Then, we can easily verify that for all $r, n \in \mathbb{N}$, $\mathbf{K}_n^{[r]} = \Psi_n^r \Lambda^r \Psi_n^{r\top}$, since

$$[\Psi_n^r \Lambda^r \Psi_n^{r\top}]_{ij} = \sum_{\ell=1}^{r} [\Psi_n^r]_{i\ell} [\Lambda^r]_{\ell\ell} [\Psi_n^r]_{j\ell} = \frac{1}{n} \sum_{\ell=1}^{r} \psi_\ell(X_i) \lambda_\ell \psi_\ell(X_j) = \frac{1}{n} k^{[r]}(X_i, X_j). \tag{3}$$

Applying Ostrowski's Theorem (Theorem A.2 and its Corollary) leads to a multiplicative bound for the eigenvalues of $\mathbf{K}_n^{[r]}$:

Lemma 6 *For* $1 \leq i \leq r \leq n$,

$$|\lambda_i(\mathbf{K}_n^{[r]}) - \lambda_i| \leq \lambda_i \|\mathbf{C}_n^r\|.$$

Proof By Ostrowski's theorem,

$$|\lambda_i(\Psi_n^r \Lambda^r \Psi_n^{r\top}) - \lambda_i(\Lambda^r)| \leq |\lambda_i(\Lambda^r)| \|\Psi_n^{r\top} \Psi_n^r - \mathbf{I}\| = \lambda_i \|\mathbf{C}_n^r\|,$$

and $\lambda_i(\Psi_n^r \Lambda^r \Psi_n^{r\top}) = \lambda_i(\mathbf{K}_n^{[r]})$, $|\lambda_i(\Lambda^r)| = \lambda_i$, since $\lambda_i \geq 0$. ■

Combining this bound for $\mathbf{K}_n^{[r]}$ with the error induced by the truncation as in Lemma 5 results in the proof of Theorem 1.

Proof (of Theorem 1) For $i \leq r$, by Lemma 6,

$$|\lambda_i(\mathbf{K}_n^{[r]}) - \lambda_i| \leq \lambda_i \|\mathbf{C}_n^r\|.$$

For $i > r$, since $\lambda_i(\mathbf{K}_n^{[r]}) = 0$,

$$|\lambda_i(\mathbf{K}_n^{[r]}) - \lambda_i| = |\lambda_i| = \lambda_i.$$

Thus,

$$|l_i - \lambda_i| \leq |l_i - \lambda_i(\mathbf{K}_n^{[r]})| + |\lambda_i(\mathbf{K}_n^{[r]}) - \lambda_i| \leq \|\mathbf{E}_n^r\| + \begin{cases} \lambda_i \|\Psi_n^{r\top} \Psi_n^r - \mathbf{I}\|, & (1 \leq i \leq r) \\ \lambda_i & (r < i \leq n). \end{cases}$$

where $|l_i - \lambda_i(\mathbf{K}_n^{[r]})|$ has been bounded using Lemma 5. Now, since $\lambda_i \leq \lambda_r$ for $r < i \leq n$,

$$|l_i - \lambda_i| \leq \lambda_i \|\mathbf{C}_n^r\| + \lambda_r + \|\mathbf{E}_n^r\|,$$

and the theorem is proven. ■

4. Estimates I: Bounded Eigenfunctions

In this section, we will prove Theorem 3. We consider the case where the eigenfunctions are uniformly bounded and there exists an $M < \infty$ such that for all $i \in \mathbb{N}$ and $x \in \mathcal{X}$,

$$|\psi_i(x)| \leq M.$$

Lemma 7 *For $1 \leq r \leq n$, with probability larger than $1-\delta$,*

$$\|\mathbf{C}_n^r\| < M^2 r \sqrt{\frac{2}{n} \log \frac{r(r+1)}{\delta}}.$$

Proof Let

$$c_{\ell m} = [\mathbf{C}_n^r]_{\ell m} = \frac{1}{n} \sum_{i=1}^{n} \psi_\ell(X_i)\psi_m(X_i) - \delta_{\ell m}.$$

Note that

$$-M^2 - \delta_{\ell m} \leq \psi_\ell(X_i)\psi_m(X_i) - \delta_{\ell m} \leq M^2 - \delta_{\ell m},$$

such that the range of $\psi_\ell(X_i)\psi_m(X_i) - \delta_{\ell m}$ is given by $2M^2$. Using Hoeffding's inequality, it follows that

$$\mathrm{P}\{|c_{\ell m}| \geq \varepsilon\} \leq 2 \exp\left(-\frac{2n\varepsilon^2}{4M^4}\right). \tag{4}$$

In order to bound $\|\mathbf{C}_n^r\|$, recall that $\|\mathbf{C}_n^r\| \leq r \max_{1\leq \ell,m\leq r} |c_{\ell m}|$ and therefore,

$$\mathrm{P}\{\|\mathbf{C}_n^r\| \geq \varepsilon\} \leq \mathrm{P}\left\{\max_{1\leq \ell,m\leq r} |c_{\ell m}| \geq \frac{\varepsilon}{r}\right\}.$$

Since $c_{\ell m} = c_{m\ell}$, there are $r(r+1)/2$ different elements in the maximum. Thus, by the union bound,

$$\mathrm{P}\left\{\max_{1\leq \ell,m\leq r} |c_{\ell m}| \geq \frac{\varepsilon}{r}\right\} \leq \sum_{\ell \geq m} \mathrm{P}\left\{|c_{\ell m}| \geq \frac{\varepsilon}{r}\right\} \leq r(r+1) \exp\left(-\frac{n\varepsilon^2}{2M^4 r^2}\right)$$

by (4). Equating the right hand side with δ and solving for ε results in the claimed inequality. ■

In order to bound the size of $\|\mathbf{E}_n^r\|$ we use a non-probabilistic upper bound.

Lemma 8 *For $r, n \in \mathbb{N}$,*

$$\|\mathbf{E}_n^r\| \leq M^2 \sum_{i=r+1}^{\infty} \lambda_i.$$

Proof Recall that the entries of $\mathbf{E}_n^r$ are constructed by evaluating the error function $e^r(x,y)$ defined in (1) on all pairs (X_i,X_j) and dividing by n. For $x,y \in \mathcal{X}$,

$$\left|\frac{1}{n}e^r(x,y)\right| = \left|\frac{1}{n}\sum_{i=r+1}^{\infty}\lambda_i\psi_i(x)\psi_i(y)\right| \leq \frac{M^2}{n}\sum_{i=r+1}^{\infty}\lambda_i.$$

Therefore,

$$\|\mathbf{E}_n^r\| \leq n\max_{1\leq i,j\leq n}\left|\frac{1}{n}e^r(X_i,X_j)\right| \leq M^2\sum_{i=r+1}^{\infty}\lambda_i.$$

■

Based on the estimates from these two lemmas, we obtain the final result:

Proof (of Theorem 3) The result is a direct consequence of Theorem 1 and plugging in the estimates from Lemma 7 and 8 for the error terms. ■

5. Estimates II: Bounded Kernel Function

In this section, we treat the case of bounded kernel functions. We have split this section into three subsections, treating the relative error term $\|\mathbf{C}_n^r\|$, the absolute error term $\lambda_r + \|\mathbf{E}_n^r\|$, and the proof of Theorem 4 separately.

Throughout this section, we assume that there exists a $K < \infty$ such that for all $x \in \mathcal{X}$, $k(x,x) \leq K$. From this condition, one can derive upper bounds on individual eigenfunctions ψ_i and the error function e^r. The following easy lemma will prove to be very useful.

Lemma 9 *For $I \subseteq \mathbb{N}$,*

$$0 \leq \sum_{i\in I}\lambda_i\psi_i^2(x) \leq k(x,x) \leq K$$

for all $x \in \mathcal{X}$, and in particular $|\psi_i(x)| \leq \sqrt{K/\lambda_i}$. Consequently, the diagonal of the error function e^r is bounded by $0 \leq e^r(x,x) \leq K$ for all $r \in \mathbb{N}$.

Proof Since all the summands $\lambda_i\psi_i^2(x)$ are positive,

$$K \geq k(x,x) = \sum_{i=1}^{\infty}\lambda_i\psi_i^2(x) \geq \sum_{i\in I}\lambda_i\psi_i^2(x) \geq 0.$$

The bound on ψ_i follows for $I = \{i\}$, and the bound on e^r for $I = \{r+1,\ldots\}$. ■

5.1 The Relative Error Term

We begin by discussing the relative error term. The first step consists in computing an upper bound on the variance of the random variables from which $\mathbf{C}_n^r$ is constructed.

Lemma 10 *For $\ell, m \in \mathbb{N}$,*

$$\begin{aligned} \mathrm{E}_\mu(\psi_\ell^2 \psi_m^2) &\le \min(K/\lambda_\ell, K/\lambda_m), \\ \mathrm{Var}_\mu(\psi_\ell \psi_m - \delta_{\ell m}) &\le \min(K/\lambda_\ell, K/\lambda_m) - \delta_{\ell m}. \end{aligned}$$

Proof By the Hölder inequality,

$$\mathrm{E}_\mu(\psi_\ell^2 \psi_m^2) \le \mathrm{E}_\mu(|\psi_\ell^2|) \sup_{x \in \mathcal{X}} |\psi_\ell^2(x)| \le \frac{K}{\lambda_\ell},$$

because $\mathrm{E}_\mu(|\psi_\ell^2|) = \|\psi_\ell\|^2 = 1$, and by Lemma 9. The same bound holds with ℓ and m interchanged which proves the first inequality.

The second inequality follows from the definition of the variance and the fact that $\mathrm{E}_\mu(\psi_i \psi_j) = \delta_{ij}$:

$$\mathrm{Var}_\mu(\psi_\ell \psi_m - \delta_{\ell m}) = \mathrm{Var}_\mu(\psi_\ell \psi_m) = \mathrm{E}_\mu(\psi_\ell^2 \psi_m^2) - (\mathrm{E}_\mu \psi_\ell \psi_m)^2 \le \min(K/\lambda_\ell, K/\lambda_m) - \delta_{\ell m}.$$

■

Lemma 11 *For $1 \le r \le n$, with probability larger than $1 - \delta$,*

$$\|\mathbf{C}_n^r\| < r \sqrt{\frac{2K}{n\lambda_r} \log \frac{r(r+1)}{\delta}} + \frac{4rK}{3n\lambda_r} \log \frac{r(r+1)}{\delta}$$

Proof Let

$$c_{\ell m} = [\mathbf{C}_n^r]_{\ell m} = \frac{1}{n} \sum_{i=1}^n \psi_\ell(X_i) \psi_m(X_i) - \delta_{\ell m}.$$

Then, for $1 \le \ell \le r$, by Lemma 9, $\sup_{x \in \mathcal{X}} |\psi_\ell(x) \psi_\ell(x)| \le K/\lambda_r$,

$$-\frac{K}{\lambda_r} - \delta_{\ell m} \le c_{\ell m} \le \frac{K}{\lambda_r} - \delta_{\ell m},$$

and the range of $c_{\ell m}$ has size $M := 2K/\lambda_r$.

We can bound the variance of $\psi_\ell(X_i)\psi_m(X_i) - \delta_{\ell m}$ using Lemma 10 as follows:

$$\mathrm{Var}_\mu(\psi_\ell \psi_m - \delta_{\ell m}) \le \frac{K}{\lambda_r} =: \sigma^2.$$

By the Bernstein inequality (see for example van der Vaart and Wellner, 1996),

$$\mathrm{P}\{|c_{\ell m}| \ge \varepsilon\} \le 2 \exp\left(-\frac{n\varepsilon^2}{2\sigma^2 + 2M\varepsilon/3} \right).$$

In the proof of Lemma 7, we showed that

$$\mathrm{P}\{\|\mathbf{C}_n^r\| \ge \varepsilon\} \le \sum_{\ell \ge m} \mathrm{P}\left\{ |c_{\ell m}| \ge \frac{\varepsilon}{r} \right\}.$$

Thus,

$$\mathrm{P}\left\{\|\mathbf{C}_n^r\| \geq \varepsilon\right\} \leq r(r+1)\exp\left(-\frac{n(\varepsilon/r)^2}{2\sigma^2 + 2M\varepsilon/3r}\right).$$

Setting the right hand side equal to δ and solving for ε yields that with probability larger than $1-\delta$,

$$\|\mathbf{C}_n^r\| < \frac{2Mr}{3n}\log\frac{r(r+1)}{\delta} + r\sqrt{\frac{2\sigma^2}{n}\log\frac{r(r+1)}{\delta}}.$$

Substituting the values for σ^2 and M yields the claimed upper bound. ■

Corollary 12 *Alternatively, using the Chebychev inequality instead of the Bernstein inequality, one obtains that for* $1 \leq r \leq n$*, with probability larger than* $1-\delta$*,*

$$\|\mathbf{C}_n^r\| \leq r\sqrt{\frac{r(r+1)K}{2\lambda_r n\delta}}.$$

Proof By the Chebychev inequality,

$$\mathrm{P}\left\{|c_{\ell m}| \geq \varepsilon\right\} < \frac{\mathrm{Var}_\mu(\psi_\ell\psi_m - \delta_{\ell m})}{n\varepsilon^2} \leq \frac{K}{\lambda_r n\varepsilon^2}.$$

Thus,

$$\mathrm{P}\left\{\|\mathbf{C}_n^r\| \geq \varepsilon\right\} \leq \frac{r(r+1)}{2}\frac{Kr^2}{\lambda_r n\varepsilon^2}.$$

Equating the right hand side to δ and solving for ε proves the corollary. ■

5.2 The Absolute Error Term

Next, we study the properties of the random variable $\|\mathbf{E}_n^r\|$. Recall that $\mathbf{E}_n^r$ is obtained by evaluating the error function e^r on all pairs of samples (X_i, X_j). First of all, note that by the definition of e^r, the error function is itself a Mercer kernel such that $\mathbf{E}_n^r$ is positive-semidefinite for all sample realizations. Thus, we can bound $\|\mathbf{E}_n^r\| = \lambda_1(\mathbf{E}_n^r)$ by the trace of $\mathbf{E}_n^r$:

$$\|\mathbf{E}_n^r\| \leq \mathrm{tr}\,\mathbf{E}_n^r = \frac{1}{n}\sum_{i=1}^n e^r(X_i, X_i).$$

By the strong law of large numbers,

$$\frac{1}{n}\sum_{i=1}^n e^r(X_i, X_i) \to_{\text{a.s.}} \mathrm{E}(e^r(X,X)) =: t_r$$

with $X \sim \mu$, the common distribution of the X_i.

In this section, we will first compute $\mathrm{E}(e^r(X,X))$ in terms of the eigenvalues of k, and then derive a probabilistic bound on $\|\mathbf{E}_n^r\|$.

Lemma 13 *For $r \in \mathbb{N}$,*

$$t_r = \Lambda_{>r} := \sum_{i=r+1}^{\infty} \lambda_i.$$

Proof We compute t_r:

$$t_r = \int_{\mathcal{X}} e^r(x,x)\mu(dx) = \int_{\mathcal{X}} \left(\sum_{\ell=r+1}^{\infty} \lambda_\ell \psi_\ell^2(x) \right) \mu(dx) \overset{(1)}{=} \sum_{\ell=r+1}^{\infty} \lambda_\ell \int_{\mathcal{X}} \psi_\ell^2(x)\mu(dx) \overset{(2)}{=} \sum_{\ell=r+1}^{\infty} \lambda_\ell,$$

where at (1), the integration and summation commute because the function $x \mapsto K$ is an integrable majorant to the sum in parenthesis and Lebesgue's theorem, and (2) holds because $\int \psi_\ell^2(x)\mu(dx) = \|\psi_\ell\|^2 = 1$. ■

Since we are interested in the situation when t_r is much smaller than K, we will use the following bound on the variance.

Lemma 14 *For $r \in \mathbb{N}$,*

$$0 \le e^r(X,X) \le K, \qquad \mathrm{Var}(e^r(X,X)) \le K\mathrm{E}(e^r(X,X)) = Kt_r.$$

Proof The first inequality has been proven in Lemma 9. The variance can be bounded using the Hölder inequality as follows:

$$\begin{aligned}\mathrm{Var}(e^r(X,X)) &= \mathrm{E}(e^r(X,X)^2) - (\mathrm{E}e^r(X,X))^2 \\ &\le \mathrm{E}(|e^r(X,X)|)K - (\mathrm{E}e^r(X,X))^2 \le \mathrm{E}(e^r(X,X))K = t_r K.\end{aligned}$$

■

Lemma 15 *For $r,n \in \mathbb{N}$, with probability larger than $1-\delta$,*

$$\|\mathbf{E}_n^r\| < t_r + \sqrt{\frac{2Kt_r}{n}\log\frac{1}{\delta}} + \frac{2K}{3n}\log\frac{1}{\delta}.$$

Proof In order to apply Bernstein's inequality, we first have to compute the size of the range of $e^r(X_i,X_i)$ and its variance. In Lemma 14, we have proven that the range of $e^r(X_i,X_i)$ has size K, and that $\mathrm{Var}(e^r(X_i,X_i)) \le Kt_r$.

Thus, by the Bernstein inequality, with probability larger than $1-\delta$,

$$\mathrm{P}\left\{\|\mathbf{E}_n^r\| - t_r \ge \varepsilon\right\} \le \exp\left(-\frac{n\varepsilon^2}{2Kt_r + \frac{2K\varepsilon}{3}}\right).$$

Setting the right hand side equal to δ and solving for ε results in the claimed upper bound. ■

Again replacing the Bernstein inequality by the Chebychev inequality, one can show an alternative confidence bound which can be considerably smaller for moderately small δ and small n.

Corollary 16 *For $r,n \in \mathbb{N}$, with probability larger than $1-\delta$,*

$$\|\mathbf{E}_n^r\| < t_r + \sqrt{\frac{Kt_r}{n\delta}}.$$

5.3 The Final Result

We finally combine the estimates from the previous two sections to obtain the bound for bounded kernel functions.

Proof (of Theorem 4) The basic perturbation bound holds by Theorem 1. The upper bounds on $\|\mathbf{C}_n^r\|$ and $\|\mathbf{E}_n^r\|$ were derived in Lemmas 11 and 15. Finally, both estimates can be combined according from the individual bounds at confidence $\delta/2$.[1]

Using the alternative bounds from Corollary 12 and 16, one obtains the bounds from Equation (2). ■

6. Asymptotic Rates

In this section, we derive the optimal growth rates (up to logarithmic factors) for $r(n)$ such that the overall bound converges to zero. The computations have to be carried out for four different settings: kernels with bounded eigenfunctions/bounded kernels, and polynomial decay/exponential decay of eigenvalues.

6.1 Case I: Bounded Eigenfunctions

Polynomial Decay Assume that $\lambda_i = O(i^{-\alpha})$ with $\alpha > 1$. For fixed i, we wish to let r grow with n such that the approximation from Theorem 3 tends to 0. The rate is given as

$$|l_i - \lambda_i| = O(r\sqrt{\log r} n^{-\frac{1}{2}} + \Lambda_{>r}).$$

We omit the $\sqrt{\log r}$ term first. From $\lambda_i = O(i^{-\alpha})$, we obtain the following condition (see Appendix A.2 for rates concerning the tail sums $\Lambda_{>r}$):

$$r n^{-\frac{1}{2}} + r^{1-\alpha} = o(1).$$

We use the following Ansatz: $r = n^{\varepsilon}$ with $\varepsilon > 0$. Thus, we wish to find ε such that

$$n^{\varepsilon - \frac{1}{2}} + n^{\varepsilon(1-\alpha)} = o(1).$$

This condition is obviously met if $\varepsilon < 1/2$. We wish to balance the two terms in order to minimize the overall rate. This rate is attained if

$$\varepsilon - \frac{1}{2} = \varepsilon(1-\alpha) \quad \rightsquigarrow \quad \varepsilon = \frac{1}{2\alpha}.$$

Plugging in this rate shows that

$$|l_i - \lambda_i| = O(n^{\frac{1-\alpha}{2\alpha}} \sqrt{\log n}).$$

1. Let X, X' be positive random variables such that $\mathrm{P}\{X > \varepsilon\} \leq \delta$, $\mathrm{P}\{X' > \varepsilon'\} \leq \delta$. Then, $\mathrm{P}\{X + X' > \varepsilon + \varepsilon'\} \leq 2\delta$, because $\mathrm{P}\{X + X' > \varepsilon + \varepsilon'\} \leq \mathrm{P}\{X > \varepsilon \text{ or } X' > \varepsilon'\} \leq \mathrm{P}\{X > \varepsilon\} + \mathrm{P}\{X' > \varepsilon'\} \leq 2\delta$.

Exponential Decay We assume that $\lambda_i = O(e^{-\beta i})$, $\beta > 0$, such that $\Lambda_{>r} = O(e^{-\beta r})$. We are looking for the slowest rate such that $\Lambda_{>r} = O(n^{-\frac{1}{2}})$. Using the Ansatz $r = \log n^{\varepsilon}$, we obtain the condition

$$e^{-\beta \log n^{\varepsilon}} = n^{-\beta\varepsilon} = O(n^{-\frac{1}{2}}) \qquad \text{if} \qquad -\beta\varepsilon \leq -\frac{1}{2} \quad \rightsquigarrow \quad \varepsilon = \frac{1}{2\beta}.$$

Plugging this choice of ε gives the overall rate of

$$|l_i - \lambda_i| = O(n^{-\frac{1}{2}}(\log n)^{\frac{3}{2}}).$$

6.2 Case II: Bounded Kernel function

In this case, the rate is (using the bound based on the Chebychev inequality)

$$|l_i - \lambda_i| = O(\lambda_r^{-\frac{1}{2}} r^2 n^{-\frac{1}{2}} + \Lambda_{>r} + \sqrt{\Lambda_{>r}} n^{-\frac{1}{2}}).$$

Polynomial Decay Plugging in $\lambda_r = r^{-\alpha}$, $\Lambda_{>r} = r^{1-\alpha}$ (omitting the constants) gives

$$r^{2+\frac{\alpha}{2}} n^{-\frac{1}{2}} + r^{1-\alpha} + r^{\frac{1-\alpha}{2}} n^{-\frac{1}{2}}.$$

We again set $r = n^{\varepsilon}$ and obtain the three terms

$$n^{\varepsilon(2+\frac{\alpha}{2})-\frac{1}{2}} + n^{\varepsilon(1-\alpha)} + n^{\varepsilon(\frac{1-\alpha}{2})-\frac{1}{2}}.$$

First of all, the first term tells us that $\varepsilon \leq \frac{1}{4+\alpha}$, otherwise the bound diverges. Also note that of the three terms, only the first two are relevant, because the third term is always smaller than the first term. They are balanced if

$$\varepsilon\left(\frac{4+\alpha}{2}\right) - \frac{1}{2} = \varepsilon(1-\alpha) \quad \rightsquigarrow \quad \varepsilon = \frac{1}{2+3\alpha},$$

which is also smaller than $\frac{1}{4+\alpha}$ for $\alpha > 1$. Plugging this into either term shows that the resulting rate is

$$|l_i - \lambda_i| = O(n^{\frac{1-\alpha}{2+3\alpha}}).$$

Exponential Decay In this case, $\lambda_r = e^{-\beta r}$, $\Lambda_{>r} = O(e^{-\beta r})$. Therefore, the rate becomes (omitting all constants)

$$e^{\frac{\beta}{2}r} r^2 n^{-\frac{1}{2}} + e^{-\beta r} + e^{-\frac{\beta}{2}r} n^{-\frac{1}{2}}.$$

With the Ansatz $r = \log n^{\varepsilon}$, we get

$$n^{\frac{\beta\varepsilon}{2}-\frac{1}{2}} (\log n^{\varepsilon})^2 + n^{-\beta\varepsilon} + n^{-\frac{\beta\varepsilon}{2}-\frac{1}{2}}.$$

From the first term we get that $\varepsilon \leq 1/\beta$, otherwise it diverges. But for $\varepsilon \leq 1/\beta$, the third term is always smaller than the second term, such that we have to balance the first and the second term. Thus, the optimal rate is given if

$$\frac{\beta\varepsilon}{2} - \frac{1}{2} = -\beta\varepsilon \quad \rightsquigarrow \quad \varepsilon = \frac{1}{3\beta}.$$

This choice results in the overall rate of

$$|l_i - \lambda_i| = O(n^{-\frac{1}{3}}(\log n)^2).$$

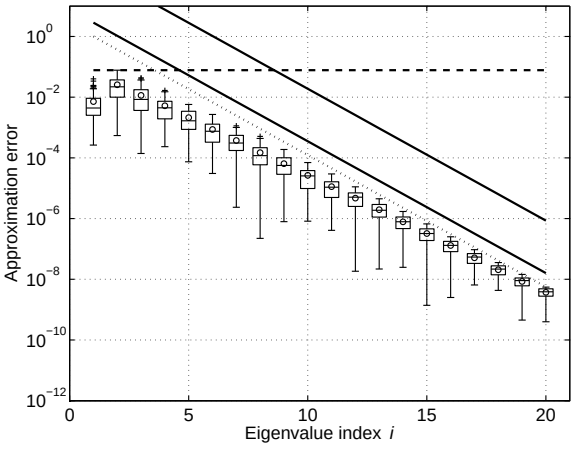


Figure 2: The example from the introduction revisited. The box plots show the distributions of the observed approximation errors for kernel matrices built from $n = 1000$ sample points over 100 re-samples. The two solid lines plot the approximation error bound derived in this work. The upper line uses the bound on $\|\mathbf{C}_n^r\|$ from Theorem 3, while the lower line uses the largest observed value of $\|\mathbf{C}_n^r\|$ on the samples in conjunction with Theorem 1, which requires knowledge of the true eigenfunctions.

7. Examples

We claim that the bounds which we have derived give realistic error estimates already for small sample sizes. In this section, we discuss several examples for both classes of kernels on numerical simulations to support our claim.

7.1 Examples for Kernels with Bounded Eigenfunctions

For the class of Mercer kernels whose eigenfunctions are uniformly bounded, we have been able to derive rather accurate finite sample size bounds. In particular, the truncation error $E(r,n)$ can be bounded in a deterministic fashion. The relative error term $C(r,n)$ scales rather moderately as $r\sqrt{\log r}$ with r, and $E(r,n)$ decays quickly, depending on the rate of decay of the eigenvalues, for both the case of polynomial and exponential decay.

Consider the following example already briefly discussed in the introduction. We construct a Mercer kernel function by specifying an orthogonal set of functions and a sequence of eigenvalues. As orthogonal functions, we use Legendre polynomials $P_n(x)$ (Abramowitz and Stegun, 1972), which are orthogonal polynomials on $[-1,1]$. We take the first 20 polynomials, and set $\lambda_i = \exp(-i)$. Then,

$$k(x,y) = \sum_{i=0}^{19} \nu_i e^{-i} P_i(x) P_i(y)$$

defines a Mercer kernel, where $\nu_i = 1/(2i+1)$ comes from normalization: $\sqrt{\nu_i}P_i$ has unit norm with respect to the probability measure induced by $\mu([a,b]) = |b-a|/2$.

For convenience, the plot from Figure 1(b) is reproduced in Figure 2. Since this kernel has only 20 non-zero eigenvalues, we obtain a purely relative bound (neglecting the round-off errors)

by setting $r = 20$. We see that the bound accurately reflects the true behavior of the approximation error.

We have also marked the smallest possible non-scaling error bound on the maximal observed approximation error. Any non-scaling error bound will necessarily be larger than this observed error with high probability. This plot illustrates the fact that it is essential for obtaining accurate estimates that the error bounds scale with the considered eigenvalue. A non-scaling bound will overestimate the error of smaller eigenvalues significantly.

The plot might suggest that our bound is actually worse for the first few large eigenvalues, but note that the dashed line is only a lower bound to any non-scaling error bound, and actual error bounds will typically be much larger.

Next, we turn to a non-degenerate kernel. In Figure 3, some examples are plotted for the sine-basis kernel, defined as follows. The eigenfunctions are given by

$$\psi_i(x) = \sqrt{2}\sin(ix/2), \qquad i \in \mathbb{N},$$

which form an orthogonal family of functions on the Hilbert space of functions defined on $[0, 2\pi]$ with the scalar product $(f,g) \mapsto \int_0^{2\pi} f(x)g(x)dx/2\pi$. These functions are uniformly bounded by $\sqrt{2}$.

Since we cannot write down the resulting kernel given some choice of eigenvalues in closed form, we truncate the expansion to the first 1000 terms, resulting in a negligible difference to the true kernel function. The resulting kernel function for different choices of eigenvalues are plotted in Figure 3(a). In Figures 3(b)–(d), three such examples are plotted, two for polynomially decaying eigenvalues, and one for exponentially decaying eigenvalues. We plot the bound for different choices of r and see that, with increasing r, the absolute error term becomes smaller such that the bound for small eigenvalues also becomes smaller while, at the same time, the bound for larger eigenvalues becomes larger.

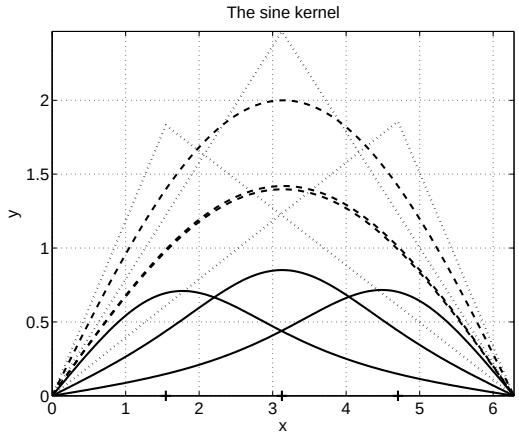
In Figure 3(d), it appears that the bound is actually smaller than the observed eigenvalues. This effect is due to the finite precision arithmetic used in the computations. These rounding errors effectively lead to an additive perturbation of the kernel matrix, which in turn results in an additive perturbation of the eigenvalues of the same magnitude. An interesting observation is that although our bounds fail to be purely relative in the general case, numerically computed eigenvalues will always display a stagnation of small eigenvalues at a certain level due to round-off errors as well. Thus, for numerically computed eigenvalues, fully relative approximation errors are not possible.

7.2 Examples for Kernels with Bounded Kernel Functions

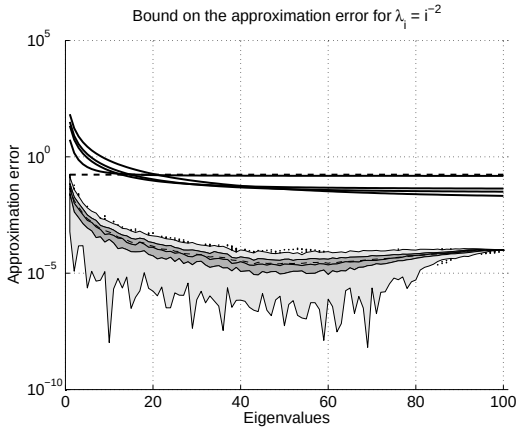
The second class of kernel functions are kernels with bounded diagonal. This class includes the important radial basis function kernels (rbf-kernels). In this case, the eigenfunctions can in principle grow unboundedly as the eigenvalues become smaller, leading to considerably larger error estimates. The most important difference to the previous case is that the relative error term depends on the eigenvalues themselves and scales with the factor $1/\sqrt{\lambda_r}$. Therefore, having smaller eigenvalues can lead to a much larger relative error term (which will nevertheless ultimately decay to zero).

The example we will consider is designed to display this slow rate of convergence. It is well-known that Bernoulli random variables maximize the variance among all bounded random variables taking values in $[0,1]$. We thus consider the following kernel: Let $(A_i)_{i=1}^{\infty}$ be a partition of $\mathcal{X}$ with $\mu(A_1) \geq \mu(A_2) \geq \ldots \geq 0$. Then, set

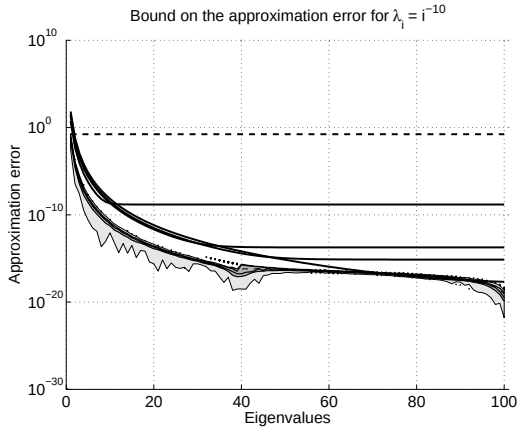
$$\lambda_i = \mu(A_i), \qquad \psi_i(x) = \frac{1}{\sqrt{\lambda_i}} 1_{A_i}(x). \tag{5}$$



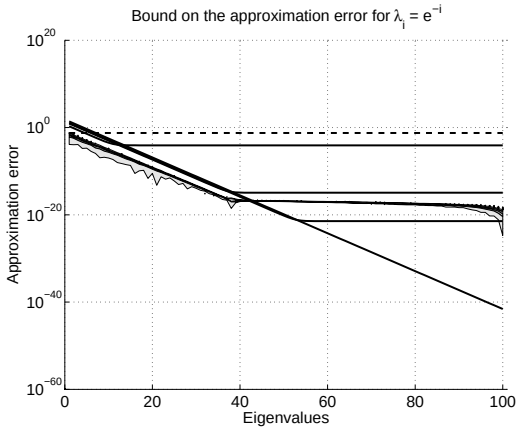
(a) The sine-kernel for different decay rates and at the three points marked by a plus-sign. The decay rates are $\lambda_i = i^{-2}$ (dotted lines), $\lambda_i = i^{-10}$ (dashed lines), $\lambda_i = e^{-i}$ (solid lines). Note that the smoothness depends on rate of decay.



(b) For quadratic decay, the bounds are only slightly better than the best possible non-scaling bound. (Shaded areas correspond to quartile ranges, similar to box plots. See explanation in figure caption.)



(c) For faster polynomial decay, the bounds are much more accurate than the best possible non-scaling bound for small eigenvalues.



(d) For exponential decay, the bounds are much more accurate than the best possible non-scaling bound as observed on the data. In fact, the actual approximation error becomes even larger than the bound due to finite precision arithmetics starting with eigenvalue λ_{40}.

Figure 3: The sine-kernel example. We consider the decay rates $\lambda_i = i^{-2}$, $\lambda_i = i^{-10}$, and $\lambda_i = e^{-i}$. In (a), some example kernel functions are plotted. In (b)–(d), we plot approximation errors as observed over 100 re-samples of $n = 200$ points uniformly sampled from $[0, 2\pi]$, and the bound for $r \in \{10, 35, 50, 100\}$ for confidence $\delta = 0.05$. The dashed line plots the best achievable non-scaling error bound. The distribution of the observed approximation errors is illustrated by differently shaded areas similar to box plots: dark gray area shows lower to upper quartile range, while light gray area shows data points which lie in 1.5 times the interquartile range. Points beyond that are plotted as small dots.

This defines a Mercer kernel

$$k(x,y) = \sum_{i=1}^{\infty} 1_{A_i}(x) 1_{A_i}(y) = \begin{cases} 1 & \text{if there exists an } i \text{ such that } x,y \in A_i, \\ 0 & \text{else.} \end{cases}$$

Note that the matrix $\mathbf{C}_n^r$ is always diagonal for this choice of basis functions because

$$\psi_i(x)\psi_j(x) = \frac{1}{\sqrt{\lambda_i \lambda_j}} \delta_{ij}.$$

Thus, we obtain a slightly improved bound over the one from Corollary 12 because $\|\mathbf{C}_n^r\| = \max_{1 \le i \le r} |[\mathbf{C}_n^r]_{ii}|$ since $\mathbf{C}_n^r$ is diagonal. Then,

$$\mathrm{P}\{\|\mathbf{C}_n^r\| \ge \varepsilon\} \le r \max_{1 \le i \le r} \mathrm{P}\{|[\mathbf{C}_n^r]_{ii}| \ge \varepsilon\} \le \frac{r}{\lambda_r n \varepsilon^2},$$

and consequently, with probability larger than $1-\delta$,

$$\|\mathbf{C}_n^r\| < \sqrt{\frac{r}{\lambda_r n \delta}}. \tag{6}$$

Figure 4 plots the bound for this example. Again, the kernel function cannot be computed in closed form, and we truncate to the first 1000 terms. We plot two different bounds, the bound from (6), and the general result from Theorem 4. Note that the error does not fluctuate after eigenvalue λ_{20}. The reason is that λ_i is so small that not a single point has hit A_i in the sample of $n = 1000$ points; the kernel is effectively degenerate and the approximate eigenvalues beyond l_{20} are equal to zero. In this case, the approximation error is equal to the true eigenvalue which explains the exponential decay. Note though, that for the first 20 eigenvalues, the (slower) rate is actually matched by the bound.

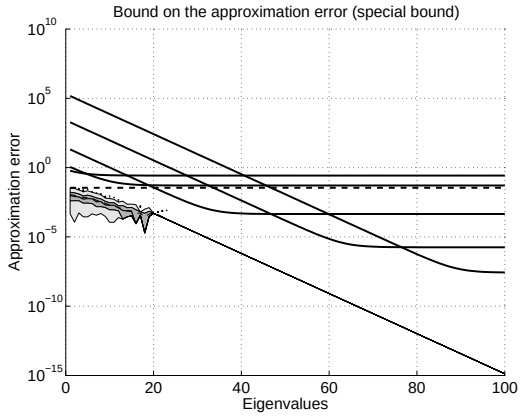
Over all, compared with the examples for bounded kernel functions, the bounds are considerably less tight, but they still correctly predict the scaling of the approximation error with regard to the true eigenvalue.

7.3 Comparisons with a Non-Scaling Hoeffding-Type Bound

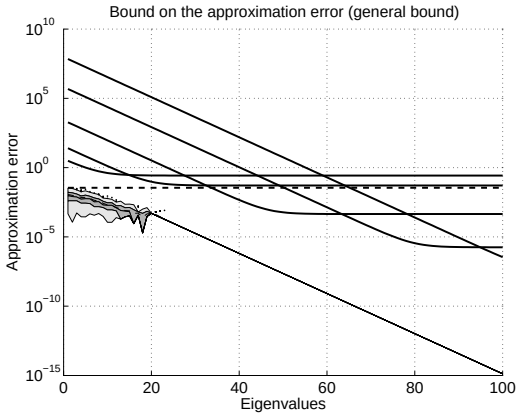
Finally, we would like compare our bound numerically against a non-scaling Hoeffding-type bound. As discussed in Section 6, while the bounds presented in this paper are more accurate for small eigenvalues, the overall rate as $n \to \infty$ is slower than the usual stochastic rate $O(n^{-1/2})$. To illustrate that the bounds can nevertheless be much more accurate even for moderately small sample sizes, we will compare our bounds against a Hoeffding-type bound which does not scale with the eigenvalue under consideration.

We face the problem of choosing an appropriate non-scaling bound. Such bounds exist, but only for tail sums of eigenvalues (see the papers by Shawe-Taylor et al., 2005, and Blanchard et al., 2006, and the discussion in Section 8). However, the full complexity of these bounds is not really necessary for the illustrative purposes we have in mind. In Theorem 6 of the paper by Shawe-Taylor et al. (2005), there is a bound on the concentration of single eigenvalues around their mean: with probability larger than $1-\delta$,

$$|l_i - \mathrm{E}(l_i)| \le K^2 \sqrt{\frac{1}{2n} \log \frac{2}{\delta}}.$$



(a) The bound based on the estimate from Equation (6) for $C(r,n)$ which was specifically derived for this example.



(b) The bound using the general result from Theorem 4. Note that although the bound becomes very large for large r, the minimum over all bounds is the final bound on the approximation error.

Figure 4: The indicator function example (see Equation (5)). This kernel has maximal variance given the constraint that the resulting kernel function is bounded. We consider eigenvalues $\lambda_i = e^{i/3}/Z$, where Z is the normalization constant. The sample size is $n = 1000$, and the bounds are computed for confidence $\delta = 0.05$. The solid lines are the bounds for $r \in \{10, 50, 200, 500\}$, while the dashed line shows the best possible non-scaling error bound.

This bound has the required asymptotic decay rate of $O(n^{-1/2})$. Terms similar to this bound also occur in the more complex bounds on tail sums of eigenvalues, albeit with larger constants. We will therefore pretend that this is an overly optimistic guess of the approximation error and compare our bounds to it.

We are particularly interested in the question if the Hoeffding-type bound, due to its better asymptotic rate, quickly compensates for its non-scaling constant and becomes as small as our bound. We therefore compare the bounds for sample sizes up to $n = 10000$. Figure 5 plots the Hoeffding-type bound with the bound derived in this work. For these plots, since the eigenvalues are known, the optimal r has been computed by numerically minimizing the bound. The optimal r with respect to i have been plotted in Figure 5(d). For polynomial decay of rate $\lambda_i = i^{-2}$, the bounds are clearly inferior to the Hoeffding-type bounds and one can also clearly see that the overall rate is sub-stochastic. However, for faster decay rates, the bounds for smaller eigenvalues are clearly superior, also demonstrating that the number of samples necessary to yield a comparably small bound using the Hoeffding-type bound is fairly large.

So far, we have compared the bounds only for the errors of individual eigenvalues. Let us now compare the bounds for sums of eigenvalues. As we will discuss in Section 8, there exist bounds which directly deal with tail sums of eigenvalues and are more accurate in this case. However, it is instructive to derive a rough estimate for tail sums based on our bounds. We start with bounding the difference between tail sums by summing the individual bounds:

$$\left| \sum_{i=r+1}^{n} l_i - \sum_{i=r+1}^{\infty} \lambda_i \right| \leq \sum_{i=r+1}^{n} |l_i - \lambda_i| + \Lambda_{>n+1} \leq \sum_{i=r+1}^{n} (\lambda_i C(r,n) + E(r,n)) + \Lambda_{>n+1}.$$

Let us roughly estimate the size of the resulting bound. The key to obtain a good estimate lies in choosing a different r for each i. Let us set $r = i$. Then, omitting constants and using the bound for bounded kernel functions which is based on the Chebychev inequality, we get that

$$\begin{aligned} &\sum_{i=r+1} \left(\sqrt{\lambda_i} i^2 n^{-\frac{1}{2}} + \Lambda_{>i} + \sqrt{\Lambda_{>i}} n^{-\frac{1}{2}} \right) + \Lambda_{>n+1} \\ &= n^{-\frac{1}{2}} \left(\sum_{i=r+1}^{n} i^2 \sqrt{\lambda_i} + \sum_{i=r+1}^{n} \sqrt{\Lambda_{>i}} \right) + \sum_{i=r+1}^{n} \Lambda_{>i} + \Lambda_{>n+1}. \end{aligned}$$

Let us consider these tail sums for $i \to \infty$. All of these sums converge if $\alpha > 6$, because

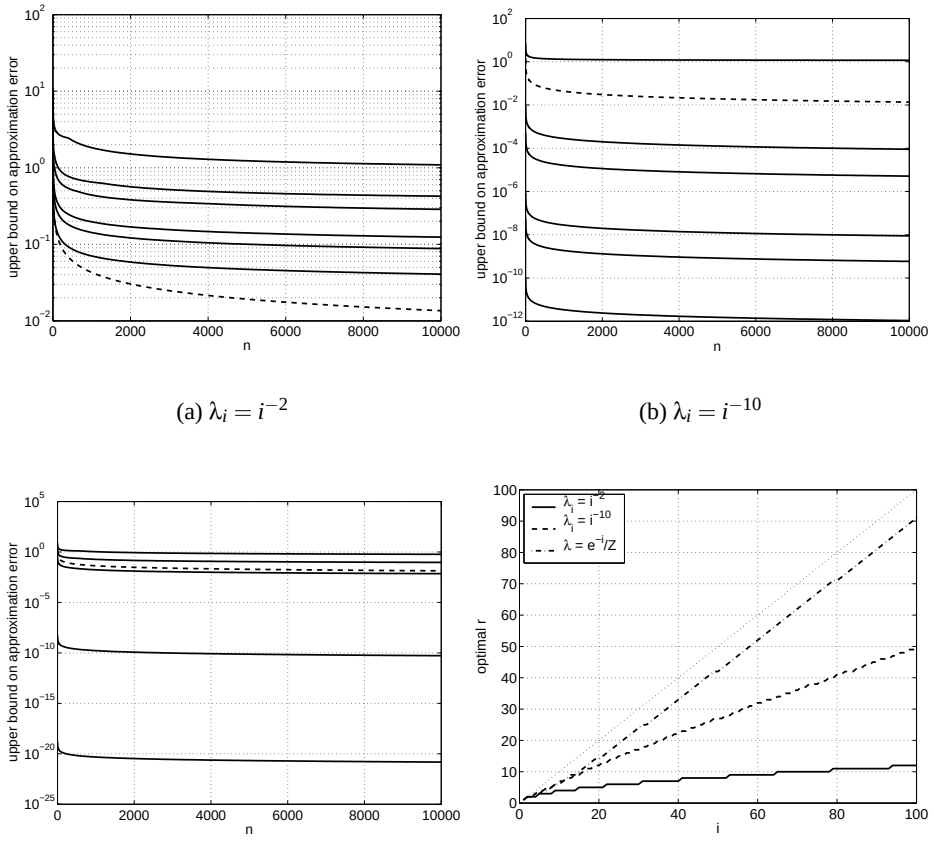
$$i^2 \sqrt{\lambda_i} = O(i^{2-\frac{\alpha}{2}}) = O(i^{-1}), \quad \sqrt{\Lambda_{>i}} = O(i^{\frac{1-\alpha}{2}}) = O(i^{-2\frac{1}{2}}), \quad \Lambda_{>i} = O(i^{1-\alpha}) = O(i^{-5}).$$

Thus, for large i, the bound on the tail sums actually becomes small, giving more accurate bounds than those obtainable by a non-scaling bound.

In Figure 6, we again compare the bound derived in this work against a Hoeffding-type bound for tail sums of eigenvalues. We do not use the rough estimate derived above, but sum the individual approximation error bounds selecting the optimal r in each case. Again we see that these bounds give much smaller estimates than the non-scaling Hoeffding-type bound.

8. Related Work

The bounds presented in this work are the first finite sample size bounds for single eigenvalues which scale with the eigenvalue under consideration. These results contribute to the already existing body of work which we briefly review in this chapter.



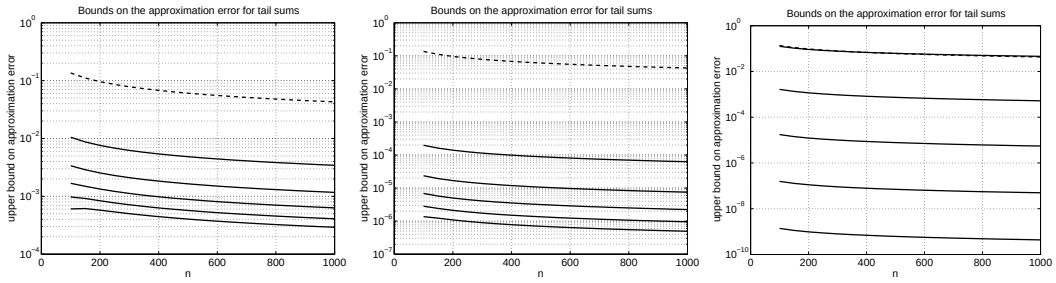
(a) $\lambda_i = i^{-2}$

(b) $\lambda_i = i^{-10}$

(c) $\lambda_i = e^{-i}/Z$ (normalization)

(d) optimal r

Figure 5: Upper bounds with the best choice of r (solid lines) compared with a Hoeffding-type non-scaling $O(n^{-\frac{1}{2}})$ bound (dashed line) for single eigenvalues. The bounds are plotted for eigenvalues $\lambda_1, \lambda_5, \lambda_{10}, \lambda_{50}, \lambda_{100}$, and for the cases of polynomial decay, also for λ_{500}. The truncation point r has been chosen optimally by explicitly minimizing the bound. The confidence was $\delta = 0.05$.



(a) $\lambda = i^{-2}$ (b) $\lambda = i^{-10}$ (c) $\lambda = e^{-i}/Z$ (normalization)

Figure 6: Upper bounds for tail sums (solid lines) compared with a Hoeffding-type non-scaling $O(n^{-\frac{1}{2}})$ bound (dashed line). The tail sums are plotted for $r \in \{10,20,30,40,50\}$. The confidence was set to $\delta = 0.5$.

The asymptotic setting is addressed for example in Dauxois et al. (1982) and Koltchinskii and Giné (2000) where central limit type results for the limit distributions of the eigenvalues are derive. The finite sample setting has been addressed more recently, in particular in Shawe-Taylor et al. (2005) and Blanchard et al. (2006).

The paper by Shawe-Taylor et al. (2005) discusses several aspects of the relation between the approximate eigenvalues and their asymptotic counterparts. These also include concentration inequalities relating the approximate eigenvalues to their expectations (similar inequalities can also be found in the Ph.D. thesis of Mika, 2002). Finally, Theorem 1 and 2 of that paper provide finite sample size bounds on the approximation error for tail sums of the eigenvalues. However, these bounds do not scale with the size of the eigenvalues, leading to the already discussed overestimation of the true approximation error in particular for small eigenvalues.

These results are further refined and extended in the paper by Blanchard et al. (2006). In particular, non-scaling bounds are derived exhibiting fast convergence rates, as well as scaling bounds for tail sums of eigenvalues. As already explained, the latter bounds are particularly important for obtaining accurate estimates for small eigenvalues.

Compared to the results presented in this work, all of these results are either dealing with the asymptotic setting or, in the case of finite sample size bounds, are non-scaling or only deal with tail sums of eigenvalues. Obtaining accurate bounds, in particular bounds which scale with the eigenvalue under consideration, was an open problem so far (see the comments below Theorem 4.2 in the paper by Blanchard et al., 2006). Note that these two problems are not interchangeable: While it is possible to construct bounds for single eigenvalues from bounds of tail sums by subtracting bounds for neighboring indices, and also vice versa by summing up bounds, the resulting scaling factors will not match the quantity under consideration.

From a technical point of view, the approach taken in this work and the one by Blanchard et al. (2006) also differ considerably. While the analysis in the latter is carried out in abstract Hilbert spaces, in this work, the analysis is based in the finite dimensional domain, having the potential advantage that the arguments are somewhat more elementary. However, one could suspect that the absolute terms occurring in our bounds are an artifact of the more elementary approach (in particular

since these terms are a side-effect of the truncation of the kernel matrix). Then, a more abstract approach might be able to obtain fully relative bounds. Note however, that Ostrowski's inequality does not easily extend to the high-dimensional case, as the convergence of the error matrix $\mathbf{C}_n^r$ scales with the dimension of the finite-dimensional case. At any rate, these questions are interesting possible direction for future research.

9. Conclusion and Outlook

We have derived finite sample size bounds on the error between single eigenvalues of the kernel matrix and their asymptotic limits. These bounds scale with the eigenvalue under consideration leading to significantly more accurate bounds for single eigenvalues than previously known bounds in particular for small eigenvalues and small sample sizes. Also for fairly large sample sizes, the bounds can still be superior to existing bounds since the number of samples necessary to make existing non-scaling bounds competitive can be unrealistically large.

For future work, we would like to suggest three possibilities. (i) If additional information on the kernel, or the probability distribution is available, the bounds on the norms of the error matrices could be improved leading to more accurate bounds.

(ii) Note that the resulting bounds require the knowledge of the true eigenvalues. From a theoretical point of view, this approach is acceptable, because we were specifically interested in approximation errors of small eigenvalues, and this assumption is codified into the knowledge of the true eigenvalues. In practical situations, however, one might be interested in obtaining a confidence bound without knowledge of the eigenvalues. This means that one has to derive some property of the true eigenvalues, for example, their rate of decay. Statistical tests could be constructed to this means based on the bounds presented here. Then, the bounds presented in this work predict that the estimated eigenvalues decay at the same rate giving confidence bounds which scale at the correct rate.

(iii) Finally, since the basic perturbation bound also holds for non-random choices of points, the result could be applied in the analysis of the numerical approximation of integral equations. The norms of the error matrices would then be bounded using approximation theory.

Acknowledgments

A substantial part of this research was performed while the author was with the Computer Vision and Pattern Recognition group headed by Professor Joachim Buhmann (now ETH Zürich) at the University of Bonn (see also Braun, 2005). The author wishes to thank Stefan Harmeling, Gilles Blanchard, Joachim Buhmann, Michael Clausen, Motoaki Kawanabe, Tilman Lange, Klaus-Robert Müller, Peter Orbanz, and Axel Munk for fruitful discussion and helpful comments. The author would also like to thank the anonymous referees whose comments helped to improved the paper further, and who made the author aware of an error contained in an earlier version of the paper. In the course of fixing this error, the estimate of the absolute error term could be significantly improved. The author also acknowledges funding received on behalf of DFG grant #Buh 914/5, and BMBF grant 16SV2231.

Appendix A. Supplementary Results

In this section, we collect some supplementary results for reference, which are used in the main text.

A.1 Perturbation of Hermitian Matrices

We use two classical results on the perturbation of eigenvalues for Hermitian matrices.

Theorem A.1 (Weyl) *(Horn and Johnson, 1985, Theorem 4.3.1) Let* $\mathbf{A}, \mathbf{E}$ *be Hermitian* $n \times n$ *matrices. Then, for each* $1 \leq i \leq n$*,*

$$\lambda_i(\mathbf{A}) + \lambda_n(\mathbf{E}) \leq \lambda_i(\mathbf{A}+\mathbf{E}) \leq \lambda_i(\mathbf{A}) + \lambda_1(\mathbf{E}).$$

This implies that

$$|\lambda_i(\mathbf{A}) - \lambda_i(\mathbf{A}+\mathbf{E})| \leq \|\mathbf{E}\|.$$

Theorem A.2 (Ostrowski) *(Horn and Johnson, 1985, Theorem 4.5.9., Corollary 4.5.11) Let* $\mathbf{A}$ *be a Hermitian* $n \times n$ *matrix, and* $\mathbf{S}$ *a non-singular* $n \times n$ *matrix. Then, for* $1 \leq i \leq n$*, there exist non-negative real* θ_i *with* $\lambda_n(\mathbf{S}\mathbf{S}^*) \leq \theta_i \leq \lambda_1(\mathbf{S}\mathbf{S}^*)$ *such that*

$$\lambda_i(\mathbf{S}\mathbf{A}\mathbf{S}^*) = \theta_i \lambda_i(\mathbf{A}).$$

Consequently,

$$|\lambda_i(\mathbf{S}\mathbf{A}\mathbf{S}^*) - \lambda_i(\mathbf{A})| \leq |\lambda_i(\mathbf{A})| \|\mathbf{S}^*\mathbf{S} - \mathbf{I}\|.$$

For the case of non-square $\mathbf{S}$, the same result holds as can be shown by extending either $\mathbf{S}$ or $\mathbf{A}$ with zeros until both matrices are square and have the same size and by a continuity argument to extend Ostrowski's theorem to singular $\mathbf{S}$ (Horn and Johnson, 1985, p. 224).

Corollary A.3 *Ostrowski's theorem also holds if* $\mathbf{S}$ *is a (non-square)* $n \times m$ *matrix.*

A.2 Asymptotics of Infinite Sums

For convenience, we collect two elementary computations to estimate the asymptotic rates of tail sums of sequences with polynomial and exponential decay.

Theorem A.4 *For* $\alpha > 1$ *and* $\beta > 0$*,*

$$\sum_{i=r+1}^{\infty} i^{-\alpha} \leq \frac{r^{1-\alpha}}{\alpha - 1} = O(r^{1-\alpha}), \qquad \sum_{i=r+1}^{\infty} e^{-\beta i} = \frac{e^{-\beta(r+1)}}{1 - e^{-\beta}} = O(e^{-\beta r}).$$

To prove these two rates, note that

$$\sum_{i=r+1}^{\infty} i^{-\alpha} \leq \int_{r+1}^{\infty} (x-1)^{-\alpha} dx = \int_{r}^{\infty} x^{-\alpha} dx = \frac{x^{1-\alpha}}{1-\alpha}\bigg|_{r}^{\infty} = 0 - \frac{r^{1-\alpha}}{1-\alpha} = \frac{r^{1-\alpha}}{\alpha - 1}.$$

Furthermore, since $\sum_{i=r+1}^{\infty} (e^{-\beta})^i$ is the tail of a geometric series,

$$\sum_{i=r+1}^{\infty} e^{-\beta i} \leq \frac{1}{1-e^{-\beta}} - \frac{1 - \left(e^{-\beta}\right)^{r+1}}{1-e^{-\beta}} = \frac{\left(e^{-\beta}\right)^{r+1}}{1-e^{-\beta}}.$$

References

Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing*, chapter 22, "Legendre Functions", and chapter 8, "Orthogonal Polynomials", pages 331–339, 771–802. Dover, New York, 1972.

Gilles Blanchard, Olivier Bousquet, and Laurent Zwald. Statistical properties of kernel principal component analysis. *Machine Learning*, 2006. (to appear, published online March 30, 2006).

Mikio L. Braun. *Spectral Properties of the Kernel Matrix and their Relation to Kernel Methods in Machine Learning*. PhD thesis, University of Bonn, Germany, 2005. Available electronically at http://hss.ulb.uni-bonn.de/diss_online/math_nat_fak/2005/braun_mikio.

J. Dauxois, A. Pousse, and Y. Romain. Asymptotic theory for the principal component analysis of a vector random function: Some applications to statistical inference. *Journal of Multivariate Analysis*, 12:136–154, 1982.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

Vladimir Koltchinskii and Evarist Giné. Random matrix approximation of spectra of integral operators. *Bernoulli*, 6(1):113–167, 2000.

Sebastian Mika. *Kernel Fisher Discriminants*. PhD thesis, Technische Universität Berlin, December 2002.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear compoment analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

John Shawe-Taylor, Christopher K. I. Williams, Nello Christianini, and Jaz Kandola. On the eigenspectrum of the gram matrix and the generalization error of kernel-PCA. *IEEE Transactions on Information Theory*, 51(7):2510–2522, July 2005.

Aad van der Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes*. Springer-Verlag, 1996.

Ulrike von Luxburg. *Statistical Learning with Similarity and Dissimilarity Functions*. PhD thesis, Technische Universität Berlin, November 2004.

Journal of Machine Learning Research 7 (2006) 2329-2367 Submitted 6/05; Revised 5/06; Published 11/06

Point-Based Value Iteration for Continuous POMDPs

Josep M. Porta PORTA@IRI.UPC.EDU
Institut de Robòtica i Informàtica Industrial
UPC-CSIC
Llorens i Artigas 4-6, 08028, Barcelona, Spain

Nikos Vlassis VLASSIS@SCIENCE.UVA.NL
Matthijs T.J. Spaan MTJSPAAN@SCIENCE.UVA.NL
Informatics Institute
University of Amsterdam
Kruislaan 403, 1098SJ, Amsterdam, The Netherlands

Pascal Poupart PPOUPART@CS.UWATERLOO.CA
David R. Cheriton School of Computer Science
University of Waterloo
200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1

Editors: Sven Koenig, Shie Mannor, and Georgios Theocharous

Abstract

We propose a novel approach to optimize Partially Observable Markov Decisions Processes (POMDPs) defined on continuous spaces. To date, most algorithms for model-based POMDPs are restricted to discrete states, actions, and observations, but many real-world problems such as, for instance, robot navigation, are naturally defined on continuous spaces. In this work, we demonstrate that the value function for continuous POMDPs is convex in the beliefs over continuous state spaces, and piecewise-linear convex for the particular case of discrete observations and actions but still continuous states. We also demonstrate that continuous Bellman backups are contracting and isotonic ensuring the monotonic convergence of value-iteration algorithms. Relying on those properties, we extend the PERSEUS algorithm, originally developed for discrete POMDPs, to work in continuous state spaces by representing the observation, transition, and reward models using Gaussian mixtures, and the beliefs using Gaussian mixtures or particle sets. With these representations, the integrals that appear in the Bellman backup can be computed in closed form and, therefore, the algorithm is computationally feasible. Finally, we further extend PERSEUS to deal with continuous action and observation sets by designing effective sampling approaches.

Keywords: planning under uncertainty, partially observable Markov decision processes, continuous state space, continuous action space, continuous observation space, point-based value iteration

1. Introduction

Automated systems can be viewed as taking inputs from the environment in the form of sensor measurements and producing outputs toward the realization of some goals. An important problem is the design of good control policies that produce suitable outputs (e.g., actions) based on the inputs received (e.g., observations). When the state of the environment is only partially observable through noisy measurements, and actions have stochastic effects, optimizing the course of action is a non-trivial task. Partially Observable Markov Decision Processes, POMDPs (Åström, 1965;

Dynkin, 1965) provide a principled framework to formalize and optimize control problems fraught with uncertainty. Such problems arise in a wide range of application domains including assistive technologies (Montemerlo et al., 2002; Boger et al., 2005), mobile robotics (Simmons and Koenig, 1995; Cassandra et al., 1996; Theocharous and Mahadevan, 2002; Pineau et al., 2003b), preference elicitation (Boutilier, 2002), spoken-dialog systems (Roy et al., 2000; Zhang et al., 2001; Williams et al., 2005), and gesture recognition (Darrell and Pentland, 1996).

Policy optimization (i.e., optimization of the course of action) can be done with or without a model of the environment dynamics. Model-free techniques such as neuro-dynamic programming (Bertsekas and Tsitsiklis, 1996), and stochastic gradient descent (Meuleau et al., 1999; Ng and Jordan, 2000; Baxter and Bartlett, 2001; Aberdeen and Baxter, 2002) directly optimize a policy by simulation. These approaches are quite versatile since there is no explicit modeling of the environment. On the other hand, the absence of explicit modeling information is compensated by simulation which may take an unbearable amount of time. In practice, the amount of simulation can be reduced by restricting the search for a good policy to a small class.

In contrast, model-based approaches assume knowledge about a transition model encoding the effects of actions on environment states, an observation model defining the correlations between environment states and sensor observations, and a reward model encoding the utility of environment states. Even when sufficient a priori knowledge is available to encode a complete model, policy optimization remains a hard task that depends heavily on the nature of the model. To date, most existing algorithms for model-based POMDPs assume discrete states, actions and observations. Even then, optimization is generally intractable (Papadimitriou and Tsitsiklis, 1987; Madani et al., 1999; Lusena et al., 2001) and one must resort to the exploitation of model-specific structural properties to obtain approximate scalable algorithms for POMDPs with large state spaces (Boutilier and Poole, 1996; Roy and Gordon, 2003; Poupart and Boutilier, 2003, 2005) and complex policy spaces (Pineau et al., 2003a; Spaan and Vlassis, 2005; Smith and Simmons, 2004; Poupart and Boutilier, 2004, 2005).

Many real-world POMDPs are naturally modeled by continuous states, actions and observations. For instance, in a robot navigation task, the state space may correspond to robot poses (x, y, θ), the observations may correspond to distances to obstacles measured by sonars or laser range finders, and actions may correspond to velocity and angular controls. Given the numerous optimization techniques for discrete models, a common approach for continuous models consists of discretizing or approximating the continuous components with a grid (Thrun, 2000; Roy et al., 2005). This usually leads to an important tradeoff between complexity and accuracy as we vary the coarseness of the discretization. More precisely, as we refine a discretization, computational complexity increases. Clearly, an important research direction is to consider POMDP solution techniques that operate directly in continuous domains, which would render the discretization of the continuous components superfluous.

Duff (2002) considered a special case of continuous POMDPs in the context of model-based Bayesian reinforcement learning, in which beliefs are maintained over the space of unknown parameters of the transition model of a discrete-state MDP. As those parameters are probabilities, the corresponding POMDP is defined over a continuous domain. Duff (2002) demonstrated that, for this special case, the optimal value function of the POMDP is parameterized by a set of functions, and for finite horizon it is piecewise-linear and convex (PWLC) over the belief space of multinomial distributions. Independently, Porta et al. (2005) considered the case of robot planning under uncertainty, modeled as a continuous POMDP over the pose (continuous coordinates) of the robot.

They also proved that the optimal value function is parameterized by an appropriate set of functions called α-functions (in analogy to the α-vectors in discrete POMDPs). Moreover, they demonstrated that the value function for finite horizon is PWLC over the robot belief space for any functional form of the beliefs. In addition, Porta et al. (2005) provided an analytical derivation of the α-functions as linear combinations of Gaussians when the transition, reward, and observation models of the POMDP are also Gaussian-based.

In this paper, we generalize the results of Duff (2002) and Porta et al. (2005), and describe a framework for optimizing model-based POMDPs in which the state space and/or action and observation spaces are continuous. We first concentrate on the theoretical basis on which to develop a sound value-iteration algorithm for POMDPs on continuous spaces. We demonstrate that the value function for arbitrary continuous POMDPs is in general convex, and it is PWLC in the particular case when the states are continuous but the actions and observations are discrete. We also demonstrate that Bellman backups for continuous POMDPs are contracting and isotonic, which guarantees the monotonic convergence of a value-iteration algorithm.

Functions defined over continuous spaces (e.g., beliefs, observation, action and reward models) can have arbitrary forms that may not be parameterizable. In order to design feasible algorithms for continuous POMDPs, it is crucial to work with classes of functions that have simple parameterizations and that yield to closed belief updates and Bellman backups. We investigate Gaussian mixtures and particle-based representations for the beliefs and linear combinations of Gaussians for the models. Using these representations, we extend the PERSEUS algorithm (Spaan and Vlassis, 2005) to solve POMDPs with continuous states but discrete actions and observations. We also show that POMDPs with continuous states, actions, and observations can be reduced to POMDPs with continuous states, discrete actions, and discrete observations using sampling strategies. As such, we extend PERSEUS to handle general continuous POMDPs.

The rest of the paper is structured as follows. Section 2 introduces POMDPs. Section 3 includes the proofs of some basic properties that are used to provide sound ground to the value-iteration algorithm for continuous POMDPs. Section 4 reviews the point-based POMDP solver PERSEUS. Section 5 investigates POMDPs with Gaussian-based models and particle-based representations for belief states, as well as their use in PERSEUS. Section 6 addresses the extension of PERSEUS to deal with continuous action and observation spaces. Section 7 presents some experiments showcasing the extended PERSEUS algorithm on a simulated robot navigation task. Section 8 gives an overview of related work on planning for continuous POMDPs. Finally, Section 9 concludes and highlights some possibilities for future work.

2. Preliminaries: MDPs and POMDPs

The Markov Decision Process (MDP) framework is a well-known planning paradigm that can be applied whenever we have an agent making decisions in a system described by

- a set of system states, S,
- a set of actions available to the agent, A,
- a transition model defined by $p(s'|s,a)$, the probability that the system changes from state s to s' when the agent executes action a, and

- a reward function defined as $r_a(s) \in \mathbb{R}$, the reward obtained by the agent if it executes action a when the system is in state s.

The dynamics of a discrete-time MDP is the following: at a given moment, the system is in a state s and the agent executes an action a. As a result, the agent receives a reward r and the system state changes to s'. The state contains enough information to allow to plan optimally, and thus a *policy* is a mapping from states to actions. To assess the quality of a given policy, π, the *value function* condenses the immediate and delayed reward that can be obtained from a given state s_0

$$V^{\pi}(s_0) = E\Big[\sum_{t=0}^{n} \gamma^t r_{\pi(s_t)}(s_t)\Big],$$

where the state evolves according to the transition model $p(s_{t+1}|s_t, \pi(s_t))$, n is the planning horizon (possibly infinite), and $\gamma \in [0,1)$ is a discount factor that trades off the importance of the immediate and the delayed reward.

The objective of MDP-based planning is to determine an *optimal policy*, π^*, that is, a policy that assigns to each state the action from which the most future reward can be expected. In the literature, there are several algorithms for computing an optimal policy for any MDP. When the transition and the reward model are known in advance, we can use planning algorithms mainly developed within the *operations research* field. Algorithms also exist for the case where the transition and reward models must be learned by the agent as it interacts with the environment. These learning algorithms are typically developed within the *reinforcement learning* community (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998).

Three popular planning algorithms are *value iteration*, *policy iteration*, and *linear programming*. We will focus on the first one, value iteration. This algorithm computes a sequence of value functions departing from an initial value function V_0 and using the following recursion

$$V_n(s) = \max_{a \in A} Q_n(s,a),$$

with

$$Q_n(s,a) = r_a(s) + \gamma \sum_{s' \in S} p(s'|s,a)\, V_{n-1}(s').$$

The above recursion is usually written in functional form

$$Q_n^a = H^a V_{n-1},$$

$$V_n = H V_{n-1}, \tag{1}$$

and it is known as the *Bellman recursion* (Bellman, 1957). This recursion converges to V^*, from which we can define an optimal policy π^* as

$$\pi^*(s) = \arg\max_a Q^*(s,a).$$

For each value function, V_n, we can readily derive an approximation to the optimal policy. Bounds on the quality of this approximation are given by Puterman (1994) in Theorem 6.3.1.

The MDP framework assumes the agent has direct knowledge of the system state. In many realistic situations, however, the agent can not directly access the state, but it receives an *observation*

that stochastically depends on it. In these cases, the system can be modeled as a Partially Observable Markov Decision Process (POMDP). This paradigm extends the MDP framework by incorporating a set of observations O, and an observation model defined by $p(o|s)$, the probability that the agent observes o when the system reaches state s. In a POMDP, the agent typically needs to infer the state of the system from the sequence of received observations and executed actions. A usual representation for the knowledge about the system state is a *belief*, that is, a probability distribution over the state space. The initial belief is assumed to be known and, if b is the belief of the agent about the state, the updated belief after executing action a and observing o is

$$b^{a,o}(s') = \frac{p(o|s')}{p(o|b,a)}\, p(s'|b,a), \tag{2}$$

with $p(s'|b,a)$ the propagation of the belief b through the transition model. For a continuous set of states S, this propagation is defined as

$$p(s'|b,a) = \int_{s\in S} p(s'|s,a)\, b(s)\, ds, \tag{3}$$

and, for a discrete set S, the integral is replaced by a sum. Under the Markov assumption, the belief carries enough information to plan optimally (see Bertsekas, 2001). Thus, a belief-based discrete-state POMDP can be seen as an MDP with a continuous state space that has one dimension per state. In the case of continuous-state POMDPs, the corresponding belief space is also continuous, but with an infinite number of dimensions since there are infinitely many physical states. This additional complexity is one of the reasons why most of the POMDP research focuses on the discrete-state case.

The belief-state MDP defined from a POMDP has a transition model

$$p(b'|b,a) = \begin{cases} p(o|b,a) & \text{if } b' = b^{a,o}, \\ 0 & \text{otherwise,} \end{cases}$$

and its policy and value function are defined on the space of beliefs. The Bellman recursion is defined as

$$V_n(b) = \sup_{a\in A} \Big\{ \langle r_a, b\rangle + \gamma \int_o p(o|b,a)\, V_{n-1}(b^{a,o})\, do \Big\}. \tag{4}$$

For discrete observation and action spaces, the integral over the observation space is replaced by a sum and the sup over the action space by a max operator. In Eq. 4, the $\langle f, b\rangle$ operation is used to express the expectation of the function f in the probability space defined by sample space S, the σ-algebra on S, and the probability distribution b. For continuous-state POMDPs, this operator is computed with an integral over S

$$\langle f, b\rangle = \int_{s\in S} f(s)\, b(s)\, ds,$$

and for discrete-state POMDPs, it corresponds to the inner product

$$\langle f, b\rangle = \sum_{s\in S} f(s)\, b(s).$$

Note that, in both cases and for a fixed f, the expectation operator is a linear function in the belief space since we have

$$\langle f, k\,b\rangle = k\,\langle f, b\rangle,$$
$$\langle f, b + b'\rangle = \langle f, b\rangle + \langle f, b'\rangle,$$

for any k independent of the integration/sum variable.

At first sight computing the POMDP value function seems intractable, but Sondik (1971) has shown that, for discrete POMDP, this function can be expressed in a simple form

$$V_n(b) = \arg\max_{\{\alpha_n^i\}_i} \langle \alpha_n^i, b\rangle,$$

with $\{\alpha_n^i\}_i$ a set of vectors. Each α-vector is generated for a particular action, and the optimal action with planning horizon n for a given belief is the action associated with the α-vector that defines V_n for that belief. Thus, the set of α_n^i vectors encodes not only the value, but also the optimal policy.

Since the $\langle \cdot, \cdot \rangle$ function is linear, the value function V_n computed as a maximum of a set of such expectations is piecewise-linear convex (PWLC) in the belief space. Using this formulation, value iteration algorithms for discrete state POMDPs typically focus on the computation of the α_n-vectors. Two basic strategies for POMDP value iteration are found in the literature. In the first one, the initial value function (i.e., at planning horizon 0) is a set of α-vectors directly defined from the reward function (Sondik, 1971; Monahan, 1982; Cheng, 1988; Kaelbling et al., 1998; Cassandra et al., 1997; Pineau et al., 2003a). In the second strategy, the initial value function is a single α-vector that lower bounds the value function for any possible planning horizon (Zhang and Zhang, 2001; Spaan and Vlassis, 2005). In both cases, exact value iteration converges to the same fixed point, but the second strategy may be more effective in approximate value iteration schemes.

3. Properties of Continuous POMDPs

In the previous section, we saw that algorithms for discrete POMDPs rely on a representation of the value function as a PWLC function based on a discrete set of supporting vectors. In this section, we show that this representation can be generalized to continuous-state POMDPs, while still assuming a discrete set of actions and observations. In Section 6, we discuss how to tackle POMDPs with continuous actions and observations via sampling.

First, we prove that the value function for a continuous POMDP is convex and, next, that it is PWLC for the case of continuous states, but discrete observations and actions. In this last case, the value function can be represented as a set of α-functions that play the same role as α-vectors in a discrete POMDP. We also prove that the continuous POMDP value-function recursion is an isotonic contraction. From these results, it follows that this recursion converges to a single fixed point corresponding to the optimal value function V^*. The theoretical results presented in this section establish that there is in principle no barrier in defining value iteration algorithms for continuous POMDPs.

3.1 The Optimal Value Function for Continuous POMDPs is Convex

To prove that the optimal value function for continuous POMDPs is convex, we first prove the following lemma.

Lemma 1 *The n-step optimal value function V_n in a continuous POMDP can be expressed as*

$$V_n(b) = \sup_{\{\alpha_n^i\}_i} \langle \alpha_n^i, b \rangle,$$

for appropriate continuous set of α*-functions* $\alpha_n^i : S \to \mathbb{R}$.

Proof The proof, as in the discrete case, is done via induction. In the following we assume that all operations (e.g., integrals) are well-defined in the corresponding spaces. For planning horizon 0, we only have to take into account the immediate reward and, thus, we have that

$$V_0(b) = \sup_{a \in A} \langle r_a, b \rangle,$$

and, therefore, if we define the continuous set

$$\{\alpha_0^i\}_i = \{r_a\}_{a \in A}, \tag{5}$$

we have that, as desired

$$V_0(b) = \sup_{\{\alpha_0^i\}_i} \langle \alpha_0^i, b \rangle.$$

For the general case, we have that, using Eq. 4

$$V_n(b) = \sup_{a \in A} \left\{ \langle r_a, b \rangle + \gamma \int_o p(o|b,a)\, V_{n-1}(b^{a,o})\, do \right\}$$

and, by the induction hypothesis,

$$V_{n-1}(b^{a,o}) = \sup_{\{\alpha_{n-1}^j\}_j} \langle \alpha_{n-1}^j, b^{a,o} \rangle.$$

From Eq. 2 and the definition of the $\langle \cdot, \cdot \rangle$ expectation operator,

$$V_{n-1}(b^{a,o}) = \frac{1}{p(o|b,a)} \sup_{\{\alpha_{n-1}^j\}_j} \int_{s'} \alpha_{n-1}^j(s')\, p(o|s')\, p(s'|b,a)\, ds'.$$

With the above

$$\begin{aligned}
V_n(b) &= \sup_{a \in A} \left\{ \langle r_a, b \rangle + \gamma \int_o \sup_{\{\alpha_{n-1}^j\}_j} \int_{s'} \alpha_{n-1}^j(s')\, p(o|s')\, p(s'|b,a)\, ds'\, do \right\} \\
&= \sup_{a \in A} \left\{ \langle r_a, b \rangle + \gamma \int_o \sup_{\{\alpha_{n-1}^j\}_j} \int_{s'} \alpha_{n-1}^j(s')\, p(o|s') \left[\int_s p(s'|s,a)\, b(s)\, ds \right] ds'\, do \right\} \\
&= \sup_{a \in A} \left\{ \langle r_a, b \rangle + \gamma \int_o \sup_{\{\alpha_{n-1}^j\}_j} \int_s \left[\int_{s'} \alpha_{n-1}^j(s')\, p(o|s')\, p(s'|s,a)\, ds' \right] b(s)\, ds\, do \right\} \\
&= \sup_{a \in A} \left\{ \langle r_a, b \rangle + \gamma \int_o \sup_{\{\alpha_{n-1}^j\}_j} \left\langle \int_{s'} \alpha_{n-1}^j(s')\, p(o|s')\, p(s'|s,a)\, ds', b \right\rangle do \right\}.
\end{aligned}$$

At this point, we can define

$$\alpha^j_{a,o}(s) = \int_{s'} \alpha^j_{n-1}(s')\, p(o|s')\, p(s'|s,a)\, ds'. \tag{6}$$

Note that these functions are independent of the belief point b for which we are computing V_n. With this, we have that

$$V_n(b) = \sup_{a\in A} \Big\{ \langle r_a, b\rangle + \gamma \int_o \sup_{\{\alpha^j_{a,o}\}_j} \langle \alpha^j_{a,o}, b\rangle \, do \Big\},$$

and we define

$$\alpha_{a,o,b} = \arg\sup_{\{\alpha^j_{a,o}\}_j} \langle \alpha^j_{a,o}, b\rangle. \tag{7}$$

The $\alpha_{a,o,b}$ set is just a subset of the $\alpha^j_{a,o}$ set defined above. Using this subset, we can write

$$\begin{aligned} V_n(b) &= \sup_{a\in A} \Big\{ \langle r_a, b\rangle + \gamma \int_o \langle \alpha_{a,o,b}, b\rangle \, do \Big\} \\ &= \sup_{a\in A} \Big\langle r_a + \gamma \int_o \alpha_{a,o,b}\, do, b \Big\rangle. \end{aligned}$$

Now

$$\{\alpha^i_n\}_i = \bigcup_{\forall b} \{r_a + \gamma \int_o \alpha_{a,o,b}\, do\}_{a\in A}, \tag{8}$$

is a continuous set of functions parameterized in the continuous action set. Intuitively, each α_n-function corresponds to a plan and, the action associated with a given α_n-function is the optimal action for planning horizon n for all beliefs that have such function as the maximizing one.

With the above definition, we have that V_n can be put in the desired form

$$V_n(b) = \sup_{\{\alpha^i_n\}_i} \langle \alpha^i_n, b\rangle, \tag{9}$$

and, thus, the lemma holds. ■

Using the above lemma we can directly prove the convex property for the value function on continuous POMDPs. Recall that, as mentioned in Section 2, for a fixed α^i_n-function the $\langle \alpha^i_n, b\rangle$ operator is *linear* in the belief space. Therefore, the convex property is given by the fact that V_n is defined as the supreme of a set of convex (linear) functions and, thus, we obtain a convex function as a result. The optimal value function, V^* is the limit for V_n as n goes to infinite and, since all V_n are convex functions so is V^*.

Lemma 2 *When the state space is continuous but the observation and action sets are discrete, the finite horizon value function is piecewise-linear convex (PWLC).*

Proof First, we have to prove that the $\{\alpha^i_n\}_i$ sets are discrete for all n. Again, we can proceed via induction. For discrete actions, $\{\alpha^i_0\}_i$ is discrete from its definition (see Eq. 5). For the general case, we have to observe that, for discrete actions and observations and assuming $M = |\{\alpha^j_{n-1}\}|$, the sets $\{\alpha^j_{a,o}\}$ are discrete: for a given action a and observation o we can generate at most M

$\alpha^j_{a,o}$-functions. Now, using a reasoning parallel to that of the enumeration phase of the Monahan's algorithm (Monahan, 1982), we have at most $|A|\,M^{|O|}$ different α^i_n-functions (fixing the action, we can select one of the M $\alpha^j_{a,o}$-functions for each one of the observations) and, thus, $\{\alpha^i_n\}_i$ is a discrete set.

From the previous lemma, we know the value function to be convex. The *piecewise-linear* part of the property is given by the fact that, as we have just seen, the $\{\alpha^i_n\}_i$ set is of finite cardinality and, therefore, V_n is defined as a finite set of linear functions. ■

When the state space is discrete, the α-functions become α-vectors and the above proof is equivalent to the classical PWLC demonstration first provided by Sondik (1971).

3.2 The Continuous POMDP Bellman Recursion is a Contraction

Lemma 3 *For the continuous POMDP value recursion H and two given value functions V and U, it holds that*

$$\|HV - HU\| \leq \beta \|V - U\|,$$

with $0 \leq \beta < 1$ and $\|\cdot\|$ the supreme norm. That is, the continuous POMDP value recursion H is a contractive mapping.

Proof The H mapping can be seen as

$$HV(b) = \max_a H^a V(b),$$

with

$$H^a V(b) = \langle r_a, b\rangle + \gamma \int_o p(o|b,a)\, V(b^{a,o})\, do.$$

Assume that $\|HV - HU\|$ is maximum at point b. Denote as a_1 the optimal action for HV at b and as a_2 the optimal one for HU

$$HV(b) = H^{a_1} V(b),$$
$$HU(b) = H^{a_2} U(b).$$

Then it holds

$$\|HV(b) - HU(b)\| = H^{a_1} V(b) - H^{a_2} U(b),$$

assuming, without loss of generality that $HV(b) \leq HU(b)$. Since a_1 is the action that maximizes HV at b we have that

$$H^{a_2} V(b) \leq H^{a_1} V(b).$$

Therefore, we have that

$$\begin{gathered}
\|HV - HU\| = \\
\|HV(b) - HU(b)\| = \\
H^{a_1} V(b) - H^{a_2} U(b) \leq \\
H^{a_2} V(b) - H^{a_2} U(b) = \\
\gamma \textstyle\int_o p(o|a_2,b)\, [V(b^{a_2,o}) - U(b^{a_2,o})]\, do \leq \\
\gamma \textstyle\int_o p(o|a_2,b)\, \|V - U\|\, do = \\
\gamma \|V - U\|.
\end{gathered}$$

Since γ is in $[0,1)$, the lemma holds. ■

The space of value functions define a *vector space* (i.e., a space closed under addition and scalar scaling) and the contraction property ensures this space to be *complete* (i.e., all Cauchy sequences have a limit in this space). Therefore, the space of value functions together with the supreme norm form a *Banach space* and the *Banach fixed-point theorem* ensures (a) the existence of a single fixed point, and (b) that the value recursion always converges to this fixed point (see Puterman, 1994, Theorem 6.2.3 for more details).

3.3 The Continuous POMDP Bellman Recursion is Isotonic

Lemma 4 *For any two value functions V and U, we have that*

$$V \leq U \Rightarrow HV \leq HU$$

that is, the continuous POMDP value recursion H is an isotonic mapping.

Proof Let us denote as a_1 the action that maximizes HV at point b and a_2 the action that does so for HU

$$HV(b) = H^{a_1}V(b),$$
$$HU(b) = H^{a_2}U(b).$$

By definition, the value for action a_1 for HU at b is lower (or equal) than that for a_2, that is

$$H^{a_1}U(b) \leq H^{a_2}U(b).$$

From a given b we can compute $b^{a_1,o}$, for an arbitrary o and, then, the following holds

$$\begin{aligned}
&V \leq U \Rightarrow \\
&\forall b,o,\ V(b^{a_1,o}) \leq U(b^{a_1,o}) \Rightarrow \\
&\int_o p(o|a_1,b)\,V(b^{a_1,o})\,do \leq \int_o p(o|a_1,b)\,U(b^{a_1,o})\,do \Rightarrow \\
&\langle r_{a_1},b\rangle + \gamma \int_o p(o|a_1,b)\,V(b^{a_1,o})\,do \leq \langle r_{a_1},b\rangle + \gamma \int_o p(o|a_1,b)\,U(b^{a_1,o})\,do \Rightarrow \\
&H^{a_1}V(b) \leq H^{a_1}U(b) \Rightarrow \\
&H^{a_1}V(b) \leq H^{a_2}U(b) \Rightarrow \\
&HV(b) \leq HU(b) \Rightarrow \\
&HV \leq HU.
\end{aligned}$$

Since b and, from it $b^{a_1,o}$, can be chosen arbitrarily, the value function is isotonic. ■

The isotonic property of the value recursion ensures that value iteration converges *monotonically*.

4. PERSEUS: A Point-Based POMDP Solver

Eqs. 6 to 8 constitute the value-iteration process for continuous POMDPs since they provide a constructive way to define the α-elements (α-functions for the continuous-state case and α-vectors for the discrete one) defining V_n from those defining V_{n-1}. The implementation of this value iteration, however, will be only computationally feasible if all the involved integrals can be either derived in closed form or approximated numerically. Moreover, the $\{\alpha_n^i\}_i$ are continuous sets and this makes the actual implementation of the described value-iteration process challenging. In this section, we concentrate on continuous-state POMDPs (POMDPs with continuous states, but discrete actions and observations). In this case, the $\{\alpha_n^i\}_i$ sets contain finitely many elements and the value function is PWLC. This allows us to adapt POMDP solving algorithms designed for the discrete case. In particular, we describe the point-based value-iteration algorithm PERSEUS (Spaan and Vlassis, 2005) which has been shown to be very efficient for discrete POMDPs. The description shown in Table 1 is generic so that it can be used for either discrete or continuous-state POMDPs. Extensions of PERSEUS to deal with continuous action and observation spaces are detailed in Section 6.

The computation of the mapping H (Eq. 1) for a given belief point b is called a *backup*. This mapping determines the α-element (α-function for continuous POMDPs and α-vector for discrete-state POMDPs) to be included in V_n for a belief point under consideration (see Eqs. 6 to 8). A full backup, that is, a backup for the whole belief space, involves the computation of all relevant α-elements for V_n. Full backups are computationally expensive (they involve an exponentially growing set of α-elements), but the backup for a single belief point is relatively cheap. This is exploited by recent point-based POMDP algorithms to efficiently approximate V_n on a fixed set of belief points (Pineau et al., 2003a; Spaan and Vlassis, 2005). The α-elements for this restricted set of belief points generalize over the whole belief space and, thus, they can be used to approximate the value function for any belief point.

The backup for a given belief point b is

$$\texttt{backup}(b) = \arg\max_{\{\alpha_n^i\}_i} \langle \alpha_n^i, b \rangle,$$

where $\alpha_n^i(s)$ is defined in Eqs. 7 and 8 from the $\alpha_{a,o}$-elements (Eq. 6). Using the backup operator, the value of V_n at b (Eq. 9) is simply

$$V_n(b) = \langle \texttt{backup}(b), b \rangle.$$

If this point-backup has to be computed for many belief points, the process can be speeded up by computing the set $\{\alpha_{a,o}^j\}_j$ for all actions, observations, and elements in V_{n-1} (see Eq. 6) since these α-elements are independent of the belief point and are the base components to define the $\alpha_{a,o,b}$ for any particular belief point, b.

Using this backup operator, PERSEUS is defined as follows. First (Table 1, line 2), we let the agent randomly explore the environment and collect a set B of reachable belief points. Next (Table 1, lines 3-5), we initialize the value function V_0 as a constant function over the state space. The value for V_0 is the minimum possible accumulated discounted reward, $\min\{R\}/(1-\gamma)$ with R the set of possible rewards. In line 3, u denotes a function on S so that

$$\langle u, b \rangle = 1,$$

for any possible belief, b and, in particular, for the beliefs in B. The exact form for u depends on the representation we use for the α-elements. For instance, for a discrete set of states, u is a constant

Perseus
Input: A POMDP.
Output: V_n, an approximation to the optimal value function V^*.

1: **Initialize**
2: $B \leftarrow$ A set of randomly sampled belief points.
3: $\alpha \leftarrow \frac{\min\{R\}}{1-\gamma}\, u$
4: $n \leftarrow 0$
5: $V_n \leftarrow \{\alpha\}$
6: **do**
7: $\forall b \in B$,
8: $\texttt{Element}_n(b) \leftarrow \arg\max_{\alpha \in V_n} \langle \alpha, b \rangle$
9: $\texttt{Value}_n(b) \leftarrow \langle \texttt{Element}_n(b), b \rangle$
10: $V_{n+1} \leftarrow \emptyset$
11: $\tilde{B} \leftarrow B$
12: **do**
13: $b \leftarrow$ Point sampled randomly from $\tilde{B}$.
14: $\alpha \leftarrow \texttt{backup}(b)$
15: **if** $\langle \alpha, b \rangle < \texttt{Value}_\texttt{n}(\texttt{b})$
16: $\alpha \leftarrow \texttt{Element}_n(b)$
17: **endif**
18: $\tilde{B} \leftarrow \tilde{B} \setminus \{b' \in \tilde{B} \,|\, \langle \alpha, b' \rangle \geq \texttt{Value}_n(b')\}$
19: $V_{n+1} \leftarrow V_{n+1} \cup \{\alpha\}$
20: **until** $\tilde{B} = \emptyset$
21: $n \leftarrow n+1$
22: **until** convergence

Table 1: The PERSEUS algorithm: a point-based value-iteration algorithm for planning in POMDPs.

vector of $|S|$ ones, and for a continuous state space, u can be approximated by a properly scaled Gaussian with a large covariance in all the dimensions of the state space.

Starting with V_0, PERSEUS performs a number of approximate value-function update stages. The definition of the value-update process can be seen on lines 10–20 in Table 1, where $\tilde{B}$ is a set of non-improved points: points for which $V_{n+1}(b)$ is still lower than $V_n(b)$. At the start of each update stage, V_{n+1} is set to $\emptyset$ and $\tilde{B}$ is initialized to B. As long as $\tilde{B}$ is not empty, we sample a point b from $\tilde{B}$ and compute the new α-elements associated with this point using the `backup` operator. If this α-element improves the value of b, that is, if $\langle \alpha, b \rangle \geq V_n(b)$, we add α to V_{n+1}. The hope is that α improves the value of many other points, and all these points are removed from $\tilde{B}$. Often, a small number of α-elements will be sufficient to improve $V_n(b)$ $\forall b \in B$, especially in the first steps of value iteration. As long as $\tilde{B}$ is not empty we continue sampling belief points from it and trying to add their α-elements to V_{n+1}.

If the α computed by the `backup` operator does not improve at least the value of b (i.e., $\langle \alpha, b \rangle < V_n(b)$, see lines 15–17 in Table 1), we ignore α and insert a copy of the maximizing element of b

from V_n in V_{n+1}. Point b is now considered improved and is removed from $\tilde{B}$, together with any other belief points that had the same function as maximizing one in V_n. This procedure ensures that $\tilde{B}$ shrinks at each iteration and that the value update stage terminates.

PERSEUS stops when a given convergence criterion holds. This criterion can be based on the stability of the value function, on the stability of the associated policy, or simply on a maximum number of iterations or maximum planning time.

5. Representations for Continuous-State POMDPs

PERSEUS can be used with different representations for the beliefs, the α-functions, and the transition, observation and reward models. The selected representations should fulfill three minimum requirements. First, the belief update has to be closed, that is, the representation used for the beliefs must be closed under the propagation through the transition model (Eq. 3) and the multiplication with the observation model (Eq. 2). The second requirement is that the representation for the α-functions must be closed under addition and scaling (to compute Eq. 8), and closed for the integration after the product with the observation and the action models (see Eq. 6). Finally, the third requirement is that the $\langle \alpha, b \rangle$ expectation operator must be computable.

For discrete-state POMDP, the belief and the α-functions are represented by vectors and the models by matrices. In this case, all operations are linear algebra that produce closed form results. Next, we describe two alternative representations for continuous-state POMDPs. The first one uses linear combinations of Gaussian distributions to represent α-functions and mixtures of Gaussian distributions to represent belief states. The second one also uses linear combinations of Gaussian distributions to represent α-functions, but uses sets of particles to represent beliefs.

5.1 Models for Continuous-State POMDPs

For POMDPs with continuous states and discrete observations, a natural observation model $p(o|s)$ would consist of a continuum of multinomial distributions over o (i.e., one multinomial for each s). Unfortunately, such an observation model will not keep α-functions in closed form when multiplied by the observation model in a Bellman backup. Instead, we consider observation models such that $p(o|s)$ is approximated by a mixture of Gaussians in s for a given observation o.

We define the observation model $p(o|s)$ indirectly by specifying $p(s|o)$. More specifically, for a fixed observation o, we assume that $p(s|o)$ is a mixture of Gaussians on the state space defined non-parametrically from a set of samples $T = \{(s_i, o_i) \,|\, i \in [1, N]\}$ with o_i an observation obtained at state s_i. The training set can be obtained in a supervised way (Vlassis et al., 2002) or by autonomous interaction with the environment (Porta and Kröse, 2004). The observation model is

$$p(o|s) = \frac{p(s|o)\, p(o)}{p(s)},$$

and, assuming a uniform $p(s)$ in the space covered by T, and approximating $p(o)$ from the samples in the training set we have

$$p(o|s) \propto \left[\frac{1}{N_o} \sum_{i=1}^{N_o} \lambda_i^o \, \phi(s|s_i^o, \Sigma_i^o)\right] \frac{N_o}{N} = \sum_{i=1}^{N_o} w_i^o \, \phi(s|s_i^o, \Sigma_i^o),$$

where s_i^o is one of the N_o points in T with o as an associated observation, ϕ is a Gaussian with mean s_i^o and covariance matrix Σ_i^o, and $w_i^o = \lambda_i^o / N$ is a weighting factor associated with that training point.

The sets $\{\lambda_i^o\}_i$ and $\{\Sigma_i^o\}_i$ should be defined so that

$$\sum_{i=1}^{N_o} \lambda_i^o = N_o,$$
$$\lambda_i^o \geq 0,$$

and so that

$$p(s) = \sum_o p(s|o)\, p(o) = \sum_o \sum_{i=1}^{N_o} w_i^o\, \phi(s|s_i^o, \Sigma_i^o),$$

is (approximately) uniform in the area covered by T.

As far as the transition model is concerned, we assume it is linear-Gaussian

$$p(s'|s,a) = \phi(s'|s+\Delta(a), \Sigma^a), \tag{10}$$

with ϕ a Gaussian centered at $s+\Delta(a)$ with covariance Σ^a. The function $\Delta(\cdot)$ is a mapping from the action space to the state space and encodes the changes in the state produced by each action. For discrete action sets, this function can be seen as a table with one entry per action.

Finally, the reward model $r_a(s)$ is defined by a linear combination of (a fixed number of) Gaussians

$$r_a(s) = \sum_i w_i\, \phi_i(s|\mu_i^a, \Sigma_i^a),$$

where μ_i^a and Σ_i^a are the mean and covariance of each Gaussian.

5.2 α-Functions Representation

As mentioned above, we require an α-function representation that allow us to get a closed expression for the $\alpha_{a,o}^j$ (Eq. 7). With the above models, the α-functions can be represented by a linear combination of Gaussians as stated in the following lemma.

Lemma 5 *The functions $\alpha_n^i(s)$ can be expressed as linear combinations of Gaussians, assuming the observation, transition and reward models are also linear combinations of Gaussians.*

Proof This lemma can be proved via induction. For $n=0$, $\alpha_0^i(s) = r_a(s)$ for a fixed a and thus it is indeed a linear combination of Gaussians. For $n>0$, we assume that

$$\alpha_{n-1}^j(s') = \sum_k w_k^j\, \phi(s'|s_k^j, \Sigma_k^j).$$

Then, with our particular models, $\alpha_{a,o}^j(s)$ in Eq. 6 is the integral of three linear combinations of Gaussians

$$\begin{aligned}
\alpha_{a,o}^j(s) &= \int_{s'} \Big[\sum_k w_k^j\, \phi(s'|s_k^j, \Sigma_k^j)\Big] \Big[\sum_l w_l^o\, \phi(s'|s_l^o, \Sigma_l^o)\Big]\, \phi(s'|s+\Delta(a), \Sigma^a)\, ds' \\
&= \int_{s'} \sum_{k,l} w_k^j\, w_l^o \phi(s'|s_k^j, \Sigma_k^j)\, \phi(s'|s_l^o, \Sigma_l^o)\, \phi(s'|s+\Delta(a), \Sigma^a)\, ds' \\
&= \sum_{k,l} w_k^j\, w_l^o \int_{s'} \phi(s'|s_k^j, \Sigma_k^j)\, \phi(s'|s_l^o, \Sigma_l^o)\, \phi(s'|s+\Delta(a), \Sigma^a)\, ds'.
\end{aligned}$$

To compute this equation, we have to perform the product of two Gaussians and a closed formula is available for this operation

$$\phi(x|a,A)\,\phi(x|b,B) = \delta\,\phi(x|c,C),$$

with

$$\begin{aligned}\delta &= \phi(a|b,A+B) = \phi(b|a,A+B),\\ C &= (A^{-1}+B^{-1})^{-1},\\ c &= C\,(A^{-1}\,a + B^{-1}\,b).\end{aligned}$$

In the above case, we have to apply this formula twice, once for $\phi(s'|s_k^j,\Sigma_k^j)$ and $\phi(s'|s_l^o,\Sigma_l^o)$ to get $(\delta_{k,l}^{j,o}\,\phi(s'|s_1,\Sigma_1))$ and once more for $(\delta_{k,l}^{j,o}\,\phi(s'|s_1,\Sigma_1))$ and $\phi(s'|s+\Delta(a),\Sigma^a)$ to get $(\delta_{k,l}^{j,o}\,\beta_{k,l}^{j,o,a}(s)\,\phi(s'|s,\Sigma))$. The scaling terms $\delta_{k,l}^{j,o}$ and $\beta_{k,l}^{j,o,a}(s)$ can be expressed as

$$\begin{aligned}\delta_{k,l}^{j,o} &= \phi(s_l^o|s_k^j,\Sigma_k^j+\Sigma_l^o),\\ \beta_{k,l}^{j,o,a}(s) &= \phi(s|s_{k,l}^{j,o}-\Delta(a),\Sigma_{k,l}^{j,o}+\Sigma^a),\end{aligned}$$

with

$$\begin{aligned}\Sigma_{k,l}^{j,o} &= [(\Sigma_k^j)^{-1}+(\Sigma_l^o)^{-1}]^{-1},\\ s_{k,l}^{j,o} &= \Sigma_{k,l}^{j,o}\,[(\Sigma_k^j)^{-1}\,s_k^j+(\Sigma_l^o)^{-1}\,s_l^o].\end{aligned}$$

With this, we have

$$\begin{aligned}\alpha_{a,o}^j(s) &= \sum_{k,l} w_k^j\,w_l^o \int_{s'} \delta_{k,l}^{j,o}\,\beta_{k,l}^{j,o,a}(s)\,\phi(s'|s,\Sigma)\,ds'\\ &= \sum_{k,l} w_k^j\,w_l^o\,\delta_{k,l}^{j,o}\,\beta_{k,l}^{j,o,a}(s)\int_{s'}\phi(s'|s,\Sigma)\,ds'\\ &= \sum_{k,l} w_k^j\,w_l^o\,\delta_{k,l}^{j,o}\,\beta_{k,l}^{j,o,a}(s).\end{aligned}$$

Using Eqs. 7 and 8, we define the elements in $\{\alpha_n^i\}$ as

$$\alpha_n^i = r_a + \gamma\sum_o \arg\max_{\{\alpha_{a,o}^j\}_j}\langle\alpha_{a,o}^j,b\rangle.$$

Since the result of the arg max is just one of the members of the set $\{\alpha_{a,o}^j\}_j$, all the elements involved in the definition of α_n^i are linear combinations of Gaussians and so is the final result. ■

One point that deserves special consideration is the explosion of the number of components in the linear combinations of Gaussians defining the α-functions. If N_o is the number of components in the observation model and C_r is the average number of components in the reward model, the number of components in the α_n-functions scales with $O((N_o)^n\,C_r)$. Appendix A details an algorithm to bound the number of components of a mixture while losing as little information as possible.

5.3 Belief Representation

To get a belief update and an expectation operator that are computable, we consider two possible representations for the beliefs. The first one is Gaussian-based and the second one is particle-based.

5.3.1 GAUSSIAN-BASED REPRESENTATION

In this first case, we will assume that belief points are represented as Gaussian mixtures

$$b(s) = \sum_j w_j \, \phi(s|s_j, \Sigma_j), \tag{11}$$

with ϕ a Gaussian with mean s_j and covariance matrix Σ_j and where the mixing weights satisfy $w_j > 0$, $\sum_j w_j = 1$. In the extreme case, Gaussian mixtures with an infinite number of components would be necessary to represent a given point in the infinite-dimensional belief space of a continuous-state POMDP. However, only Gaussian mixtures with few components are needed in practical situations.

The belief update on Eq. 2 can be implemented in our model taking into account that it consists of two steps. The first one is the application of the action model on the current belief state. This can be computed as the propagation of the Gaussians representing $b(s)$ (Eq. 11) through the transition model (Eq. 10)

$$p(s'|b,a) = \int_s p(s'|s,a)\, b(s)\, ds = \sum_j w_j \, \phi(s|s_j + \Delta(a), \Sigma_j + \Sigma^a).$$

In the second step of the belief update, the prediction obtained with the action model is corrected using the information provided by the observation model

$$\begin{aligned} b^{a,o}(s') &\propto \Big[\sum_i w_i^o \, \phi(s'|s_i^o, \Sigma_i^o)\Big] \Big[\sum_j w_j \, \phi(s|s_j + \Delta(a), \Sigma_j + \Sigma^a)\Big] \\ &= \sum_{i,j} w_i^o \, w_j \, \phi(s'|s_i^o, \Sigma_i^o) \, \phi(s|s_j + \Delta(a), \Sigma_j + \Sigma^a). \end{aligned}$$

As mentioned, the product of two Gaussian functions is a scaled Gaussian. Therefore, we have that

$$b^{a,o}(s') \propto \sum_{i,j} w_i^o \, w_j \, \delta_{i,j}^{a,o} \, \phi(s'|s_{i,j}^{a,o}, \Sigma_{i,j}^{a,o}),$$

with

$$\begin{aligned} \delta_{i,j}^{a,o} &= \phi(s_j + \Delta(a) \,|\, s_i^o, \Sigma_i^o + \Sigma_j + \Sigma^a), \\ \Sigma_{i,j}^{a,o} &= ((\Sigma_i^o)^{-1} + (\Sigma_j + \Sigma^a)^{-1})^{-1}, \\ s_{i,j}^{a,o} &= \Sigma_{i,j}^{a,o}((\Sigma_i^o)^{-1} \, s_i^o + (\Sigma_j + \Sigma^a)^{-1} \, (s_j + \Delta(a))). \end{aligned}$$

Finally, we can rearrange the terms to get

$$b^{a,o}(s') \propto \sum_k w_k \, \phi(s'|s_k, \Sigma_k),$$

with $w_k = w_i^o \, w_j \, \delta_{i,j}^{a,o}$, $s_k = s_{i,j}^{a,o}$, and $\Sigma_k = \Sigma_{i,j}^{a,o}$ for all possible i, j. The proportionality in the definition of $b^{a,o}(s')$ implies that the weights (w_k, $\forall k$) should be scaled to sum to one

$$b^{a,o}(s') = \frac{1}{\sum_k w_k} \sum_k w_k \, \phi(s'|s_k, \Sigma_k).$$

An increase in the number of components representing a belief occurs when computing the belief update just detailed. If b_0 has C_b components and $p(o|s)$ is represented with an average of C_o components, the number of components in the belief b_t scales with $O(C_b(C_o)^t)$. As in the case of the α-functions, the procedure detailed in Appendix A could be used to bound the number of components in the beliefs.

Taking into account that the α-functions are also Gaussian-based, the expectation operator $\langle \cdot, \cdot \rangle$ can be computed in closed form as

$$\begin{aligned}
\langle \alpha, b \rangle &= \int_s \Big[\sum_k w_k \, \phi(s|s_k, \Sigma_k) \Big] \Big[\sum_l w_l \, \phi(s|s_l, \Sigma_l) \Big] \, ds \\
&= \sum_{k,l} w_k \, w_l \int_s \phi(s|s_k, \Sigma_k) \, \phi(s|s_l, \Sigma_l) \, ds \\
&= \sum_{k,l} w_k \, w_l \, \phi(s_l|s_k, \Sigma_k + \Sigma_l) \int_s \phi(s|s_{k,l}, \Sigma_{k,l}) \, ds \\
&= \sum_{k,l} w_k \, w_l \, \phi(s_l|s_k, \Sigma_k + \Sigma_l).
\end{aligned}$$

5.3.2 PARTICLE-BASED REPRESENTATION

An alternative to parameterize the belief densities using Gaussian mixtures is to represent the belief using N random samples, or particles, positioned at points s_i and with weights w_i. Thus, the belief is

$$b_t(s) = \sum_{i=1}^{N} w_i \, d(s - s_i),$$

where $d(s - s_i)$ is a Dirac's delta function centered at 0. Particle-based representations have been very popular in recent years, and they have been used in many applications from tracking to Simultaneous Localization and Mapping, SLAM, (see Doucet et al., 2001, for a review).

A particle-based representation has many advantages: it can approximate arbitrary probability distributions (with an infinite number of particles in the extreme case), it can accommodate non-linear transition models without the need of linearizing the model, and it allows several quantities of interest to be computed more efficiently than with the Gaussian-based belief representation. In particular, the integral in the belief update equation becomes a simple sum

$$b^{a,o}(s') \propto p(o|s') \sum_{i=1}^{N} w_i \, p(s'|s_i, a).$$

The central issue in the particle filter approach is how to obtain a set of particles to approximate $b^{a,o}(s')$ from the set of particles approximating $b(s)$. The usual Sampling Importance Re-sampling (SIR) approach (Dellaert et al., 1999; Isard and Blake, 1998) samples particles s'_i using the motion model $p(s'|s_i, a)$, then it assigns a new weight to each one of these particles proportional to the likelihood $p(o|s'_i)$, and finally it re-samples particles using these new weights in order to make all particles weights equal. The main problem of the SIR approach is that it requires many particles to converge when the likelihood $p(o|s')$ is too peaked or when there is only a small overlap between the prior and the posterior likelihood.

In the auxiliary particle filter (Pitt and Shephard, 1999) the sampling problem is addressed by inserting the likelihood inside the mixture

$$b^{a,o}(s') \propto \sum_{i=1}^{N} w_i \, p(o|s') \, p(s'|s_i, a).$$

The state s' used to define the likelihood $p(o|s')$ is not observed when the particles are resampled and we have to resort to approximations

$$b^{a,o}(s') \propto \sum_{i=1}^{N} w_i \, p(o|\mu_i) \, p(s'|s_i, a).$$

with μ_i any likely value associated with the *i-th* component of the transition density $p(s'|s_i, a)$, for example its mean. In this case, we have that $\mu_i = s_i + \Delta(a)$. Then $b^{a,o}(s')$ can be regarded as a mixture of the N transition components $p(s'|s_i, a)$ with weights $w_i \, p(o|\mu_i)$. Therefore, sampling a new particle s'_j to approximate $b^{a,o}(s')$ can be carried out by selecting one of the N components, say i_j, with probability $w_i \, p(o|\mu_i)$ and then sampling s'_j from the corresponding component $p(s'|s_{i_j}, a)$. Sampling is performed in the intersection of the prior and the likelihood and, consequently, particles with larger prior and larger likelihood (even if this likelihood is small in absolute value) are more likely to be used.

After the set of states for the new particles is obtained using the above procedure, we have to define their weights. This is done using

$$w'_j \propto \frac{p(o|s'_j)}{p(o|\mu_{i_j})}.$$

Using the sample-based belief representation the averaging operator $\langle \cdot, \cdot \rangle$ becomes

$$\begin{aligned}
\langle \alpha, b \rangle &= \int_s \Big[\sum_k w_k \, \phi(s|s_k, \Sigma_k) \Big] \Big[\sum_l w_l \, d(s - s_l) \Big] \, ds \\
&= \sum_k w_k \int_s \phi(s|s_k, \Sigma_k) \sum_l w_l \, d(s - s_l) \, ds \\
&= \sum_k w_k \sum_l w_l \, \phi(s_l|s_k, \Sigma_k) \\
&= \sum_{k,l} w_k \, w_l \, \phi(s_l|s_k, \Sigma_k).
\end{aligned}$$

Other re-sampling strategies such as those proposed by Fox (2003) that on-line adapt the number of sampled particles can also be applied here.

Given the common features between beliefs and α-functions in value iteration (i.e., beliefs and α-functions are both continuous functions of the state space), the α-functions also admit a particle representation. Note however that we cannot have both beliefs and α-functions represented by particles since the computation of $\langle \alpha, b \rangle$ requires that either b or α be in functional form to generalize over the entire state space.

Perseus
Input: A POMDP.
Output: V_n, an approximation to the optimal value function, V^*.

Initialize
  $B \leftarrow$ A set of randomly sampled belief points.
  $\alpha \leftarrow \frac{\min\{R\}}{1-\gamma} U$
  $n \leftarrow 0$
  $V_n \leftarrow \{\alpha\}$
do
  $\forall b \in B$,
    $\texttt{Element}_n(b) \leftarrow \arg\max_{\alpha \in V_n} \langle \alpha, b \rangle$
    $\texttt{Value}_n(b) \leftarrow \langle \texttt{Element}_n(b), b \rangle$
  $V_{n+1} \leftarrow \emptyset$
  $\tilde{B} \leftarrow B$
  do
    $b \leftarrow$ Point sampled randomly from $\tilde{B}$.
    $A \leftarrow \texttt{SampleActions}(b)$
    $\alpha \leftarrow \texttt{backup}(b)$
    if $\langle \alpha, b \rangle < \texttt{Value}_n(b)$
      $\alpha \leftarrow \texttt{Element}_n(b)$
    endif
    $\tilde{B} \leftarrow \tilde{B} \setminus \{b' \in \tilde{B} \mid \langle \alpha, b' \rangle \geq \texttt{Value}_n(b')\}$
    $V_{n+1} \leftarrow V_{n+1} \cup \{\alpha\}$
  until $\tilde{B} = \emptyset$
  $n \leftarrow n+1$
until convergence

Table 2: Modification of the PERSEUS algorithm in Table 1 to deal with large or continuous action spaces.

6. Extensions to Continuous Action and Observation Spaces

In Section 4, we presented a point-based value iteration algorithm to deal with continuous-state POMDPs. Now, we describe how to extend the presented framework to deal with continuous sets of actions and observations so that fully continuous POMDPs can also be addressed. The basic idea is that general continuous POMDPs can be cast in the continuous-state POMDP paradigm via sampling strategies.

6.1 Dealing with Continuous Actions

The backup operator in continuous-state value iteration requires computing a set of α-functions in Eq. 8, one function for each action $a \in A$, and then choosing the best function to back up. When the action space A is finite and small, the above optimization can be carried out efficiently by enumerating all possible actions and choosing the best, but in very large discrete action spaces this is

computationally inefficient. In this case, or when actions are continuous, one can resort to sampling-based techniques. As proposed by Spaan and Vlassis (2005), we can replace the full maximization over actions with a *sampled max* operator that performs the maximization over a subset of actions randomly sampled from A. One may devise several sampling schemes for choosing actions from A, for example, uniform over A or using the best action up to a given moment. Actions sampled uniformly at random can be viewed as exploring actions, while the latter can be viewed as exploiting current knowledge. Spaan and Vlassis (2005) provide more details on this point.

The use of a sampled max operator is very well suited for the point-based backups of PERSEUS, in which we only require that the values of belief points do not decrease over two consecutive backup stages. However, some modifications need to be introduced in the action and reward models described in Section 5.1. The action model described can be easily extended to continuous actions defining a continuous instead of a discrete function $\Delta : A \rightarrow S$ and evaluating it for the actions in the newly sampled A. As far as the reward model is concerned, we simply need to evaluate it for the sampled actions. Table 2 describes a modification of the basic PERSEUS algorithm to deal with large or continuous action spaces. Observe that, before computing the `backup` for the randomly selected belief point b (line 14b), we have to sample a new set of actions, A (line 14a) and the transition and reward models have to be modified accordingly, since they depend on the action set. Beside the action sampling and the on-line models computation, the rest of the algorithm proceeds the same as the one in Table 1. Note however, that the actions are sampled specifically for each belief b and, therefore we can not compute something similar to the $\alpha^j_{a,o}$-elements (see Eq. 6) that are common for all beliefs.

6.2 Dealing with Continuous Observations

In value iteration, the backup of a belief point b involves computing the expectation over observations

$$V_n(b) = \arg\max_a \Big\{ \langle r_a, b\rangle + \gamma V_{n-1}(b^a) \Big\},$$

with

$$V_{n-1}(b^a) = \int_o p(o|b,a)\, V_{n-1}(b^{a,o})\, do.$$

Using the definition of value function, the above reads

$$V_{n-1}(b^a) = \int_o p(o|b,a) \max_{\{\alpha^j_{n-1}\}_j} \langle \alpha^j_{n-1}, b^{a,o}\rangle \, do. \tag{12}$$

Building on an idea proposed by Hoey and Poupart (2005), assuming a finite number of α-elements α^j_{n-1}, observation spaces can always be discretized without any loss of information into regions corresponding to each α-element. In Eq. 12, all observations that lead to belief states $b^{a,o}$ with the same maximizing α-element can be aggregated together into one meta observation $O^j_{a,b}$ defined as follows

$$O^j_{a,b} = \{o \,|\, \alpha^j_{n-1} = \arg\max_{\{\alpha^j_{n-1}\}_j} \langle \alpha, b^{a,o}\rangle\}.$$

Using $O^j_{b,a}$, we can rewrite Eq. 12 as a sum over α-elements

$$V_{n-1}(b^a) = \sum_j \int_{o \in O^j_{a,b}} p(o|b,a)\, \langle \alpha^j_{n-1}, b^{a,o}\rangle \, do.$$

Rewriting the observation probabilities $p(o|b,a)$ in terms of s', we obtain

$$V_{n-1}(b^a) = \Big\langle \sum_j \int_{s'} \alpha^j_{n-1}(s') \Big[\int_{o \in O^j_{a,b}} p(o|s')\, do \Big]\, p(s'|s,a)\, ds', b \Big\rangle.$$

Hoey and Poupart (2005) assume a discrete state space, in which case the above quantity can be simplified by accumulating probability masses over observations in $O^j_{a,b}$, that is, defining $p(O^j_{a,b}|s') = \int_{o \in O^j_{a,b}} p(o|s')\, do$ for each state s', and then approximating $p(O^j_{a,b}|s')$ by sampling observations from $p(o|s')$. When the variable s' is continuous we can sample observations by *importance sampling* from some proposal distribution $q(o)$. With this we have

$$p(O^j_{a,b}|s') \simeq \frac{1}{N} \sum_{i=1}^{N_j} \frac{p(o^j_i|s')}{q(o^j_i)},$$

with $O^j_{a,b} = \{o^j_1, \ldots, o^j_{N_j}\}$ the set of observations for which α^j_{n-1} is maximal. The proposal distribution $q(o)$ can be, for instance, $p(o|b')$ with b' uniform in S or $p(o|b')$ with b' the current belief point. In our experiments we simply used a uniform distribution in O.

When working with continuous observations, the model given in Section 5.1 is no longer valid. However, we can assume the observation model to be defined using kernel smoothing from a training set including state-observation tuples. From those samples we can define

$$p(o,s) = \sum_{i=1}^{N} \lambda_i\, \phi(o|o_i, \Sigma^o_i)\, \phi(s|s_i, \Sigma^s_i),$$

and, using that,

$$p(o|s) = \frac{p(o,s)}{p(s)}.$$

Assuming a uniform $p(s)$, we have that $p(o|s)$ for a fixed observation is a Gaussian in s', which guarantees that α-functions remain closed under Bellman backups (see Section 5.1). With the observation model in the above form, we can further simplify $p(O^j_{a,b}|s')$ since we have that

$$\begin{aligned}
p(O^j_{a,b}|s') &\simeq \frac{1}{N} \sum_{i=1}^{N_j} \frac{p(o^j_i|s')}{q(o^j_i)} \\
&= \frac{1}{N} \sum_{i=1}^{N_j} \frac{1}{q(o^j_i)} \Big[\sum_{k=1}^{N} \lambda_k\, \phi(o^j_i|o_k, \Sigma^o_k)\, \phi(s|s_k, \Sigma^s_k) \Big] \\
&= \sum_{k=1}^{N} \Big[\frac{1}{N} \sum_{i=1}^{N_j} \frac{1}{q(o^j_i)} \lambda_k\, \phi(o^j_i|o_k, \Sigma^o_k) \Big]\, \phi(s|s_k, \Sigma^s_k) \\
&= \sum_{k=1}^{N} \rho^j_k\, \phi(s|s_k, \Sigma^s_k)
\end{aligned}$$

with

$$\rho^j_k = \frac{\lambda_k}{N} \sum_{i=1}^{N_j} \frac{\phi(o^j_i|o_k, \Sigma^o_k)}{q(o^j_i)}.$$

With the discretized observation model we can define

$$\alpha^j_{a,b} = \sum_j \int_{s'} \alpha^j_{n-1}(s')\, p(O^j_{a,b}|s')\, p(s'|s,a)\, ds',$$

that plays the same role in the Bellman backup as the $\alpha^j_{a,o,b}$-functions introduced in Eq. 7. With the above we have

$$V_{n-1}(b^a) = \langle \alpha^j_{a,b}, b \rangle,$$

and the α-elements for V_n at belief b are defined as

$$\{\alpha^i_n\}_i = \{r_a + \gamma \alpha^j_{a,b}\}_{a \in A}.$$

Thus, as far as implementation is concerned, continuous observation spaces introduce a modification in the `backup`, but this modification is independent of the rest of the algorithm. Therefore this new operator can be used both in PERSEUS with either discrete or sampled continuous actions (see Table 1 and Table 2, respectively).

Note that if we work with continuous observation spaces, the $\alpha^j_{a,b}$-functions are computed specifically for each belief and, therefore no precomputation similar to those of the $\alpha^j_{a,o}$-elements is possible.

7. Experiments and Results

To demonstrate the viability of our method we carried out some experiments in a simulated robotic domain. The simulation was programmed in Matlab 7.1 using a Pentium Xeon at 3 GHz running under Linux. In the simulated problem (see Fig. 1-a), a robot is moving along a corridor with four doors, where the state space is the continuous interval $[-21, 21]$. The target for the robot is to locate the second door from the right and enter it. The robot only receives positive reward when it enters the target door (see Fig. 1-c). When the robot tries to move beyond the end of the corridor (either right or left), or when it tries to enter a door at a wrong position, it receives negative reward. The reward function is represented using a linear combination of nine Gaussian functions. Three Gaussians are placed at each extreme of the corridor to represent the negative reward for trying to move beyond the end of the corridor (with means $\pm 21, \pm 19, \pm 17$, covariance 0.05, and weight -2). Two Gaussians represent the negative reward for trying to enter a door at the wrong position (with means ± 25, covariance 12.5, and weight -10). Finally, one Gaussian is used for the positive reward associated with entering the correct door (with mean 3, covariance 0.15, and weight 2).

In all reported experiments, the set of beliefs B used in the PERSEUS algorithm contains 500 unique belief points collected using random walks departing from a uniform belief, the latter being approximated with a Gaussian mixture with four components. The walks of the robot along the corridor are organized in episodes of 30 actions (thus, for instance, the robot can repetitively try to enter the correct door accumulating positive reward). In all experiments we set $\gamma = 0.95$.

In the first experiment we assume discrete observations and actions. There are four distinct observations, *left-end*, *right-end*, *door*, and *corridor*. The observation model, shown in Fig. 1-b, is approximated using a training set of 22 samples evenly placed every two space units from -21 to 21 (with $\Sigma^o = 4$). The five right/left-most samples correspond to observations *right-end* and *left-end*, respectively, each sample taken in front of a door corresponds to observation *door*, and the rest of the samples correspond to observation *corridor*. There are three distinct actions: the robot can

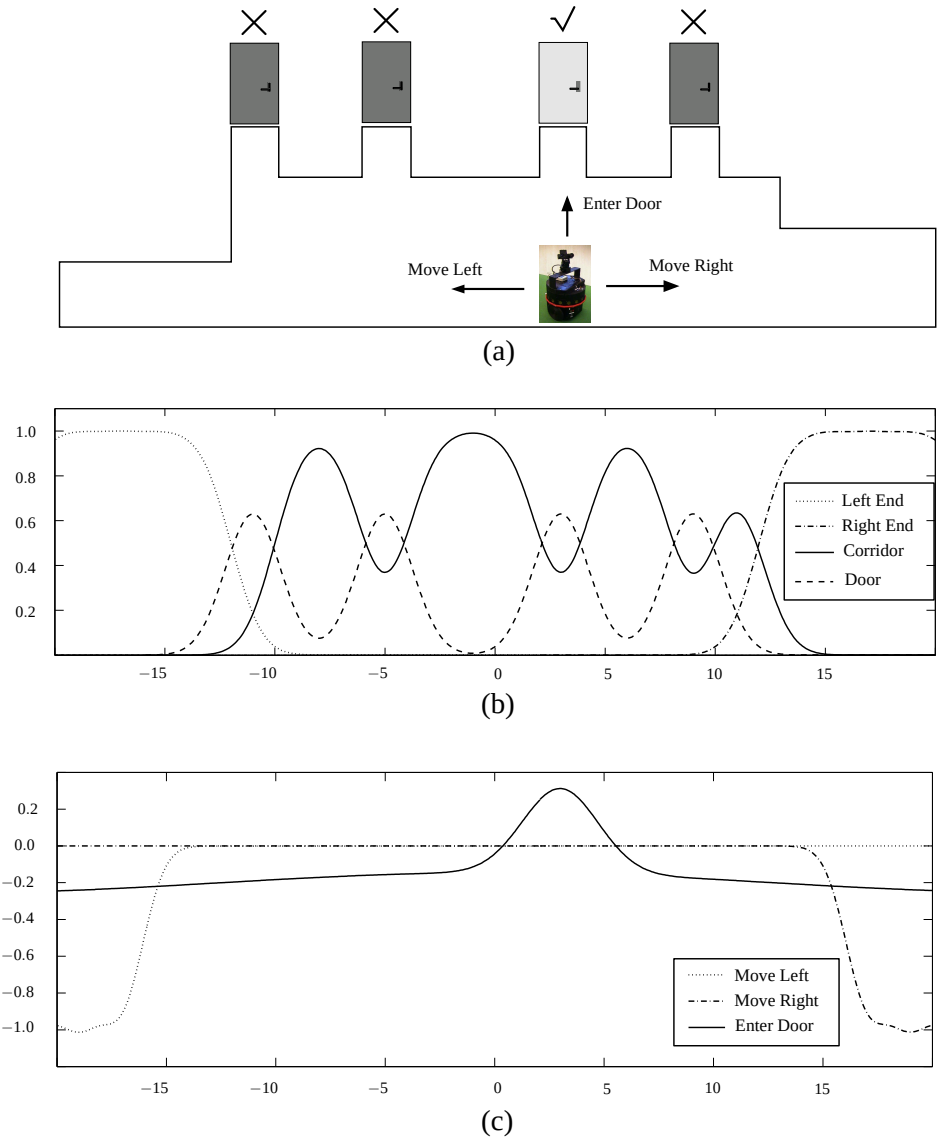


Figure 1: A pictorial representation of the test problem (a), the corresponding observation model (b), and the reward model (c).

move two units either to the left or to the right (with $\Sigma^a = 0.05$), or it can try to enter a door at any point. In this experiment we used Gaussian mixtures to represent the beliefs, compressing them, if necessary, to a maximum of four components, and similarly we used α-functions with a maximum of nine Gaussian components.

Fig. 2 shows the average results obtained after 10 runs of the version of PERSEUS described in Section 4. The first plot (top-left) shows the convergence of the value computed as $\sum_b V(b)$. The second plot (top-right) shows the expected discounted reward computed by running for 50 episodes the policy available at the corresponding time slice. The fact that this plot converges to a positive

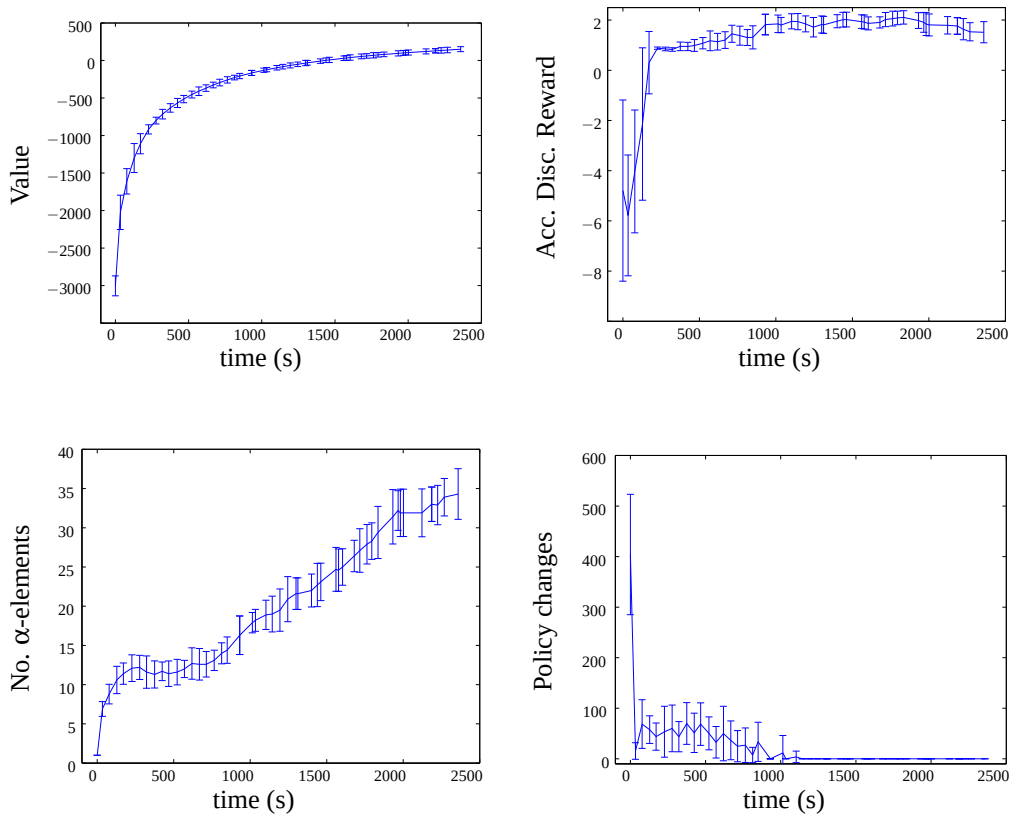


Figure 2: Results for the simulated robotic problem using continuous states but discrete actions and observations. Top: Evolution of the value for all the beliefs in B and the average accumulated discounted reward for 10 episodes. Bottom: Number of elements in V_n and the number of policy changes. Results are averaged for 10 repetitions and the bars represent the standard deviation.

value indicates that the robot successfully learns to avoid collisions, to find out its position, and to identify the target door. Next plot (bottom-left) shows the number of α-functions used to represent the value function. We can see that the number of α-functions increases, but it remains far below 500, the maximum possible number of α-functions (in the extreme case we would use a different α-function for each point in B). Finally, the bottom-right plot shows the number of changes in the policy from one time step to another. The changes in the policy are computed as the number of beliefs in B with a different optimal action from one time slice to the next. The number of policy changes drops to zero, indicating convergence with respect to the particular B. In Fig. 3 we show a typical trajectory of the robot when executing a policy found at convergence of PERSEUS. The snapshots show the evolution of the belief of the robot, and the actions taken, from the beginning of the episode (the robot starts at location 7) until the target door is entered.

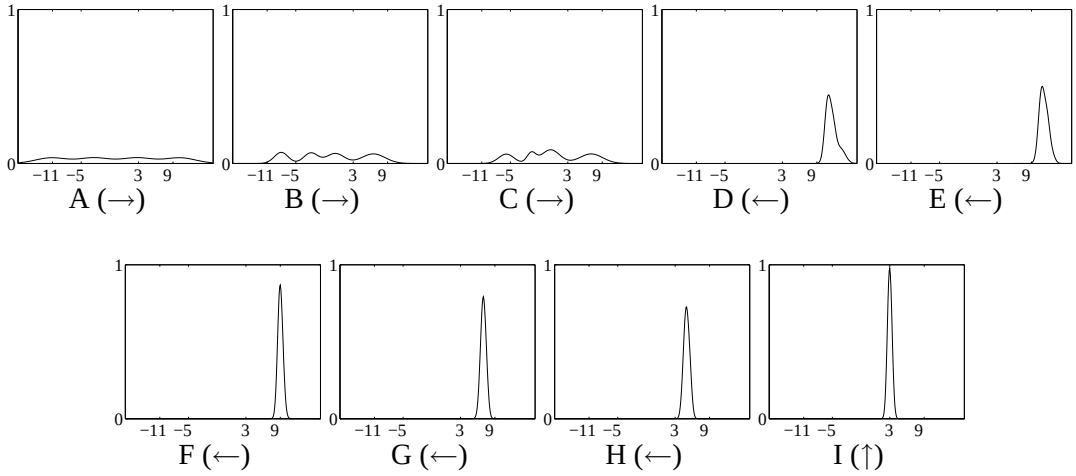


Figure 3: Evolution of the belief when following the discovered policy. The arrows under the snapshots represent the actions: → for moving right, ← for moving left, and ↑ for entering the door. The four numbers on the x-axis indicate the locations of the four doors.

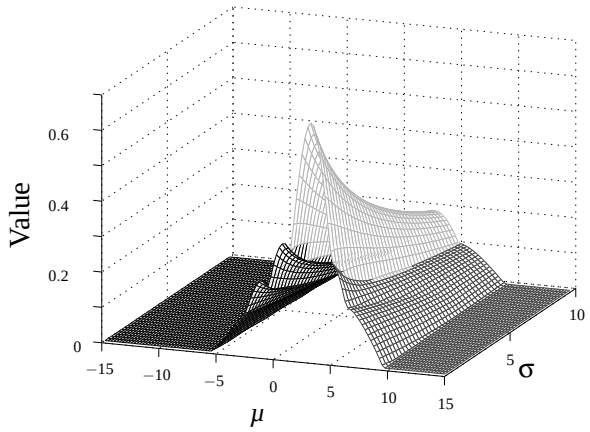


Figure 4: Value function for single-component beliefs as a function of the mean μ and the standard deviation σ.

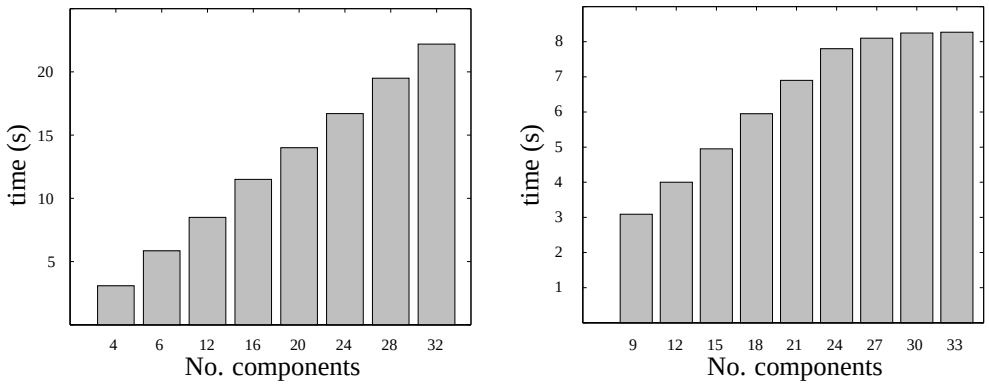


Figure 5: Execution time in seconds for the first iteration of PERSEUS as the number of components representing the beliefs increase (left) and as the number of components representing the α-functions increase (right).

Since the state space is one-dimensional in this example, beliefs with a single (Gaussian) component can be fully characterized by their sufficient statistics, that is, the mean μ and the variance σ^2. In Fig. 4 we plot the value of single-Gaussian beliefs for different μ and σ. We note that, as the uncertainty about the position of the robot grows (i.e., the σ is larger), the value of the corresponding belief decreases. The colors/shadings in the figure correspond the different actions: black for moving to the right, light-gray for entering the door, and dark-gray for moving to the left. This plot demonstrates that a value function that is convex over the belief space may not necessarily be convex over the space of sufficient statistics of the beliefs.

Fig. 5-left shows the increase in the execution time as more components are used to represent the beliefs. The plotted data correspond to the time in seconds for the first PERSEUS value update stage, that is, for the computation of the first `backup` (line 14 in Table 1) and the new value for all the beliefs in B (line 18 in the algorithm). The cost of executing the first iteration is an indicator of the computational complexity of the system that is independent of the problem at hand; the cost of later stages of PERSEUS scales with the number of elements in the previous value function approximation, V_{n-1}, and the number of elements to be generated for the new approximation, V_n, and both quantities are problem-dependent. We can see that the increase in the execution time is rather linear with the number of components in the belief. In all the experiments summarized in Fig. 5-left, we used nine components to represent the α-functions. To assess the effects of increasing the number of components in the α-functions, in Fig. 5-right we show the increase in the execution time for the first PERSEUS iteration when beliefs are represented with four components and the α-functions are represented with an increasing number of elements. We can observe that after about 24 components the execution time is almost constant. This is due to the fact that, for the problem at hand, no more components are needed to represent the α-functions. The Gaussian mixture condensation algorithm detailed in Appendix A has the property of discarding some components from the output if these are not necessary.

The effect on the quality of the solution when reducing the number of components for the beliefs and the α-functions can be seen in Fig. 6 where we depict the average accumulated discounted

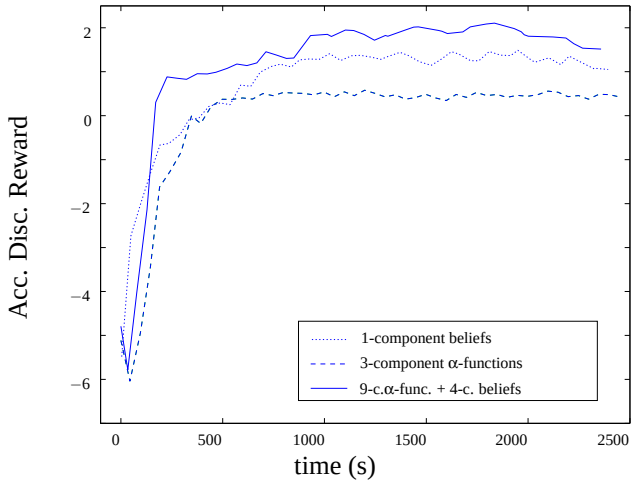


Figure 6: Reduction in the obtained average accumulated discounted reward when reducing the number of components in the beliefs to just one (dotted line) and in the α-functions to three (dashed line). The solid line is the average accumulated reward when using 4 components for the beliefs and 9 for the α-functions.

reward when representing beliefs with one component and α-functions with three components. We observe that when using fewer components for the α-functions, the algorithm may converge to a suboptimal policy. We also noticed that when using more than nine components, the improvements in the final policy were marginal. When representing the beliefs with just one component, the quality of the obtained policy also decreases. This is due to the fact that the problem at hand presents some degree of perceptual aliasing (i.e., states for which different actions are required but where the same observation is obtained). This aliasing can only be solved properly when using a multi-modal belief representation, which is not the case for single Gaussians.

We note that the advantage of using a continuous state space is that we obtain a scale-invariant solution. If we have to solve the same problem in a longer corridor, we can just scale the Gaussians used in the problem definition and we will obtain the solution with the same cost as we have now. The only difference is that more actions would be needed in each episode to reach the correct door.

Another way to solve this problem would be to discretize the state space and then apply the PERSEUS algorithm for discrete POMDPs. When discretizing the environment, the granularity has to be in accordance with the size of the actions taken by the robot (± 2 left/right) and, thus, the number of states and, consequently, the cost of the planning grows as the environment grows. Fig. 7-left shows the execution time in seconds for the first stage of PERSEUS in a discretized version of the problem as the number of states grows. The discretization is performed by selecting n states uniformly sampled on the state space and then using the continuous models to define the discrete ones. As we can see in the figure, the increase in the execution time with respect to the number of states is higher than linear. With more than 100 states the execution is slower than that for the continuous version when using 4 components for the beliefs and 9 for the α-functions (the dashed line in Fig. 7-left is the time for the execution of the first iteration of PERSEUS in this case).

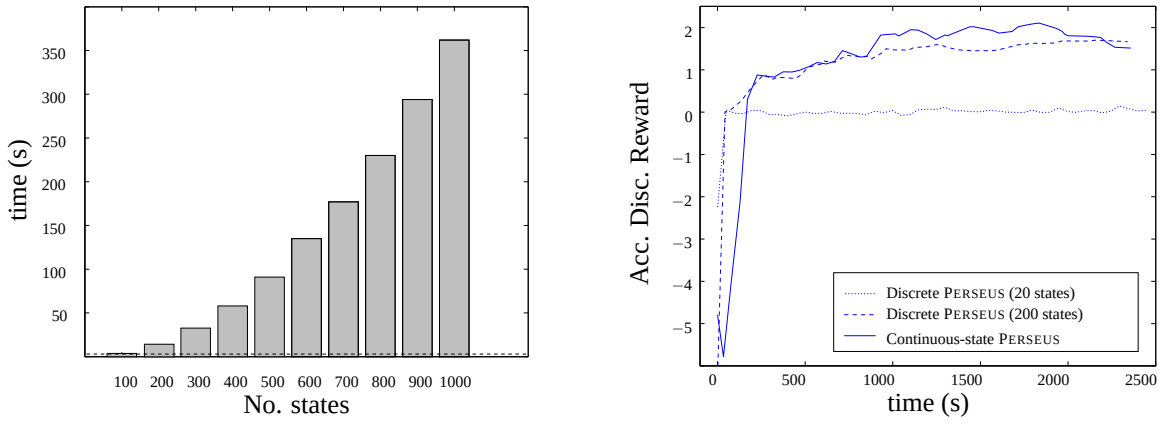


Figure 7: Left: Execution time in seconds for the first iteration of PERSEUS in a discretized version of the problem as the number of states grows. The dashed line is the time for the first iteration in the continuous-state version of the same problem. Right: Average accumulated discounted reward obtained with the continuous-state version of PERSEUS with 4 components for the beliefs and 9 for the α-functions (solid line) compared with the one obtained with the discrete version of PERSEUS using only 20 states (dotted line) and using 200 states (dashed line) .

Note that the discrete version of PERSEUS relies on linear algebra operations that can be sped up by taking advantage of the sparsity of the matrices and vectors defining the models and the beliefs, however, such speedups are not implemented in the version of PERSEUS we use for the experiments. A remarkable difference between the continuous and the discrete-state version is that the first one spends most of the time in the computation of the value for all beliefs (i.e., in the $\langle \cdot, \cdot \rangle$ operator) while the second one spends most of the time in the computation of the $\alpha^j_{a,o}$ vectors that are later on used in the backup. Fig. 7-right shows the average accumulated discounted reward obtained with the discrete version of PERSEUS working on different number of states compared with the one obtained in the first experiment (see Fig. 2). We can see that, when using a too coarse discretization (only 20 states) the discrete version of the problem does not capture all the features of the continuous one and, therefore, we observe convergence to a sub-optimal solution. Only when using enough states in the discretization the discrete version of PERSEUS delivers a plan that is as good as the one obtained with the continuous PERSEUS. The average accumulated discounted reward with a discretization with 200 states is shown in Fig. 7-right.

In the following experiment, the same problem was solved using particles to represent the beliefs instead of Gaussian mixtures. In this case, the α-functions are still represented as Gaussian mixtures, with 9 components. The results obtained using 75 particles are shown in Fig. 8. Note that the results are similar to those obtained when using Gaussian mixtures to represent the beliefs (see Fig. 2) but they are obtained in about 5 times more execution time. This is reasonable since, although the basic operations implementing the expectation operator $\langle \cdot, \cdot \rangle$ are more efficient when using particle-based beliefs, this is compensated by the fact that, in general, we have to use a large amount of particles. Therefore, the use of particles might only be advantageous when the belief cannot be represented

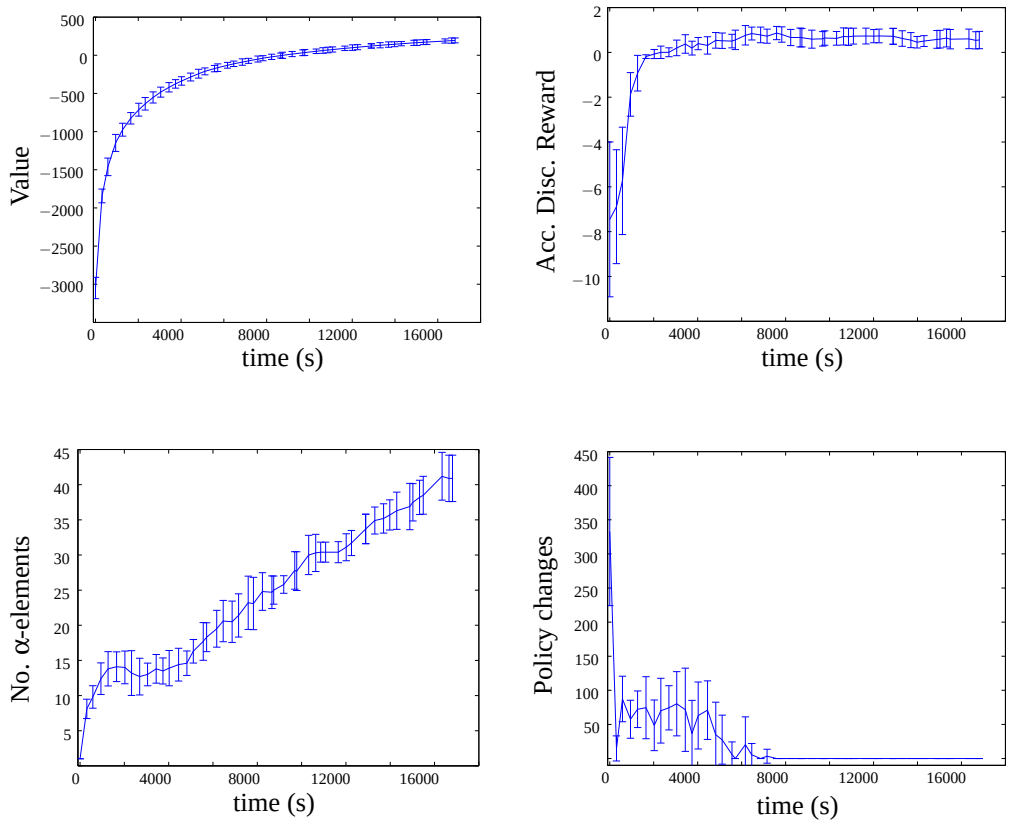


Figure 8: Results when using 75 particles to represent the beliefs. Top: Evolution of the value for all the beliefs in B and the average accumulated discounted reward for 10 episodes. Bottom: Number of elements in V_n and the number of policy changes. Results are averaged for 10 repetitions and the bars represent the standard deviation.

with a few-components Gaussian mixture, when the action model is not linear, or when using an on-line mechanism to dynamically adjust the number of particles (Fox, 2002, 2003).

Fig. 9 shows the average accumulated discounted reward using two different sets of actions. When using an action set including short robot movements (± 1), the number of steps to reach the target increase and, since positive reward is only obtained at the end of the run when entering the door, the average accumulated reward decreases. When using a set of actions with too large movements (± 4) the robot has problems aiming the correct door and the average accumulated reward also decreases. Since the appropriate set of actions for each problem is hard to forecast, it would be nice to have a planning system able to determine a proper set of actions by itself. For this purpose in the next experiment we let the robot execute actions in the continuous range $[-6, 6]$, where an action can be regarded here as a measure of velocity of the robot. When the robot is almost stopped (i.e., its velocity is below 5% of the maximum one) we interpret this as trying to enter a door. In each backup at planning stage n, we consider the optimal action according to V_{n-1} and three more

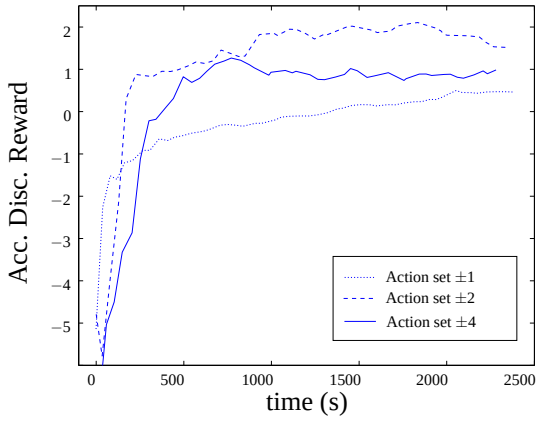


Figure 9: Average accumulated discounted reward using different sets of actions.

actions selected at random with uniform distribution in the range $[-6,6]$. Fig. 10 shows the average results obtained by 10 repetitions. The policy change in the bottom-right plot is computed as the sum squared difference of the actions in two consecutive PERSEUS iterations for all beliefs. The fact that this norm goes to zero means that policy gets stable and, observing the plot for the reward, we can see that the discovered policy is better than the one in Fig. 2, meaning that the algorithm is able to determine better motion actions than the ones we manually fixed in the initial version of the problem (± 2), and that is able to select *enter door* actions when necessary.

Finally, we modified the initial problem so that it is formalized with continuous state, action, and observation spaces. Here we assume that the robot observations are obtained with a noisy sensor that measures the width of the corridor. The observations are noise-perturbed versions of four nominal integer values: 1 for the right extreme, 2 for the left one, 4 for the doors, and 3 for the rest of positions (see Fig. 1-a). The sensor noise is assumed white Gaussian with covariance equal to 0.3, resulting in a continuous set of observations in the range $[0,5]$. For each backup, we still use four actions (the optimal one up for V_{n-1} and three randomly selected ones) and we discretize the observation space by uniformly sampling 100 observations. Fig. 11 shows the results obtained in this case. We see a performance similar to the one obtained with continuous states, continuous actions, and discrete observations, implying that the observation-space discretization does not affect the quality of the final policy. An interesting observation is that the algorithm converges faster and that the optimal value is approximated with fewer α-functions than when using discrete observations. This is probably due to the fact that the observation discretization takes advantage (and relies on) the structure of the previous approximation to the value function.

With this experiment we conclude our demonstration that the full continuous POMDP case can be addressed with the techniques proposed in this paper.

8. Related Work

The literature on POMDPs with continuous states is still relatively sparse. A common approach is to assume a discretization of the state space, which can be a poor model of the underlying system. However, when the system is linear and the reward function is quadratic, an exact solution for continuous-state POMDPs is known that can be computed in closed form (Bertsekas, 2001). While

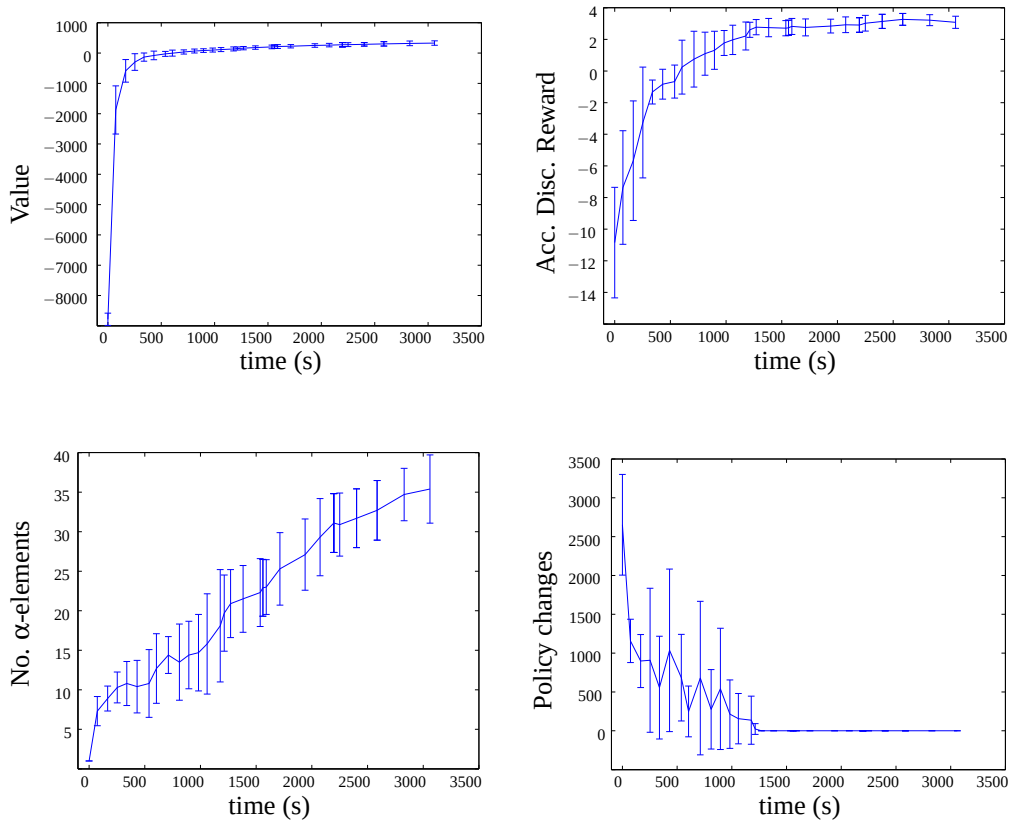


Figure 10: Results when the problem is modeled with continuous states and actions. Top: Evolution of the value for all the beliefs in B and the average accumulated discounted reward for 10 episodes. Bottom: Number of elements in V_n and the average policy change. Results are averaged for 10 repetitions and the bars represent the standard deviation.

such an assumption on the reward function can be reasonable in certain control applications, it is a severe restriction for the type of AI applications that we consider.

Roy (2003) has proposed compression techniques for handling POMDPs with large (discrete) state spaces, one of which compresses beliefs to two parameters: the state with maximum likelihood and the belief's entropy. Such a representation may lead to poor performance when multi-modal beliefs are likely to occur in a particular application. Recently, Brooks et al. (2006) have proposed a related parameterization of the beliefs using the sufficient statistics of an appropriately chosen parametric family (e.g., Gaussians). Both methods compute an approximate value function on a grid in their low-dimensional parameter spaces, and do not use the PWLC property of the POMDP value function. In contrast, we exploit the known shape of the value function, which offers an attractive potential for generalization through the use of α-functions, analogous to the effective exploitation of α-vectors in discrete-state POMDPs.

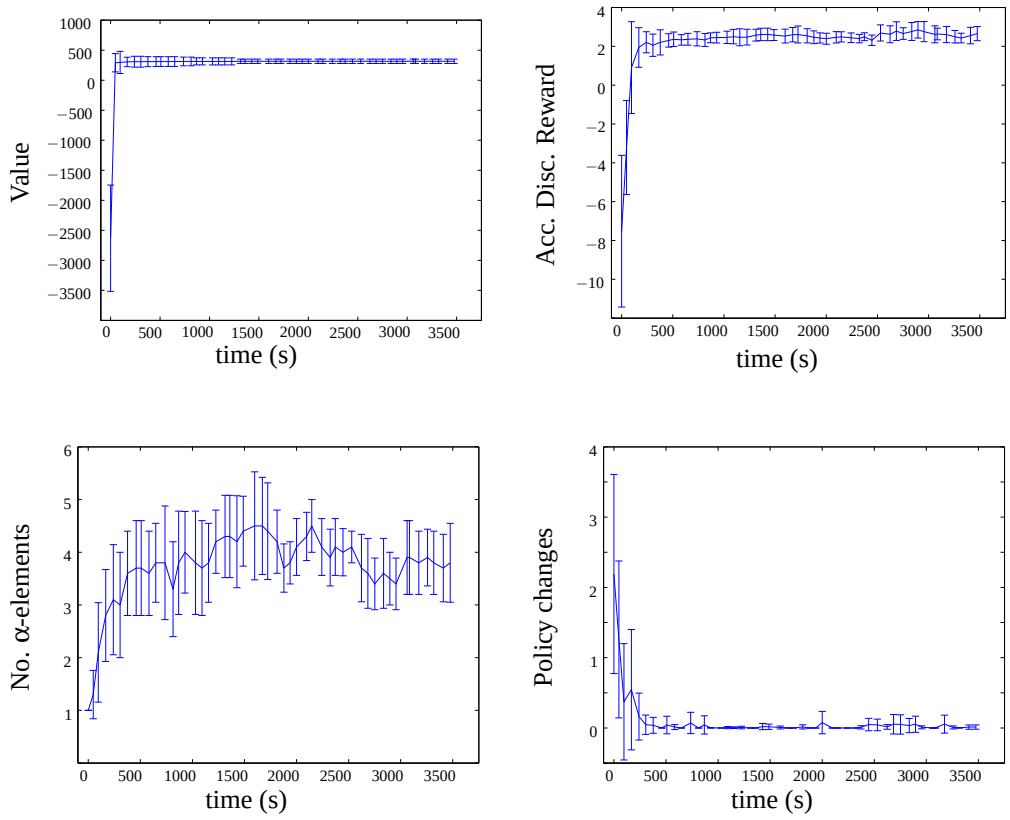


Figure 11: Results when the problem is modeled with continuous states, observations and actions. Top: Evolution of the value for all the beliefs in B and the average accumulated discounted reward for 100 episodes. Bottom: Number of elements in V_n and the number of policy changes. Results are averaged for 10 repetitions and the bars represent the standard deviation.

An approach to continuous-state POMDPs that is closely related to ours is the Monte Carlo POMDP (MC-POMDP) method of Thrun (2000), in which real-time dynamic programming is applied on a POMDP with a continuous state and action space. In that work beliefs are represented by sets of samples drawn from the state space, while $Q(b,a)$ values are approximated by nearest-neighbor interpolation from a (growing) set of prototype values and are updated by online exploration and the use of sampling-based Bellman backups. The MC-POMDP method approximates the Bellman backup operator by sampling from the belief transition model, whereas in our case, we compute the Bellman backup operator analytically given the particular value-function representation. In the MC-POMDP algorithm nearest-neighbor interpolation is used to approximate the value of beliefs outside the set. This is in contrast with our Gaussian-mixture representation, in which the value function achieves generalization through a set of α-functions. When the value function maintained by MC-POMDP does not contain enough neighbors within a certain distance for an encountered belief, the belief is added to the value function. PERSEUS operates on a fixed set of

beliefs, and does not require such an online expansion. Furthermore, the PERSEUS value function is likely to generalize better over the belief space through the use of α-functions. In contrast with PERSEUS, the MC-POMDP method does not exploit the piecewise linearity and convexity of the value function.

Duff (2002) considered the problem of Bayesian reinforcement learning, in which the parameters of the transition model of an MDP are treated as random variables. Experience in the form of observed state transitions and received rewards is used to estimate the unknown MDP models. In contrast with straightforward exploration strategies such as ε-greedy, Bayesian reinforcement-learning techniques try to identify the action that will maximize long-term reward. Such an optimally-exploring action might sacrifice expected immediate payoff for refining the model estimates, thus facilitating better control in the future. Duff (2002) models the Bayesian reinforcement-learning problem as a POMDP, in which the parameters of the transition model form the state of the system, and experienced transition tuples (s,a,s') are the possible observations. Such a POMDP has a continuous state space as the transition probabilities can be any real number between zero and one. A Monte Carlo algorithm is proposed for learning a (stochastic) finite-state controller for this particular class of POMDPs, where the required integrals are approximated by sampling and numerical methods. Recently, Poupart et al. (2006) demonstrated that the optimal value function in Bayesian reinforcement learning can be represented by a set of multivariate polynomials, in direct analogy to the α-function representations for Gaussian-based POMDPs in this paper.

In the case of continuous action spaces only few methods exist that can handle continuous action spaces directly (Thrun, 2000; Ng and Jordan, 2000; Baxter and Bartlett, 2001). Certain policy search methods tackle continuous actions, for instance Pegasus (Ng and Jordan, 2000), which estimates the value of a policy by simulating trajectories from the POMDP using a fixed random seed, and adapts its policy in order to maximize this value. Pegasus can handle continuous action spaces at the cost of a sample complexity that is polynomial in the size of the state space (Ng and Jordan, 2000, Theorem 3). Baxter and Bartlett (2001) propose a policy gradient method that searches in the space of randomized policies, and which can also handle continuous actions. The main disadvantages of policy search methods are the need to choose a particular policy class and the fact that they are prone to local optima.

Traditional POMDP methods assume discretized observation spaces. POMDPs with continuous observation spaces have mainly been studied in model-free settings, for instance to learn policies for a partially observable version of the classic pole-balancing task (Whitehead and Lin, 1995; Meuleau et al., 1999; Bakker, 2003). Rudary et al. (2005) extend Predictive State Representations (PSRs) to the linear-Gaussian case, which allows them to learn a PSR model of a linear dynamical system with a continuous observation space. Finally, the analytic solution for the quadratic reward case mentioned above can also handle continuous observations with a linear noise model (Bertsekas, 2001).

9. Conclusions and Future Work

In this paper we described an analytical framework for optimizing POMDPs with continuous states, actions, and observations. For POMDPs with continuous states, we demonstrated the piecewise linearity and convexity of value functions defined over infinite-dimensional belief states induced by continuous states. We also demonstrated that continuous Bellman backups are isotonic and contracting, allowing value iteration to be adapted to continuous POMDPs. In particular, we extended

the PERSEUS algorithm with linear combinations of Gaussians and particle-based representations for belief states. These are expressive representations that are closed under Bellman backups and belief updates. Finally, we also extended PERSEUS to continuous actions and observations by particular sampling strategies that reduce the problem to a continuous state POMDP that can be tackled by the PERSEUS algorithm.

In the near future, we plan to investigate reinforcement learning approaches for scenarios where the POMDP model is unknown. The particle-based approach may be adaptable to reinforcement learning since particles may be thought as sampled values. Conversely, note that discrete Bayesian reinforcement learning can be cast as a POMDP with continuous states (Duff, 2002). Poupart et al. (2006) recently developed a similar technique to optimize policies in environments with (partially) unknown transition dynamics modeled by multinomials. It would be interesting to follow up on this work by tackling Bayesian reinforcement learning problems with Gaussian-based dynamics.

Another interesting research direction would be to investigate which families of functions (beyond mixtures of Gaussians) are closed under Bellman backups and belief updates for different types of transition, observation and reward models. In particular, mixtures of log-linear distributions provide an expressive parameterization that is likely to possess the necessary properties.

Acknowledgments

We would like to thank Jakob Verbeek and Wojtek Zajdel for their contributions to the work reported here. Josep M. Porta has been partially supported by a *Ramón y Cajal* contract from the Spanish government and by the EU PACO-PLUS Project FP6-2004-IST-4-27657. Nikos Vlassis and Matthijs Spaan are supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414. Pascal Poupart is supported by the Canada's National Science and Engineering Research Council.

Appendix A.

As a large number of components representing beliefs and α-functions slows down the basic operations of the algorithm, an efficient implementation of the algorithm is required to keep the number of components reasonably bounded.

We use the procedure described by Goldberger and Roweis (2005) that transforms a given Gaussian mixture with k components to another Gaussian mixture with at most m components, $m < k$, while retaining the initial component structure. The algorithm is detailed in Table 3.

The algorithm uses the Kullback-Leibler, KL, distance between to Gaussian distributions $f_i = N(\mu,\Sigma)$, $g_j = N(\mu',\Sigma')$ that is

$$KL(f_i \| g_j) = \frac{1}{2}\left(\log\frac{|\Sigma'|}{|\Sigma|} + Tr((\Sigma')^{-1}\Sigma) + (\mu-\mu')^\top(\Sigma')^{-1}(\mu-\mu') - c\right),$$

with c the dimensionality of the space where the Gaussians are defined.

Observe that the above procedure is defined for Gaussian mixtures (with positive weights that sum to 1), but our α-functions are linear combinations of Gaussian (with possibly negative weights). Therefore, for the α-function compression, we use a modified version of the procedure just described where the weights are normalized after taking its absolute value. This way, the distance

Gaussian Mixture Condensation(f, m)
Input: A Gaussian mixture $f = \sum_{i=1}^{k} w_i \, f_i(x|\mu_i, \Sigma_i)$.
The maximum number of components
in the output mixture, m, $m < k$.
Output: A Gaussian mixture $g = \sum_{i=1}^{m} w'_i \, g_i(x|\mu'_i, \Sigma'_i)$ that
locally minimizes $\sum_{i=1}^{k} w_i \, min_{j\in[1,m]} KL(f_i \| g_j)$

1: **Initialize**
2: **for** $j = 1$ **to** m
3: $w'_j \leftarrow w_j$
4: $\mu'_j \leftarrow \mu_j$
5: $\Sigma'_j \leftarrow \Sigma_j$
6 : $d \leftarrow \sum_{i=1}^{k} w_i \min_{j\in[1,m]} KL(f_i \| g_j)$
7: **do**
8: **Compute the mapping from** f **to** g
9: **for** $i = 1$ **to** k
10: $\pi(i) \leftarrow \arg\min_{j\in[1,m], w'_j>0} KL(f_i \| g_j)$
11: **Define a new** g
12: **for** $j = 1$ **to** m
13: $I_j \leftarrow \{i \,|\, \pi(i) = j, \, i \in [1,k]\}$
14: $w'_j \leftarrow \sum_{i\in I_j} w_i$
15: $\mu'_j \leftarrow \frac{1}{w'_j} \sum_{i\in I_j} w_i \, \mu_i$
16: $\Sigma'_j \leftarrow \frac{1}{w'_j} \sum_{i\in I_j} w_i \, (\Sigma_i + (\mu_i - \mu'_j)(\mu_i - \mu'_j)^\top)$
17: $d' \leftarrow d$
18: $d \leftarrow \sum_{i=1}^{k} w_i KL(f_i \| g_{\pi(i)})$
19: **until** $d < \varepsilon$ **or** $\frac{|d-d'|}{d} < \varepsilon$

Table 3: Gaussian mixture condensation algorithm where ε is a sufficiently small threshold (10^{-5} in our implementation).

(locally) minimized by the algorithm in Table 3 is

$$d = \sum_{i=1}^{k} |w_i| \min_{j\in[1,m]} KL(f_i \| g_j).$$

Therefore, the algorithm tries to preserve the relevant peaks (either positive or negative) in the original mixture. After the compression, the weights are re-computed taken into account the original weights and the map π provided by the algorithm above.

References

K. J. Åström. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

D. Aberdeen and J. Baxter. Scaling internal-state policy-gradient methods for POMDPs. In *Proceedings of the International Conference on Machine Learning*, pages 3–10, Sydney, Australia, 2002.

B. Bakker. Reinforcement learning with long short-term memory. In *Advances in Neural Information Processing Systems 15 (NIPS-2002)*, pages 1475–1482. MIT Press, 2003.

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

R. E. Bellman. *Dynamic Programming*. Princenton University Press, 1957.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2001. 2nd Edition.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1293–1299, Edinburgh, Scotland, 2005.

C. Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 239–246, Edmonton, AB, 2002.

C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1168–1175, Portland, OR, 1996.

A. Brooks, A. Makarenko, S. Williams, and H. Durrant-Whyte. Planning in continuous state spaces with parametric POMDPs. In *Reasoning with Uncertainty in Robotics, Workshop of the International Joint Conference on Artificial Intelligence*, pages 40–47, 2006.

A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty: discrete Bayesian models for mobile-robot navigation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 963–972, 1996.

A. R. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact algorithm for partially observable Markov decision processes. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 54–61, 1997.

H. T. Cheng. *Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, University of British Columbia, 1988.

T. Darrell and A. P. Pentland. Active gesture recognition using partially observable Markov decision processes. In *IEEE International Conference on Pattern Recognition*, pages 984–988, Vienna, Austria, 1996.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte-Carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1322–1328, 1999.

A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo in Practice*. Springer-Verlag, New York, 2001.

M. Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massassachusetts Amherst, 2002.

E. B. Dynkin. Controlled random sequences. *Theory of probability and its applications*, 10(1): 1–14, 1965.

D. Fox. Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14 (NIPS-2001)*, pages 713–720. MIT Press, 2002.

D. Fox. Adapting the sample size in particle filters through kld-sampling. *International Journal of Robotics Research*, 22(10-11):985–1004, 2003.

J. Goldberger and S. Roweis. Hierarchical clustering of a mixture model. In *Advances in Neural Information Processing Systems 17 (NIPS-2004)*, pages 505–512. MIT Press, 2005.

J. Hoey and P. Poupart. Solving pomdps with continuous or large discrete observation spaces. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1332–1338, 2005.

M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

C. Lusena, J. Goldsmith, and M. Mundhenk. Nonapproximability results for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:83–103, 2001.

O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 541–548, Orlando, FL, 1999.

N. Meuleau, L. Peshkin, K.-E. Kim, and L. P. Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 427–436, Stockholm, 1999.

G. E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma. Experiences with a mobile robotic guide for the elderly. In *Proceedings of the National Conference on Artificial Intelligence*, pages 587–592, Edmonton, AB, 2002.

A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 406–415, Stanford, CA, 2000.

C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003a.

J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: challenges and results. *Robotics and Autonomous Systems*, 42(3-4):271–281, 2003b.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statististical Association*, 94(446):590–599, 1999.

J. M. Porta and B. J. A. Kröse. Appearance-based concurrent map building and localization using a multi-hypotheses tracker. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 3424–3429, Sendai, Japan, 2004.

J. M. Porta, M. T. J. Spaan, and N. Vlassis. Robot planning in partially observable continuous domains. In *Robotics: Science and Systems I*, pages 217–224, MIT, Cambridge, MA, 2005.

P. Poupart and C. Boutilier. Value-directed compressions of POMDPs. In *Advances in Neural Information Processing Systems 15 (NIPS-2002)*, pages 1547–1554. MIT Press, 2003.

P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16 (NIPS-2003)*, pages 823–830. MIT Press, 2004.

P. Poupart and C. Boutilier. VDCBPI: an approximate scalable algorithm for large POMDPs. In *Advances in Neural Information Processing Systems 17 (NIPS-2004)*, pages 1081–1088. MIT Press, 2005.

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 697–704, 2006.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, Inc., 1994.

N. Roy. *Finding Approximate POMDP Solutions Through Belief Compression*. PhD thesis, Carnegie Mellon University, 2003.

N. Roy and G. Gordon. Exponential family PCA for belief compression in POMDPs. In *Advances in Neural Information Processing Systems 15 (NIPS-2002)*, pages 1635–1642. MIT Press, 2003.

N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

N. Roy, J. Pineau, and S. Thrun. Spoken dialog management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 93–100, Hong Kong, 2000.

Matt Rudary, Satinder Singh, and David Wingate. Predictive linear-gaussian models of stochastic dynamical systems. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 501–508, 2005.

R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 520–527, Banff, Alberta, 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*, 24:195–220, 2005.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

G. Theocharous and S. Mahadevan. Approximate planning with hierarchical partially observable Markov decision processes for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1347–1352, 2002.

S. Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems 12 (NIPS-1999)*, pages 1064–1070. MIT Press, 2000.

N. Vlassis, B. Terwijn, and B.J.A. Kröse. Auxiliary particle filter robot localization from high-dimensional sensor observations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 7–12, 2002.

Steven D. Whitehead and Long-Ji Lin. Reinforcement learning of non-Markov decision processes. *Artificial Intelligence*, 73(1-2):271–306, 1995.

J. Williams, P. Poupart, and S. Young. Using factored Markov decision processes with continuous observations for dialogue management. Technical Report CUED/F-INFEG/TR.520, Cambridge University, Engineering Department, 2005.

B. Zhang, Q. Cai, J. Mao, and B. Guo. Planning and acting under uncertainty: a new model for spoken dialogue systems. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 572–579, Seattle, WA, 2001.

N. L. Zhang and W. Zhang. Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51, 2001.

Journal of Machine Learning Research 7 (2006) 2369-2397 Submitted 2/05; Revised 10/06; Published 11/06

Learning Parts-Based Representations of Data

David A. Ross DROSS@CS.TORONTO.EDU
Richard S. Zemel ZEMEL@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road
Toronto, Ontario M5S 3H5, CANADA

Editor: Pietro Perona

Abstract

Many perceptual models and theories hinge on treating objects as a collection of constituent parts. When applying these approaches to data, a fundamental problem arises: how can we determine what are the parts? We attack this problem using learning, proposing a form of generative latent factor model, in which each data dimension is allowed to select a different factor or part as its explanation. This approach permits a range of variations that posit different models for the appearance of a part. Here we provide the details for two such models: a discrete and a continuous one. Further, we show that this latent factor model can be extended hierarchically to account for correlations between the appearances of different parts. This permits modeling of data consisting of multiple categories, and learning these categories simultaneously with the parts when they are unobserved. Experiments demonstrate the ability to learn parts-based representations, and categories, of facial images and user-preference data.

Keywords: parts, unsupervised learning, latent factor models, collaborative filtering, hierarchical learning

1. Introduction

Many collections of data exhibit a common underlying structure: they consist of a number of parts or factors, each with a range of possible states. When data are represented as vectors, parts manifest themselves as subsets of the data dimensions that take on values in a coordinated fashion. In the domain of digital images, these parts may correspond to the intuitive notion of the component parts of objects, such as the arms, legs, torso, and head of the human body. Prominent theories of computational vision, such as Biederman's *Recognition-by-Components* (Biederman, 1987) advocate the suitability of a parts-based approach for recognition in both humans and machines. Recognizing an object by first recognizing its constituent parts, then validating their geometric configuration has several advantages:

1. Highly articulate objects, such as the human body, are able to appear in a wide range of configurations. It would be difficult to learn a holistic model capturing all of these variants.

2. Objects which are partially occluded can be identified as long as some of their parts are visible.

3. The appearances of certain parts may vary less under a change in pose than the appearance of the whole object. This can result in detectors which, for example, are more robust to rotations of the target object.

4. New examples from an object class may be recognized as simply a novel combination of familiar parts. For example a parts-based face detection system could generalize to detect faces with both beards and sunglasses, having been trained only on faces containing one, but not both, of these features.

The principal difficulty in creating such systems is determining which parts should be used, and identifying examples of these parts in the training data.

In the part-based detectors created by Mohan et al. (2001) and Heisele et al. (2000) parts were chosen by the experimenters based on intuition, and the component-detectors—support vector machines—were trained on image subwindows containing only the part in question. Obtaining these subwindows required that they be manually extracted from hundreds or thousands of training images. In contrast, the parts-based detector created by Weber et al. (2000) proposed a way to automate this process. During training of the geometric model, parts were selected from an initial set of candidates to include only those which lead to the highest detection performance. The resulting detector relied on a very small number of parts (e.g., 3) corresponding to very small local features. Unlike the SVMs, which were trained on a range of appearances of the part, each of these part-detectors could identify only a single fixed appearance.

Parts-based representations of data can also be learned in an entirely unsupervised fashion. These parts can be used for subsequent supervised learning, but the models constructed can also be valuable on their own. A parts-based model provides an efficient, distributed representation, and can aid in the discovery of causal structure within data. For example, a model consisting of K parts, each with J possible states, can represent J^K different objects. Inferring the state of each part can be done efficiently, as each part depends only on a fraction of the data dimensions.

These computational considerations make parts-based models particularly suitable for modeling high-dimensional data such as user preferences in collaborative filtering problems. In this setting, each data vector contains ratings given by a human subject to a number of items, such as books or movies. Typically there are thousands of unique items, but for each user we can only observe ratings for a small fraction of them. The goal in collaborative filtering is to make accurate predictions of the unobserved ratings. Parts can be formed from groups of related items, and the states of a part correspond to different attitudes towards the items. Unsupervised learning of a parts-based model allows us to learn the relationships between items, which allows for efficient online and active learning.

Here we propose a probabilistic generative approach to learning parts-based representations of high-dimensional data. Our key assumption is that the dimensions of the data can be separated into several disjoint subsets, or factors, which take on values independently of each other. Each factor has a range of possible states or appearances, and we investigate two ways of modeling this variability. First we address the case in which each factor has a small number of discrete states, and model it using a *vector quantizer*. In some situations, however, continually-varying descriptions of parts are more suitable. Thus, in our second approach we model part-appearances using *factor analysis*. Given a set of training examples, our approach learns the association of data dimensions with factors, as well as the states of each factor. Inference and learning are carried out efficiently via variational algorithms. The general approach, as well as details for the models, are given in

Section 2. Experiments showing parts-based representations learned for real data sets follow in Section 3.

Although we initially assume that parts take on states independently, clearly in real-world situations there are dependencies. For example, consider the case of modeling images of human faces. A model could be constructed representing the various parts of the face (eyes, nose, mouth, etc.), and the various appearances of each part. If one part were to select an appearance with high pixel intensities, due to lighting conditions or skin tone, then it seems likely that the other parts should appear similarly bright. Realizing this, in Section 4 we propose a method of learning these dependencies between part selections hierarchically, by introducing an additional higher-level cause, or 'class' variable, on which state selections for the factors are conditioned. This allows us to model different categories of data using the same vocabulary of parts, and to induce the categories when they are not available in advance. We conclude with a comparison to related methods, and a final discussion in Sections 5 and 6.

2. An Approach to Learning Parts-Based Models

We approach the problem of learning parts by posing it as a stochastic generative model. We assume that there are K factors, each a probabilistic model for the range of appearances of one of the parts. To generate an observed data vector of D dimensions, $\mathbf{x} \in \Re^D$, we stochastically select one state for each factor, and one factor for each data dimension, x_d. The first selection allows each part to independently choose its appearance, while the second dictates how the parts combine to produce the observed data vector.

This approach differs from a traditional mixture model in that each dimension of the data is generated by a *different* linear combination of the factors. Rather than clustering the data vectors based on which mixture component gives the highest probability, we are grouping the data dimensions based on which part provides the best explanation.

The selection of factors for each dimension are represented as binary latent variables, $\mathbf{R} = \{r_{dk}\}$, for $d = 1...D, k = 1...K$. The variable $r_{dk} = 1$ if and only if factor k has been selected for data dimension d. These variables can be described equivalently as multinomials, $\mathbf{r}_d \in 1...K$, and are drawn according to their respective prior distributions, $\mathrm{P}(r_{dk}) = a_{dk}$. The choice of state for each factor is also a latent variable, which we will represent by $\mathbf{s}_k$. Using θ_k to represent the parameters of the k^{th} factor, we arrive at the following complete likelihood function:

$$\mathrm{P}(\mathbf{x}, \mathbf{R}, \mathbf{S}|\theta) = \prod_{d,k} (a_{dk}{}^{r_{dk}}) \prod_{k} (\mathrm{P}(\mathbf{s}_k)) \prod_{d,k} (\mathrm{P}(x_d \,|\, \theta_k, \mathbf{s}_k)^{r_{dk}}) \,. \tag{1}$$

This probability model is depicted graphically in Figure 1.

As described thus far, the approach is independent of the particular choice of model used for each of the factors. We now provide details for two particular choices: a discrete model of factors, vector quantization; and a continuous model, factor analysis.

2.1 Multiple Cause Vector Quantization

In Multiple Cause Vector Quantization, first proposed in Ross and Zemel (2003), we assume that each part has a small number of appearances, and model them using a vector quantizer (VQ) of J possible states. To generate an observed data example, we stochastically select one state for each

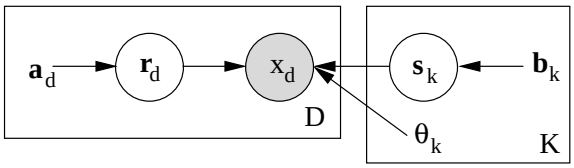


Figure 1: Graphical representation of the parts-based learning model. We let $\mathbf{r}_{d=1}$ represent all the variables $r_{d=1,k}$, which together select a factor for x_1. Similarly, $\mathbf{s}_{k=1}$ selects a state for factor 1. The plates depict repetitions across the D input dimensions and the K factors. To extend this model to multiple data cases, we would include an additional plate over $\mathbf{r}$, x, and $\mathbf{s}$.

VQ, and, as described above, one VQ for each dimension. Given these selections, a single state from a single VQ determines the value of each data dimension x_d.

As before, we represent the selections using binary latent variables, $\mathbf{S} = \{s_{kj}\}$, for $k = 1...K$, $j = 1...J$, where $s_{kj} = 1$ if and only if state j is selected for VQ k. Again we introduce prior selection probabilities $\mathrm{P}(s_{kj}) = b_{kj}$, with $\sum_j b_{kj} = 1$.

Assuming each VQ state specifies the mean as well as the standard deviation of a Gaussian distribution, and the noise in the data dimensions is conditionally independent, we have (where $\theta_k = \{\mu_{dkj}, \sigma_{dkj}\}$, and $\mathcal{N}$ is the Gaussian pdf)

$$\mathrm{P}(\mathbf{x}|\mathbf{R},\mathbf{S},\theta) = \prod_{d,k,j} \mathcal{N}(x_d\,;\,\mu_{dkj},\sigma^2_{dkj})^{r_{dk}\,s_{kj}}.$$

The resulting model can be thought of as a Gaussian mixture model over $J \times K$ possible states for each data dimension (x_d). The single state kj is selected if $s_{kj} r_{dk} = 1$. Note that this selection has two components. The selection in the j component is made jointly for the different data dimensions, and in the k component it is made independently for each dimension.

2.1.1 LEARNING AND INFERENCE

The joint distribution over the observed vector $\mathbf{x}$ and the latent variables is

$$\begin{aligned}\mathrm{P}(\mathbf{x},\mathbf{R},\mathbf{S}|\theta) &= \mathrm{P}(\mathbf{R}|\theta)\mathrm{P}(\mathbf{S}|\theta)\mathrm{P}(\mathbf{x}|\mathbf{R},\mathbf{S},\theta), \\ &= \prod_{d,k}(a_{dk}^{r_{dk}})\prod_{k,j}\left(b_{kj}^{s_{kj}}\right)\prod_{d,k,j}\mathcal{N}(x_d\,;\,\theta_k)^{r_{dk}s_{kj}}.\end{aligned}$$

Given an input $\mathbf{x}$, the posterior distribution over the latent variables, $\mathrm{P}(\mathbf{R},\mathbf{S}|\mathbf{x},\theta)$, cannot tractably be computed, since all the latent variables become dependent.

We apply a variational EM algorithm to learn the parameters θ, and infer latent variables given observations. For a given observation, we could approximate the posterior distribution using a factored distribution, where g and m are variational parameters related to r and s respectively:

$$Q_0(\mathbf{R},\mathbf{S}|\mathbf{x},\theta) = \prod_{d,k}(g_{dk}^{r_{dk}})\prod_{k,j}\left(m_{kj}^{s_{kj}}\right). \tag{2}$$

The model is learned by optimizing the following objective function (Neal and Hinton, 1998), also known as the variational free energy:

$$\begin{aligned}\mathcal{F}(Q_0,\theta) &= E_{Q_0}\left[\log P(\mathbf{x},\mathbf{R},\mathbf{S}|\theta) - \log Q_0(\mathbf{R},\mathbf{S}|\mathbf{x},\theta_k)\right], \\ &= E_{Q_0}\left[-\sum_{d,k} r_{dk}\log(g_{dk}/a_{dk}) - \sum_{k,j} s_{kj}\log(m_{kj}/b_{kj}) + \sum_{d,k,j} r_{dk}s_{kj}\log\mathcal{N}(x_d;\theta)\right], \\ &= -\sum_{d,k} g_{dk}\log\frac{g_{dk}}{a_{dk}} - \sum_{k,j} m_{kj}\log\frac{m_{kj}}{b_{kj}} - \sum_{d,k,j} g_{dk}\,m_{kj}\,\varepsilon_{dkj},\end{aligned}$$

where $\varepsilon_{dkj} = \log\sigma_{dkj} + \frac{(x_d-\mu_{dkj})^2}{2\sigma_{dkj}^2}$. The objective function $\mathcal{F}$ is a lower bound on the log likelihood of generating the observations, given the particular model parameters. The variational EM algorithm improves this bound by iteratively maximizing $\mathcal{F}$ with respect to Q_0 (E-step) and to θ (M-step).

Extending this to handle multiple observations—the columns of $\mathbf{X} = [\mathbf{x}^1 \ldots \mathbf{x}^C]$—we must now consider approximating the posterior $\mathrm{P}(\mathcal{R},\mathcal{S}|\mathbf{X},\theta)$, where $\mathcal{R} = \{r_{dk}^c\}$ and $\mathcal{S} = \{s_{kj}^c\}$ are the latent selections for all training cases c. Our aim is to learn models which have a posterior distribution over factor selections for each data dimension that is consistent across all data (that is to say, regardless of the data case, each x_d will typically be associated with the same part or parts). Thus, in the variational posterior we constrain the parameters $\{g_{dk}\}$ to be the same across all observations $\mathbf{x}^c, c = 1 \ldots C$. In this general case, the variational approximation to the posterior becomes (cf. Equation (2))

$$Q(\mathcal{R},\mathcal{S}|\mathbf{X},\theta) = \prod_{c,d,k}\left(g_{dk}{}^{r_{dk}^c}\right)\prod_{c,k,j}\left(m_{kj}^{c}{}^{s_{kj}^c}\right). \tag{3}$$

It is important to point out that this choice of variational approximation is somewhat unorthodox, nonetheless it is perfectly valid and has produced good results in practice. A comparison to more conventional alternatives appears in Appendix A.

Under this formulation, only the $\{m_{kj}^c\}$ parameters are updated during the E step for each observation $\mathbf{x}^c$:

$$m_{kj}^c = b_{kj}\exp\Big(-\sum_d g_{dk}\,\varepsilon_{dkj}^c\Big) \Big/ \sum_{\rho=1}^{J} b_{k\rho}\exp\Big(-\sum_d g_{dk}\,\varepsilon_{d\rho k}^c\Big).$$

The M step updates the parameters, μ and σ, which relate each latent state kj to each input dimension d, the parameters of Q related to factor selection $\{g_{dk}\}$, and the priors $\{a_{dk}\}$ and $\{b_{kj}\}$:

$$g_{dk} = a_{dk}\exp\Big(-\frac{1}{C}\sum_{c,j} m_{kj}^c\,\varepsilon_{dkj}^c\Big) \Big/ \sum_{\rho=1}^{K} a_{d\rho}\exp\Big(-\frac{1}{C}\sum_{c,j} m_{j\rho}^c\,\varepsilon_{dj\rho}^c\Big), \tag{4}$$

$$\mu_{dkj} = \frac{\sum_c m_{kj}^c x_d^c}{\sum_c m_{kj}^c}, \qquad \sigma_{dkj}^2 = \frac{\sum_c m_{kj}^c (x_d^c - \mu_{dkj})^2}{\sum_c m_{kj}^c},$$

$$a_{dk} = g_{dk}, \qquad b_{kj} = \frac{1}{C}\sum_c m_{kj}^c.$$

As can be seen from the update equations, an iteration of EM learning for MCVQ has computational complexity linear in each of C, D, J, and K.

A variational approximation is just one of a number of possible approaches to performing the intractable inference (E) step in MCVQ. One alternative, known as *Monte Carlo EM*, is to approximate the posterior with a set of samples $\{\mathcal{R}_n, \mathcal{S}_n\}_{n=1}^N$ drawn from the true posterior $\mathrm{P}(\mathcal{R}, \mathcal{S}|\mathbf{X}, \theta)$ via Gibbs sampling. The M step now becomes a maximization of the approximate likelihood $\frac{1}{N}\sum_n \mathrm{P}(\mathbf{X}, \mathcal{R}_n, \mathcal{S}_n|\theta)$ with respect to the model parameters θ. Note that the intractable marginalization over $\{r_{dk}^c, s_{kj}^c\}$ has been replaced with the less-costly sum over N samples. Details of this approach for a related model can be found in Ghahramani (1995), and a general description in Andrieu et al. (2003).

2.2 Multiple Cause Factor Analysis

In MCVQ, each factor is modeled as a set of basis vectors, one of which is chosen at a time when generating data. A more general approach is to allow data to be generated by arbitrary linear combinations of the basis vectors in each factor. This extension (with the appropriate choice of prior distribution) amounts to modeling the range of appearances for each part with a factor analyzer.

A factor analysis (FA) model (e.g., Ghahramani and Hinton, 1996) proposes that the data vectors come from a low-dimensional linear subspace, represented by the basis vectors of the *factor loading matrix*, $\Lambda \in \Re^{D\times J}$, and an offset $\boldsymbol{\rho} \in \Re^D$ from the origin. A data vector is produced by taking a linear combination $\mathbf{s} \in \Re^J$ of the columns of Λ. The linear combination is treated as an unobserved latent variable, with a standard Gaussian prior $\mathrm{P}(\mathbf{s}) = \mathcal{N}(\mathbf{s}; \mathbf{0}, \mathbf{I})$. To this is added zero-mean Gaussian noise, independent along each dimension. The probability model is

$$\mathrm{P}(\mathbf{x}, \mathbf{s}|\theta) = \mathcal{N}(\mathbf{x}; \Lambda\mathbf{s} + \boldsymbol{\rho}, \Psi)\ \mathcal{N}(\mathbf{s}; \mathbf{0}, \mathbf{I}) \tag{5}$$

where Ψ is a $D \times D$ diagonal covariance matrix.

As with MCVQ, we assume that the data contains K parts and model each using a factor analyzer $\theta_k = (\Lambda^k, \boldsymbol{\rho}_k, \Psi^k)$. A data vector is again generated by stochastically selecting one state $\mathbf{s}_k$ per factor k, and choosing one factor per data dimension. Under this model factor analyzer k proposes that the value of x_d has a Gaussian distribution centered at $\Lambda_d^k \mathbf{s}_k + \rho_{dk}$ with variance Ψ_d^k (where Λ_d^k indicates the d^{th} *row* of factor loading matrix k, and Ψ_d^k the d^{th} entry on the diagonal of Ψ^k). Thus the likelihood is

$$\mathrm{P}(\mathbf{x}|\mathbf{R}, \mathbf{S}, \theta) = \prod_{d,k} \mathcal{N}(x_d; \Lambda_d^k \mathbf{s}_k + \rho_{dk}, \Psi_d^k)^{r_{dk}}.$$

2.2.1 LEARNING AND INFERENCE

Again this model produces an intractable posterior over latent variables $\mathbf{S}$ and $\mathbf{R}$, so we resort to a variational approximation:

$$Q(\mathcal{R}, \mathcal{S}|\mathbf{X}, \theta) = \prod_{c,d,k} \left(g_{dk}^{r_{dk}^c}\right) \prod_{c,k} \mathcal{N}(\mathbf{s}_k^c; \mathbf{m}_k^c, \Omega_{ck}).$$

This differs from the approximation used for MCVQ, Equation (3), in that here we assume the state variables $\mathbf{s}_k$ have Gaussian posteriors. Thus, in the E step we must now estimate the first *and* second moments of the posterior over $\mathbf{s}_k$. As before, we also tie the $\{g_{dk}\}$ parameters to be the same across all data cases. Setting up the objective function and differentiating, gives us the following updates

for the E-step:[1]

$$\mathbf{m}_k^c = \Omega_{ck} \Lambda^{k^T} \Psi_k^{-1} ((\mathbf{x}^c - \boldsymbol{\rho}_k) .* \mathbf{g}_k),$$

$$\Omega_{ck} = \left(\Lambda^{k^T} \frac{\mathbf{g}_k}{\Psi^k} \Lambda^k + \mathbf{I} \right)^{-1}, \qquad \langle \mathbf{s}_k^c \mathbf{s}_k^{c^T} \rangle = \Omega_{ck} + \mathbf{m}_k^c \mathbf{m}_k^{c^T},$$

where $\frac{\mathbf{g}_k}{\Psi^k}$ is a diagonal matrix with entries g_{dk}/Ψ_d^k. Note that the expression for Ω_{ck} is independent of the index over training cases, c, thus we need only have one $\Omega_k = \Omega_{ck}, \forall c$.

The M-step involves updating the prior and variational posterior factor-selection parameters, $\{a_{dk}\}$ and $\{g_{dk}\}$, as well as the parameters $(\Lambda^k, \boldsymbol{\rho}_k, \Psi^k)$ of each factor analyzer. At each iteration the prior, $a_{dk} = g_{dk}$ is set to the posterior at the previous iteration. The remaining updates are

$$g_{dk} \propto \frac{a_{dk}}{|\Psi_d^k|^{1/2}} \exp \left[\frac{-1}{2C\Psi_d^k} \sum_c \left((x_d^c - \rho_{dk})^2 + \Lambda_d^k \langle \mathbf{s}_k^c \mathbf{s}_k^{c^T} \rangle \Lambda_d^{k^T} - 2(x_d^c - \rho_{dk}) \Lambda_d^k \mathbf{m}_k^c \right) \right],$$

$$\Lambda^k = \left(\mathbf{X} - \boldsymbol{\rho}_k \mathbf{1}^T \right) \mathbf{M}_k^T \left(\sum_c \langle \mathbf{s}_k^c \mathbf{s}_k^{c^T} \rangle \right)^{-1},$$

$$\boldsymbol{\rho}_k = \frac{1}{C} \sum_c \left(\mathbf{x}^c - \Lambda^k \mathbf{m}_k^c \right),$$

$$\Psi_d^k = \frac{1}{C} \sum_c \left((x_d^c - \rho_{dk})^2 + \Lambda_d^k \langle \mathbf{s}_k^c \mathbf{s}_k^{c^T} \rangle \Lambda_d^{k^T} - 2(x_d^c - \rho_{dk}) \Lambda_d^k \mathbf{m}_k^c \right).$$

where ($\mathbf{M}_k$ is a $J \times C$ matrix in which the c^{th} column is $\mathbf{m}_k^c$).

2.3 Related Algorithms

Here we present the details of two related algorithms, principal components analysis (PCA), and non-negative matrix factorization (NMF). A more detailed comparison of these algorithms with MCVQ and MCFA appears below, in Section 5.

The goal of PCA is to learn a factorization of the data matrix $\mathbf{X}$ into the product of a coefficient matrix $\mathbf{S}$ and an orthogonal basis Λ so that $\mathbf{X} \approx \Lambda \mathbf{S}$. Typically Λ has fewer columns than rows, so $\mathbf{S}$ can be thought of as a reduced-dimensionality approximation of $\mathbf{X}$, and Λ as the key features of the data. Using a squared-error cost function, the optimal solution is to let Λ be the top eigenvectors of the sample covariance matrix, $\frac{1}{C-1}\mathbf{X}\mathbf{X}^T$, and let $\mathbf{S} = \Lambda^T \mathbf{X}$.

PCA can also be posed as a probabilistic generative model, closely related to factor analysis (Roweis, 1997; Tipping and Bishop, 1999). In fact, probabilistic PCA proposes the same generative model, Equation (5), except that the noise covariance, Ψ, is restricted to be a scalar times the identity matrix: $\Psi = \sigma^2 \mathbf{I}$.

Non-negative matrix factorization (Lee and Seung, 1999) also learns a factorization $\mathbf{X} \approx \mathbf{W}\mathbf{H}$ of the data matrix, but includes the additional restriction that $\mathbf{X}$, $\mathbf{W}$, and $\mathbf{H}$ must be entirely non-negative. By allowing only additive combinations of a non-negative basis, Lee & Seung propose to obtain basis vectors that correspond to the intuitive parts of the data. Instead of squared error, NMF seeks to minimize the divergence $D(\mathbf{X} \| \mathbf{W}\mathbf{H}) = \sum_{d,c} \left[x_d^c \log(\mathbf{W}\mathbf{H})_{dc} - (\mathbf{W}\mathbf{H})_{dc} \right]$ which is the

1. The second uncentered moment $\langle \mathbf{s}_k^c \mathbf{s}_k^{c^T} \rangle$ need not be computed explicitly, since it can be expressed as a combination of the first and second centered moments, $\mathbf{m}_k$ and Ω_{ck}. It is, however, a useful subexpression for computing the M-step updates and likelihood bound.

negative log-probability of the data, assuming a Poisson density function with mean $\mathbf{WH}$. A local minimum of the divergence is obtained by iterating the following multiplicative updates:

$$w_{dj} \leftarrow w_{dj} \sum_c \frac{x_d^c}{(\mathbf{WH})_{dc}} h_{jc}, \qquad w_{dj} \leftarrow \frac{w_{dj}}{\sum_{d'} w_{d'j}}, \qquad h_{jc} \leftarrow h_{jc} \sum_d w_{dj} \frac{x_d^c}{(\mathbf{WH})_{dc}}.$$

Despite the probabilistic interpretation, $\mathbf{X} \sim \mathit{Poisson}(\mathbf{WH})$, NMF is not a proper probabilistic generative model, since it does not specify a prior distribution over the latent variable $\mathbf{H}$. Thus NMF does not specify how new data could be generated from a learned basis.

Like the above methods, MCVQ and MCFA can be viewed as matrix factorization methods, where the left-hand matrix is formed from the $\{g_{dk}\}$ and the collection of VQ/FA basis vectors, while the right-hand matrix is comprised of the latent variables $\{m_{kj}^c\}$. This view highlights the key contrast between the assumptions about the data embodied in these earlier models, as opposed to the proposed model. Here the data is viewed as a concatenation of components or parts, corresponding to particular subsets of data dimensions, each of which is modeled as a convex combination of appearances.

3. Experiments

In this section we examine the ability of MCVQ and MCFA to learn parts-based representations of data from two different problem domains.

We begin by modeling sets of digital images, in this case images of human faces. The parts learned are fixed subsets of the data dimensions (pixels), corresponding to fixed regions of the images, that closely resemble intuitive notions of the parts of faces. The ability to learn parts is robust to partial occlusions in the training images.

The application of MCVQ and MCFA to image data assumes that the images are normalized, that is, that the head is in a similar pose in each image, and aligned with respect to position and scale. This constraint is standard for learning methods that attempt to learn visual features beyond low level edges and corners, though, if desired, the model could be extended to perform automatic alignment of images (Jojic and Caspi, 2004). While normalization may require a preprocessing step for image applications, in many other types of applications, the input representation is more stable. For instance, in collaborative filtering each data vector consists of a single user's ratings for a fixed set of items; each data dimension always corresponds to the same item.

Thus, we also explore the application of MCVQ and MCFA to the problem of predicting ratings of unseen movies, given observed ratings for a large set of users. Here parts correspond to subsets of the movies which have correlated ratings.

Code implementing MCVQ and MCFA in MATLAB, as used for the following experiments, can be obtained at http://www.cs.toronto.edu/~dross/mcvq/.

3.1 Face Images

The face data set consisted of 2429 images from the CBCL Face database #1 (MIT-CBCL, 2000). Each image contained a single frontal or near-frontal face, depicted in 19×19 pixel grayscale. The images were histogram equalized, and pixel values rescaled to lie in $[-2, 2]$. Sample training images are shown in Figure 2.

Using these images as input, we trained MCVQ and MCFA models, each containing $K = 6$ factors. The MCVQ model with $J = 10$ states converged in 120 iterations of Monte Carlo EM, while

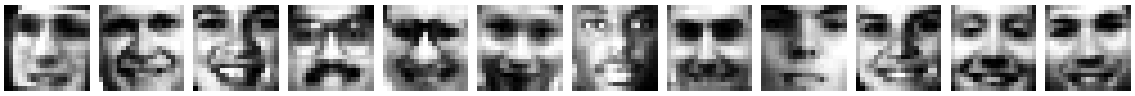


Figure 2: Sample training images from the CBCL Face database #1.

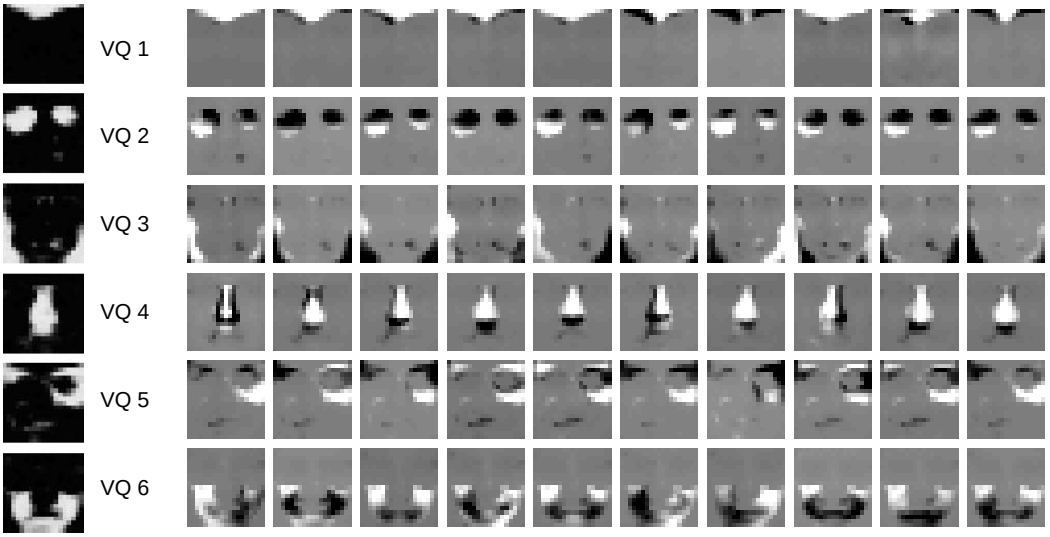


Figure 3: The parts-based model of faces learned by MCVQ. On the left are plots showing the posterior probability with which each pixel selects the indicated VQ. On the right are the means, for each state of each VQ, masked by the aforementioned selection probabilities $(\boldsymbol{\mu}_{kj} .* \mathbf{g}_k)$.

the MCFA model with $J = 4$ basis vectors converged in 15 variational EM iterations. In practice, the Monte Carlo approach to inference leads to better local maxima of the objective function, and better parts-based models of the data. For both MCVQ and MCFA, prior probabilities were initialized to uniform, and states/basis vectors to randomly selected training images. The learned parts-based decompositions are depicted in Figures 3 and 4.

On the left of each figure is a plot of posterior probabilities $\{g_{dk}\}$ that each pixel selects the indicated factor as its explanation. White indicates high probability of selection, and black low. As can be seen, each $\mathbf{g}_k$ can be thought of as a mask indicating which pixels 'belong' to factor k.

In the noise-free case, each image generated by one of these models is a sum of the contributions from the various factors, where each contribution is 'masked' by the probability of pixel selection. For example in Figure 3 the probability of selecting VQ #4 is non-zero only around the nose, thus the contribution of this VQ to the remaining areas of the face in any generated image is negligible.

On the right of each figure is a plot of the 10 states/4 basis vectors for each factor. Each image has been masked (via element-wise multiplication) with the corresponding $\mathbf{g}_k$. For MCVQ this is $(\mu_{dkj} .* g_{dk})$, and for MCFA this is $(\Lambda^k_{dj} .* g_{dk})$. In the case of MCVQ, each state is an alternative appearance for the corresponding part. For example VQ #4 gives 10 alternative noses to select from

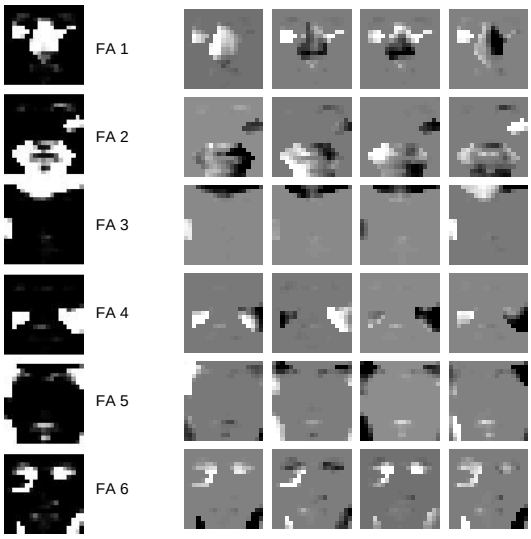


Figure 4: The parts-based model of faces learned by MCFA.

when generating a face. These range from thin to wide, with shadows of the left or right, and with light or dark upper-lips (possibly corresponding to moustaches). In MCFA, on the other hand, the basis vectors are not discrete alternatives. Rather these vectors are combined via arbitrary linear combinations to generate an appearance of a part.

To test the fidelity of the learned representations, the trained models were used to probabilistically classify held-out test images as either *face* or *non-face*. The test data consisted of the 472 face images and 472 randomly-selected non-face images from the CBCL database test set. To classify test images, we evaluated their probability under the model, labeling an image as *face* if its probability exceeded a threshold, and *non-face* if it did not. For each model the threshold was chosen to maximize classification performance. Evaluating the probability of a data vector under MCVQ and MCFA can be difficult, since it requires marginalizing over all possible values of the latent variables. Thus, in practice, we use a Monte Carlo approximation, obtained by averaging over a sample of possible selections. The results of this experiment are shown in Table 1. In addition to MCVQ and MCFA, we performed the same experiment with four other probabilistic models: probabilistic principal components analysis (PPCA), mixture of Gaussians, a single Gaussian distribution, and cooperative vector quantization (Hinton and Zemel, 1994) (which is described further in Section 5). It is important to note that in this experiment an increase in model size (using more VQs, states, or basis vectors) does not necessarily improve the ability to discriminate faces from non-faces. For example, PPCA achieves its highest accuracy when using only three basis vectors, and a mixture of Gaussians with 60 states outperforms one with 84 states. As can be seen, the highest performance is achieved by an MCVQ model with 6 VQs and 14 states per VQ.

Another way of validating image models it to use them to generate new examples and see how closely they resemble images from the observed data—in this case how much the generated images resemble actual faces. Examples of images generated from MCVQ and MCFA are shown in Figure 5.

Model	Accuracy
MCVQ (6 VQs, 14 states each)	**0.8305**
Mixture of Gaussians (60 states)	0.8072
MCVQ (6 VQs, 10 states each)	0.8030
Probabilistic PCA (3 components)	0.7903
Gaussian distribution (diagonal covariance)	0.7871
Mixture of Gaussians (84 states)	0.7680
MCFA (6 FAs, 3 basis vectors each)	0.7415
MCFA (6 FAs, 4 basis vectors each)	0.7267
Cooperative Vector Quantization (6 VQs, 10 states each)	0.6208

Table 1: Results of classifying test images as *face* or *non-face*, by computing their probabilities under trained generative models of faces.

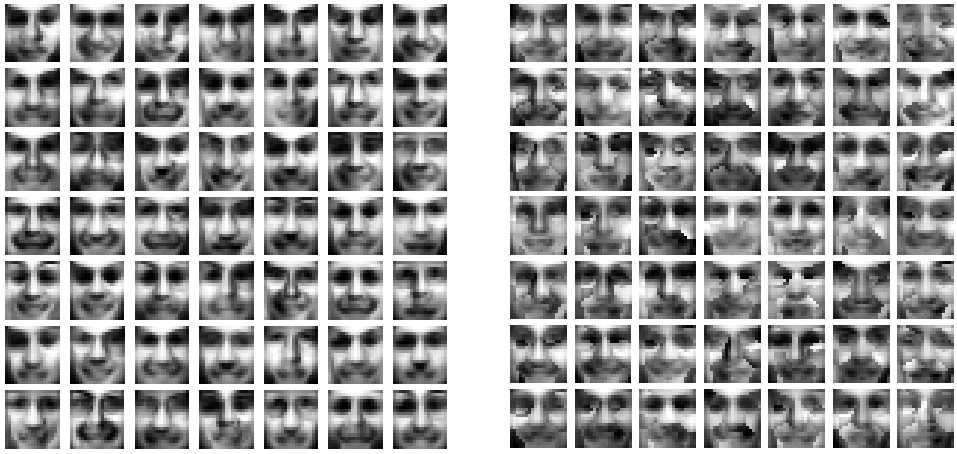


Figure 5: Synthetic images generated using MCVQ (left) and MCFA (right).

The parts-based models learned by MCVQ and MCFA differ from those learned by NMF and PCA, as depicted in Figure 6, in two important ways. First, the basis is sparse—each factor contributes to only a limited region of the image, whereas in NMF and PCA basis vectors include more global effects. Secondly MCVQ and MCFA learn a grouping of vectors into related parts, by explicitly modeling the sparsity via the g_{dk} distributions.

A further point of comparison is that, in images generated by PPCA and NMF, a significant proportion of the pixels in the generated images lie outside the range of values appearing in the training data (7% for PPCA and 4.5% for NMF), requiring that the generated values be thresholded. On the other hand, MCVQ will never generate pixel values outside the observed range. This is a simple result, since each generated image is a convex combination of basis vectors, and each basis vector is a convex combination of data vectors. Although no such guarantee exists for MCFA, in practice pixels it generates are all within-range.

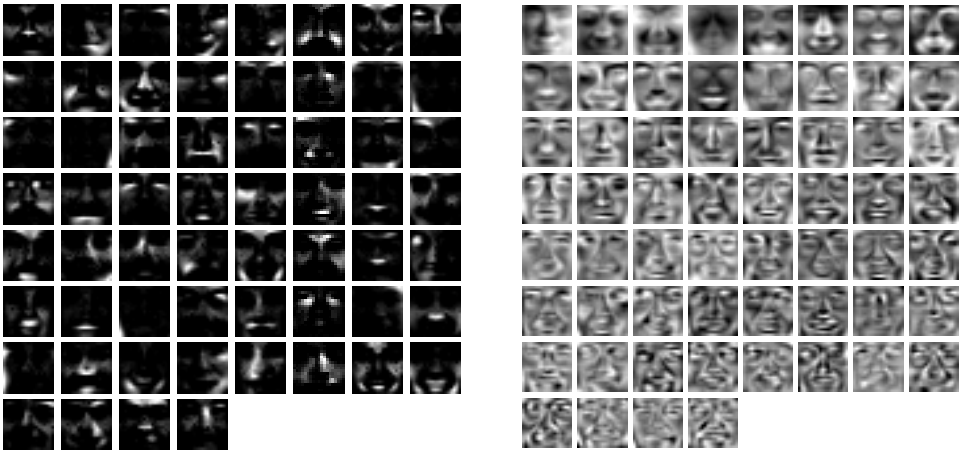


Figure 6: Bases learned by non-negative matrix factorization (left) and probabilistic principal components analysis (right), when trained on the face images.

3.1.1 PARTIAL OCCLUSION

Parts-based models can also be reliably learned from image data in which the object is partially occluded. As an illustration, we trained an MCVQ model on a data set containing partially-occluded faces. The occlusions were generated by replacing pixels in randomly-selected contiguous regions of each training image with noise. The noise, which covered $\frac{1}{4}$ to $\frac{1}{2}$ of each image, had the same first- and second-order statistics as the unoccluded pixels. The resulting training set contained 4858 images, half of which were partially occluded.

Examples of the training images, and the resulting model, can be seen in Figure 7. In the model each VQ has learned at least one state containing a blurry region, which corresponds to an occluded view of the respective part (e.g., for VQ 1, the second state from the right). Thus this model is able to generate and reconstruct (recognize) partially-occluded faces.

3.2 Collaborative Filtering

Here we test MCVQ on a collaborative filtering task, using the EachMovie data set, where the input vectors are ratings by viewers of movies, and a given element always corresponds to the same movie. The original data set contains ratings, on a scale from 1 to 6, of a set of 1649 movies, by 74,424 viewers. In order to reduce the sparseness of the data set, since many viewers rated only a few movies, we only included viewers who rated at least 20 movies, and removed unrated movies. The remaining data set, containing 1623 movies and 36,656 viewers, was still very sparse (95.7%).

We evaluated the performance of MCVQ using the framework proposed by Marlin (2004) for comparison of collaborative filtering algorithms. From each viewer in the EachMovie set, a single randomly-chosen rating was held out. The performance of the model at predicting these held-out ratings was evaluated using 3-fold cross validation. In each fold, the model was fit to a training set

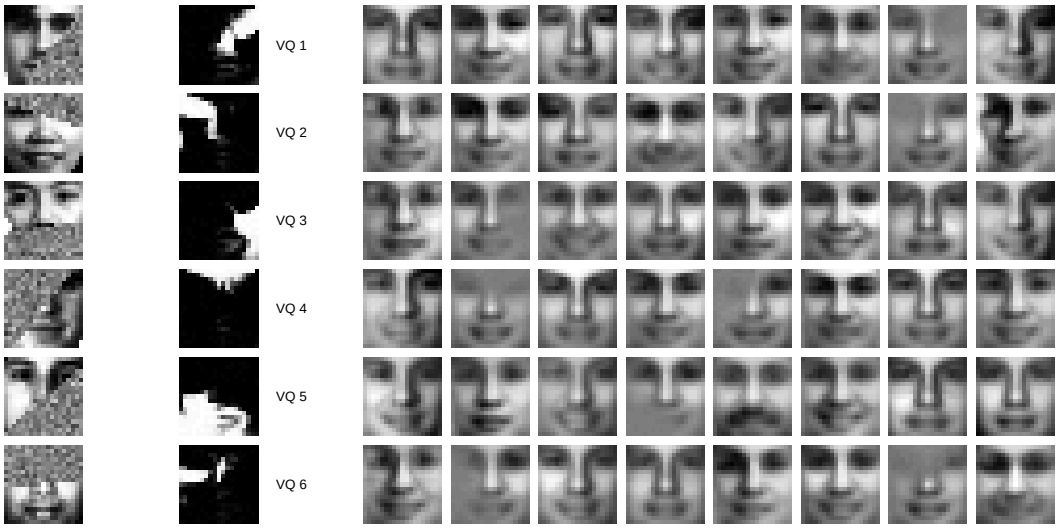


Figure 7: The model learned by MCVQ on partially-occluded faces. On the left are six representative images from the training data. In the middle are plots of the posterior part selection probability (the g_{dk}'s) for each VQ. On the right are the (unmasked) means for each state of each VQ. Note that each VQ has learned at least one state to represent partial occlusion of the corresponding part.

of 2/3 of the viewers, with the remaining 1/3 viewers retained as a testing set.[2] This model was used to predict the held-out ratings of the training viewers, a task referred to as *weak generalization*, as well as the held-out ratings of the testing viewers, called *strong generalization*. Each prediction was obtained by first inferring the latent states of the factors m_{kj}, then averaging the mean of each state by its probability of selection: $\sum_{k,j} g_{dk}\, m_{kj}\, \mu_{kjd}$.

Prediction performance was calculated using normalized mean absolute error (NMAE). Given a set of predictions $\{p_i\}$, and the corresponding ratings $\{r_i\}$, the NMAE is

$$\frac{1}{zn}\sum_{i=1}^{n}|r_i - p_i|$$

where z is a normalizing factor equal to the expectation of the mean absolute error, assuming uniformly distributed p_i's and r_i's (Marlin, 2004).[3] For the EachMovie data set this value is $z = 35/18 \approx 1.9444$.

We trained several models on this data set, including MCVQ, MCFA, factor analysis, mixture of Gaussians, as well as a baseline algorithm which, for each movie, simply predicts the mean of

2. This differs slightly from Marlin's approach. He used three randomly sampled training and test sets of 30000 and 5000 viewers respectively. However, his test sets were not disjoint, introducing a potential bias in the error estimate. We, instead, partitioned the data into thirds, using 2/3 of the viewers for training and 1/3 for testing in each of the three folds. The resulting estimates of error should be less biased, but remain directly comparable to Marlin's results.
3. For data sets in which the ratings range over the integers, from a minimum of m to a maximum of M, there is a simple expression for z. Let $N = M - m$ be the largest possible absolute error. Then $z = \frac{N(N+2)}{3(N+1)}$.

Model	Weak Generalization	Strong Generalization
MCVQ $K = 5, J = 12$	0.4896	0.4942
Gaussian mixture $J = 60$	0.4945	0.4895
Factor Analysis $J = 60$	0.5314	0.5318
MCFA $K = 4, J = 6$	0.5502	0.5503
Predict mean observed rating	0.5621	0.5621

Table 2: Collaborative filtering performance. For each algorithm we show the prediction error, in terms of NMAE (see text), on held out ratings from the training data (weak generalization) and the test data (strong generalization). K and J give the number of factors and basis vectors per factor respectively.

the observed ratings for that movie. MCVQ and MCFA were both trained using variational EM, converging in 15 iterations. We selected this approach due to its efficiency, as running each of these algorithms on the large data sets used in these experiments via the Monte Carlo EM approach would require considerable computation time. Note that in the graphical model (Fig. 1), all the observation dimensions are leaves, so a data variable whose value is not specified in a particular observation vector will not play a role in inference or learning. This makes inference and learning with sparse data rapid and efficient in MCVQ and MCFA.

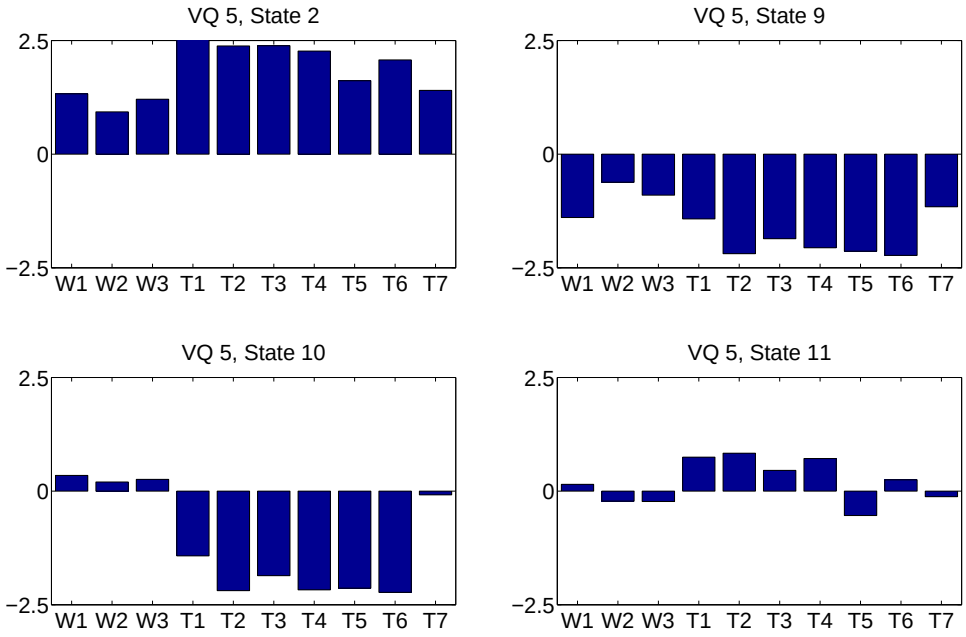
The results of rating prediction are summarized in Table 2 For each model, we have included the best performance, under the restriction that $JK \leq 60$. For comparison, the current state-of-the-art performance on this experiment is 0.4422 weak and 0.4557 strong generalization, by Marlin's User Rating Profile model (Marlin, 2004).

In addition to its utility for rating prediction, the parts-based model learned by MCVQ can also provide key insights regarding the relationships between the movies. These relationships are largely captured by the learned $\{g_{dk}\}$ parameters. For a movie d, the probability distribution $\mathbf{g}_d = [g_{d1} \ldots g_{dK}]$ indicates the affinity of the movie for each of the K factors. To study these relationships we retrained MCVQ on the EachMovie set, using all available data, again with $K = 5$ and $J = 12$.

When trained on the full data set, the average entropy of the $\mathbf{g}_d$ distributions was only 0.0475 bits. In other words, for 98% of the movies, at least one factor was consistently selected with a posterior probability exceeding 0.9. The movies in the training set were distributed fairly evenly amongst the available factors. Specifically, using a probability of $g_{dk} > 0.9$ to indicate strong association, VQs 1 to 5 were assigned 14%, 33%, 9%, 20% and 22% respectively of the movies. Thus MCVQ learned a partitioning of the movies into approximately disjoint subsets.

In the parts-based model, movies with related ratings were associated with the same factor. For example, all seven movies from the *Amityville Horror* series were assigned to VQ #1. Similarly the three original *Star Wars* movies, as well as seven of the eight *Star Trek* movies were all associated with VQ #5. By examining VQ #5 in more detail we can see that each state of the VQ corresponds to a different attitude towards the associated movies, or 'ratings profile'. In Figure 8 we compare the ratings given to the *Star Wars* and *Star Trek* movies by various states of VQ #5. For each state we have plotted the difference between the predicted rating and the mean rating, $\mu_{dkj} - \sum_{j'}(\mu_{dkj'}/J)$, for each of the 10 movies. This score will be large for a particular movie if the state, or profile, gives the movie a much higher rating than it ordinarily receives.

The four states depicted in Figure 8 show a range of possible ratings profiles. State 2 shows a strong preference for all the movies, with each receiving a rating 1-2.5 points higher than usual. In contrast, state 9 shows an equally strong dislike for the movies. State 10 indicates a viewer who is fond of *Star Wars*, but not *Star Trek*. Finally, state 11 shows an ambivalent attitude: slight preference for some films, and slight dislike for others.



Star Wars	W1	Star Trek III: The Search for Spock	T3
The Empire Strikes Back	W2	Star Trek IV: The Voyage Home	T4
Return of the Jedi	W3	Star Trek V: The Final Frontier	T5
Star Trek: The Motion Picture	T1	Star Trek VI: The Undiscovered Country	T6
Star Trek II: The Wrath of Khan	T2	Star Trek: First Contact	T7

Figure 8: Ratings profiles learned by MCVQ on the EachMovie data, for *Star Wars* and *Star Trek* movies. Each state represents a different attitude towards the movies. Positive scores correspond to above-average ratings for the movie, and negative to below-average.

3.2.1 ACTIVE LEARNING

When collaborative filtering is posed in an online or active setting, the set of available data is continually growing, as viewers are queried and new ratings are observed. When a viewer provides a new rating for an item, the system's beliefs about his preferences must be updated. In generative models, such as we have described, this entails updating the distribution over latent state variables for the viewer. This update is often costly, and can pose a problem when the system must be used interactively. Furthermore, queries of the sort "What is the rating of x_d?" must be generated online, based on the viewer's previous responses, to maximize the value of the information obtained. This

value could simply be the system's certainty as to the viewer's preferences, or, more interestingly, its ability to recommend movies that he might enjoy.

The parts based models we describe significantly simplify both of these computations. By learning a partitioning of the items, a new rating will only affect beliefs about unobserved ratings associated with the same factor. Specifically, only one m_{kj} will be updated for each new rating, and the update will depend only on a fraction of the observations.

Also, since we explicitly model the relationships between items, we can determine how the possible responses to a putative query would affect predictions. Thus, we can formulate a query designed to identify movies with high predicted ratings. More details on a successful application of MCVQ to active collaborative filtering, using these ideas, can be found in Boutilier et al. (2003).

4. Hierarchical Learning

Thus far it has been assumed that the states of each factor are selected independently. Although this assumption has proven useful, it is unrealistic to suppose that, for example, the appearance of the eyes and mouth in a face are entirely uncorrelated. Rather than simply being a violation of our modeling assumptions, these correlations can be viewed as an additional source of information from which we can learn higher-order structure present in the data.

Here we relax the assumption of independence by including an additional higher-level latent variable upon which state selections are conditioned. This variable has two possible interpretations. First, if the higher-level cause is unobserved, it can be learned, inducing a categorization of the data vectors based on their state selections. Secondly, if the variable is observed, it can be treated as side information available during learning. In this case the prior over state selections will be adapted to account for the side information.

We now develop both approaches, and demonstrate their use on a set of images containing different facial expressions.

4.1 Extending the Generative Model

Suppose that each data vector comes from one of N different classes. We assume that for a given data vector the selections of the states for each factor (the $\mathbf{s}_k$'s) depend on the class, but that the selections of a factor per data dimension (the $\mathbf{r}_d$'s) do not. Specifically, for training case $\mathbf{x}$, we introduce a new multinomial variable $\mathbf{y}$ which selects exactly one of the N classes. $\mathbf{y}$ can be thought of as an indicator vector $\mathbf{y} \in \{0,1\}^N$, where $y_n = 1$ if and only if class n has been selected. Using a prior $\mathrm{P}(y_n = 1) = \beta_n$ over $\mathbf{y}$, the complete likelihood takes the following form (cf. Equation (1)):

$$\begin{aligned} \mathrm{P}(\mathbf{x},\mathbf{R},\mathbf{S},\mathbf{y}|\theta) &= \mathrm{P}(\mathbf{x}|\mathbf{R},\mathbf{S},\theta)\mathrm{P}(\mathbf{R})\mathrm{P}(\mathbf{S}|\mathbf{y})\mathrm{P}(\mathbf{y}), \\ &= \prod_{d,k} (\mathrm{P}(x_d|\theta_k,\mathbf{s}_k)^{r_{dk}}) \prod_{d,k} \left(a_{dk}^{r_{dk}}\right) \prod_{n,k,j} \left(b_{nkj}^{s_{kj}y_n}\right) \prod_n (\beta_n^{y_n}). \end{aligned} \tag{6}$$

The graphical model representation is given in Figure 9.

Note that for each class n there is a different prior distribution over the state selections. We represent these distributions with $\{b_{nkj}\}_{nkj}$, $\sum_j b_{nkj} = 1$, where b_{nkj} is the probability of selecting state j from factor k given class n. Since the model contains N priors over $\mathbf{S}$, one to be selected for each data vector, then we can think of this model as incorporating an additional vector quantization or clustering, this time over distributions for $\mathbf{S}$.

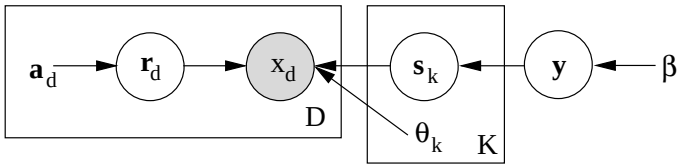


Figure 9: Graphical model with additional class variable, **y**. Note that **y** can be either observed or unobserved.

If the class variable **y** corresponds to an observed label for each data vector, then the new model can be thought of as a standard MCVQ or MCFA model, but with the requirement that we learn a different prior over **S** for each class. On the other hand, if **y** is unobserved, then the new model learns a mixture over the space of possible priors for **S**, rather than the single maximum likelihood point estimate learned in standard MCVQ and MCFA.

Below we derive the EM updates for the hierarchical MCVQ model. The derivation of hierarchical MCFA is similar.

4.2 Unsupervised Case

In the unsupervised case, $\mathbf{y}^c$, for each training case $c = 1 \ldots C$ is an unobserved variable, like $\mathbf{R}^c$ and $\mathbf{S}^c$. As in standard MCVQ, the posterior $\mathrm{P}(\mathbf{R}, \mathbf{S}, \mathbf{y} | \mathbf{x}, \theta)$ over latent variables cannot tractably be computed. Instead, we use the following factorized variational approximation:

$$Q(\mathcal{R}, \mathcal{S}, \mathcal{Y}) = \Big(\prod_{c,d,k} g_{dk}^{r_{dk}^c} \Big) \Big(\prod_{c,n,k,j} m_{nkj}^{c\, s_{kj}^c y_n^c} \Big) \Big(\prod_{c,n} z_n^{c\, y_n^c} \Big).$$

Note that, as in standard MCVQ, we restrict the posterior selections of VQ made for each training case to be identical, that is, $\{g_{dk}\}$ is independent of c. Using the variational posterior and the likelihood, Equation (6), we obtain the following lower bound on the log-likelihood:

$$\begin{aligned}
\mathcal{F} &= E_Q \left[\log \mathrm{P}(\mathcal{X}, \mathcal{R}, \mathcal{S}, \mathcal{Y} | \theta) - \log Q(\mathcal{R}, \mathcal{S}, \mathcal{Y} | \mathbf{X}, \theta) \right], \\
&= E_Q \left[\sum_c \log \mathrm{P}(\mathbf{x}^c | \mathbf{R}^c, \mathbf{S}^c, \mathbf{y}^c, \theta) + \sum_c \log \mathrm{P}(\mathbf{R}^c, \mathbf{S}^c, \mathbf{y}^c) - \sum_c \log Q(\mathbf{R}^c, \mathbf{S}^c, \mathbf{y}^c) \right], \\
&= - \sum_{c,d,k} g_{dk} \log \frac{g_{dk}}{a_{dk}} - \sum_{c,n,k,j} m_{nkj}^c z_n^c \log \frac{m_{nkj}^c}{b_{nkj}} - \sum_{c,n} z_n^c \log \frac{z_n^c}{\beta_n} \\
&\quad - \sum_{c,d,k,j} g_{dk} \Big(\sum_n m_{nkj}^c z_n^c \Big) \varepsilon_{dkj}^c - \frac{CD}{2} \log(2\pi).
\end{aligned}$$

By differentiating $\mathcal{F}$ with respect to each of the parameters and latent variables, and solving for their respective maxima, we obtain the EM updates used for learning the model. The E-step updates for m_{nkj}^c and z_n^c are

$$m_{nkj}^c \propto b_{nkj} \exp \Big(- \sum_d g_{dk} \varepsilon_{dkj}^c \Big),$$

$$z_n^c \propto \beta_n \exp\left(\sum_{kj} m_{nkj}^c \log \frac{b_{nkj}}{m_{nkj}^c} - \sum_{dkj} g_{dk} m_{nkj}^c \varepsilon_{dkj}^c\right).$$

The M-step updates for $a_{dk}, g_{dk}, \mu_{dkj}$, and σ_{dkj} are unchanged from standard MCVQ, except for the substitution of $\sum_n (m_{nkj}^c z_n^c)$ in place of m_{kj}^c, wherever it appears. The updates for β_n and b_{nkj} are

$$\beta_n = \frac{1}{C}\sum_c z_n^c, \qquad\qquad b_{nkj} = \frac{\sum_c m_{nkj}^c z_n^c}{\sum_c z_n^c}.$$

A useful interpretation of the latent variable $\mathbf{y}$ is that, for a given data vector, it indicates the assignment of that datum to one of N clusters. Specifically, z_n^c can be thought of as the posterior probability that example c belongs to cluster n.

4.3 Supervised Case

In the supervised case, we are given a set of labeled training data $(\mathbf{x}^c, \mathbf{y}^c)$ for $c = 1 \ldots C$. Note that this case is essentially the same as the unsupervised case—we can obtain the supervised updates by constraining $\mathbf{z}^c = \mathbf{y}^c, \forall c$. Since the class is known, we may now drop the subscript n from m_{nkj}^c.

As stated earlier, when the $\mathbf{y}$'s are given, we learn a different prior over state selections for each class n:

$$b_{nkj} = \frac{1}{C\beta_n}\sum_c m_{kj}^c y_n^c,$$

where the prior probability of observing class n, β_n, can be calculated from the training labels:

$$\beta_n = \frac{1}{C}\sum_c y_n^c.$$

The posterior probabilities of state selection for each example c, m_{kj}^c, depend only on the prior corresponding to c's class:

$$m_{kj}^c \propto b_{y^c kj} \exp\left(-\sum_d g_{dk} \varepsilon_{dkj}^c\right).$$

4.4 Experiments

In this section we present experimental results obtained by training the above models on images taken from the AR Face Database (Martinez and Benavente, 1998). This data consists of images of frontal faces of 126 subjects under a number of different conditions. The five conditions used for these experiments were: 1) anger, 2) neutral, 3) scream, 4) smile, and 5) sunglasses.

The data set in its raw form contains faces which, although roughly centered, appear at different locations, angles, and scales. Since the learning algorithms do not attempt to compensate for these differences, we manually aligned each face such that the eyes always appeared in the same location. Next we cropped the images tightly around the face, and subsampled to reduce the size to 29×22 pixels. Finally, we converted the image data to grayscale, with pixel values ranging from -1 to 1. Examples of the preprocessed images can be seen in Figure 10.

The above preprocessing steps are standard when applying unsupervised learning methods, such as NMF, PCA, ICA, etc., to image data.

Figure 10: Examples of preprocessed images from the AR Face Database. These images, from left to right, correspond to the conditions: anger, neutral, scream, smile, and sunglasses.

4.4.1 SUPERVISED CASE: LEARNING CLASS-CONDITIONAL PRIORS

We first trained a supervised model on the entire data set, with a goal of learning a different prior over state selections for each of the five classes of image (anger, neutral, scream, smile, and sunglasses).

The model consisted of 5 VQs, 10 states each. The image regions that each VQ learned to explain are shown in Figure 11. For each VQ k we have plotted, as a grayscale image, the prior probability of each pixel selecting k (i.e., a_{dk}, d = pixel index). Two regions which we expected to be class discriminative, the mouth and the eyes, were captured by VQs 3 and 4 respectively.

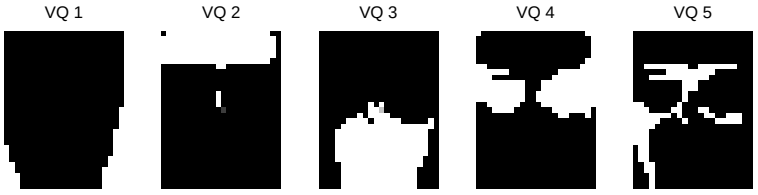


Figure 11: Image regions explained by each VQ. The prior probability of a VQ being selected for each pixel is plotted as a gray value between 0=black and 1=white.

The priors over state selection for these VQs varied widely depending on the class of the image being considered. As an example, in Figure 12 we have plotted the prior probability, given the class, of selecting each of the states from VQ 3. In the figure we can see that the prior probabilities closely matched our intuition as to which mouth shapes corresponded to which facial expressions. For example the first state (top left) appears to depict a smiling mouth. Accordingly, the *smile* class assigned it the highest prior probability, 0.34, while the other classes each assigned it 0.05 or less. Also, given the *scream* class, we see that the highest prior probabilities were assigned to the third and fourth states—both widely screaming mouths. States 6 through 9 (second row) range from depicting a neutral to an angry mouth.

Note that for *sunglasses*, which does not presuppose a mouth shape, the prior showed a preference for the more *neutral* mouths. The explanation for this is simply that most subjects in the data adopted a neutral expression when wearing sunglasses.

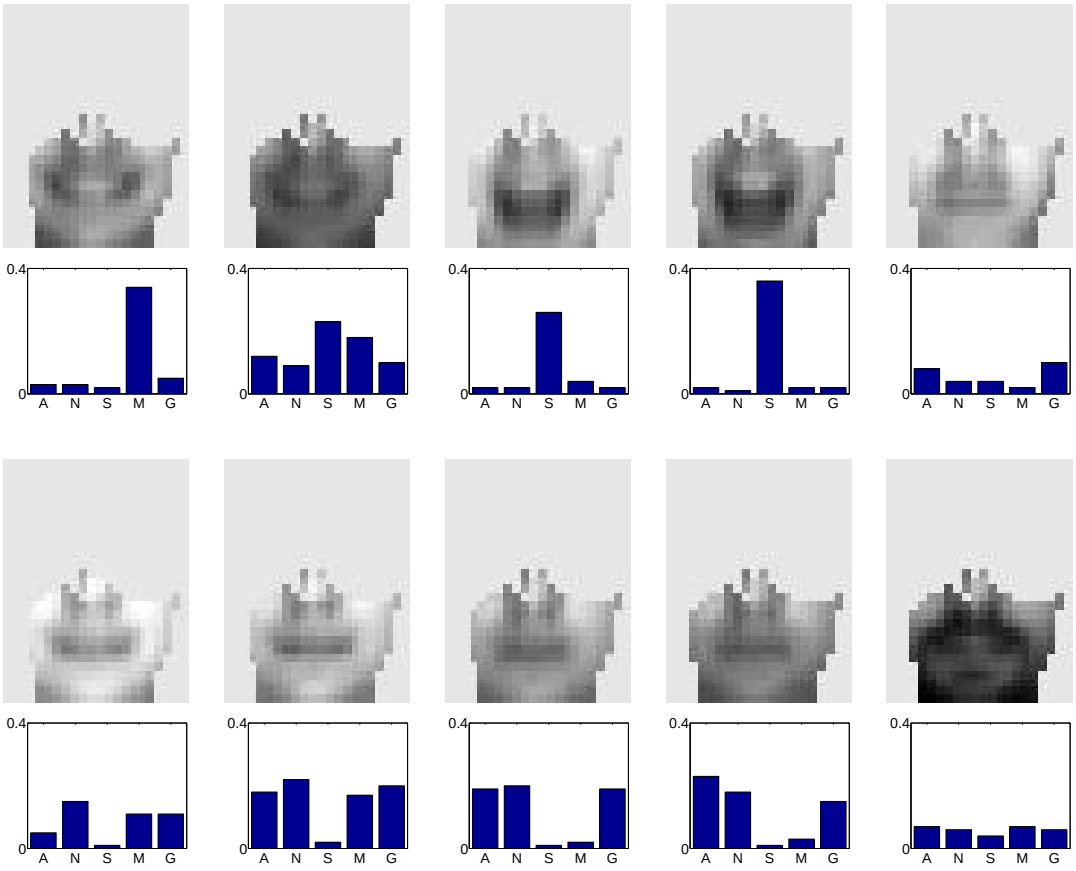


Figure 12: Class-conditional priors over mouth selections. Each image displayed is the mean of a state learned for VQ 3, masked (multiplied) by the prior probability of VQ 3 being selected for that pixel. The bar charts indicate, for each state, the prior probability of it being selected given each of the five class labels. From left to right these are anger (A), neutral (N), scream (S), smile (M), and sunglasses (G). On each chart, the y-axis extends up to a probability of 0.4.

One method for qualitatively evaluating the suitability of a class-conditional prior is to use it to generate novel images from the model, and see how well they match the class. Samples drawn this way using each of the five priors can be seen in Figure 13.

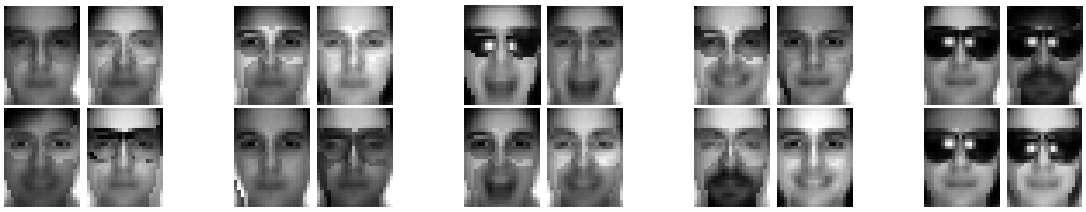


Figure 13: Sample images generated from the supervised hierarchical MCVQ model. Four images are depicted for each of the five class-conditional priors. Note the generated images can contain a novel combination of parts not seen in the training data, such as a person screaming *and* wearing glasses (Scream, upper-left). (The images do not include the Gaussian noise described in the generative process.)

4.4.2 UNSUPERVISED CASE

To evaluate the performance of the unsupervised model, we trained a model on a subset of the five classes, with the hope that it would learn clusters corresponding to the original classes.

The data set was restricted to contain only three classes—*neutral*, *scream*, and *sunglasses*—totaling 765 images. To prevent the images from being clustered simply by their overall brightness levels, for this experiment we performed equalization of the intensity histogram for each of the training images. The model we trained consisted of 15 VQs, 10 states each, and the latent class variable had 3 settings (i.e., 3 clusters).

In Table 3 we see the relationship between learned clusters and classes. The first cluster corresponded to the *sunglasses* class, containing all but two of the *sunglasses* images. The second and third clusters contained approximately equal numbers of *neutral* and *scream* images. Closer examination revealed that, of those not wearing sunglasses, 88% of the males had been placed in cluster 2, and 80% of the females in cluster 3. While we predicted that the model would learn classes corresponding to *neutral* and *scream*, it appears instead to have learned classes corresponding to gender. Examples of training images assigned to each of the clusters are shown in Figure 14.

Figure 14: Two training examples randomly selected from each of the three learned clusters.

cluster	1	2	3
neutral	0	154	101
scream	0	139	116
sunglasses	253	0	2
totals	253	293	219

cluster	1	2	3
female	112	46	184
male	141	247	35
totals	253	293	219

Table 3: Number of images from each class assigned to each cluster. We consider an image to belong to the cluster with the highest posterior probability (z_n^c).

4.5 Discussion

In this section we have presented an extension to MCVQ that allows higher-level causes or relationships to be learned from the data. Specifically, assuming the data comes from a pre-specified number of classes, this extension models the relationships between data vectors, based on the state selections each class favours in an MCVQ model.

Given a set of labeled data, such as facial images classified by the expression of the subject, this approach learns a single vocabulary of parts, and the likelihood of each part appearing in images of a given class. These probabilities are of interest since, by applying Bayes' rule, we can discover how the possible states for each feature affect what class a data vector will belong to.

Finally, when the data are not labeled, the proposed method can learn a clustering of the data into classes while simultaneously learning the relationships described above.

5. Related Models

MCVQ and MCFA fall into the expanding class of unsupervised algorithms known as *factorial methods*, in which the aim of the learning algorithm is to discover multiple independent causes, or factors, that can well characterize the observed data. Their direct ancestor is Cooperative Vector Quantization (Zemel, 1993; Hinton and Zemel, 1994; Ghahramani, 1995), which has a very similar generative model to MCVQ, but lacks the stochastic selection of one VQ per data dimension. Instead, a data vector is generated cooperatively: each VQ selects one vector, and these vectors are summed to produce the data (again using a Gaussian noise model). The contrast between these approaches mirrors the development of the competitive mixture-of-experts algorithm (Jacobs et al., 1991) which grew out of the inability of a cooperative, linear combination of experts to decompose inputs into separable experts.

Unfortunately Cooperative Vector Quantization can learn unintuitive global features which include both additive and subtractive effects. The aforementioned non-negative matrix factorization (NMF) (Lee and Seung, 1999, 2001; Mel, 1999) overcomes this problem by proposing that each data vector is generated by taking a non-negative linear combination of non-negative basis vectors. Since each basis vector contains only non-negative values, it is unable to 'subtract away' the effects of other basis vectors it is combined with. This property encourages learning a basis of sparse vectors, each capturing a single instantiation of one of the independent latent factors, for example a local feature of an image. Like NMF, given non-negative data MCVQ will learn a non-negative basis, taken only in non-negative combinations. Unlike MCVQ and MCFA, NMF provides no mechanism for learning compositional structure - how basis images or parts may be combined

to form a valid whole. Rather, it considers any non-negative linear combination of basis vectors to be equally suitable, and hence NMF and MCVQ models differ in the range of novel examples they can generate. Interestingly, the conditions under which non-negative matrix factorization will learn a correct parts-based decomposition, as shown by Donoho and Stodden (2004), closely resemble the generative model proposed by MCVQ. However one of these conditions—that the data set contain a complete factorial sampling of all J^K possible part configurations—seems difficult to achieve in practice. Other work such as Li et al. (2001) suggests that, when using realistic data sets, non-negativity alone may not be sufficient to ensure the learned basis corresponds to localized parts.

MCVQ also resembles a wide range of generative models developed to address image segmentation (Williams and Adams, 1999; Hinton et al., 2000; Jojic and Frey, 2001). These are generally complex, hierarchical models designed to focus on a different aspect of this problem than that of MCVQ: to dynamically decide which pixels belong to which objects. The chief obstacle faced by these models is the unknown pose (primarily limited to position) of an object in an image, and they employ learned object models to find the single object that best explains each pixel. MCVQ adopts a more constrained solution with respect to part locations, assuming that these are consistent across images, and instead focuses on the assembling of input dimensions into parts, and the variety of instantiations of each part. The constraints built into MCVQ limit its generality, but also lead to rapid learning and inference, and enable it to scale up to high-dimensional data.

A recent generative model closely related to MCVQ is the Probabilistic Index Map (PIM) (Jojic and Caspi, 2004; Winn and Jojic, 2005). PIMs propose that an image is generated by selecting, for each pixel, one colour from a palette of K colours (in the simplest case—the palette could also contain texture, filter coefficients, etc.). Pixels are grouped together if they select the same index into the palette. Across a collection of images, segmentation of the pixels into consistent parts is accomplished via a shared prior distribution over the palette index, and variation between images is accounted for by learning a different palette for each image. When learning parts, PIMs group together pixels which are self-similar (e.g., a similar colour), while MCVQ groups pixels which are *highly correlated*, regardless of their relative intensities.

Connections can also be made between MCVQ and algorithms for *biclustering*, which aim to produce a simultaneous clustering of both the rows and the columns of the data matrix (Mirkin, 1996). Biclustering has recently become popular in bioinformatics as a tool for analyzing DNA microarray data, which presents the expression levels for different genes under multiple experimental conditions as a matrix (Cheng and Church, 2000). Assuming column-vector data, the selection of a VQ for each data dimension in MCVQ produces a clustering of the rows. MCVQ differs from other biclustering methods in that it produces not one but K clusterings of the columns, one for each of the K VQs. In Section 4 we presented an hierarchical extension that combines the clusterings, allowing MCVQ to produce a single biclustering of the data.

Finally, MCVQ also closely relates to sparse matrix decomposition techniques, such as the *aspect model* (Hofmann, 1999), a latent variable model which associates an unobserved class variable, the aspect z, with each observation. Observations consist of co-occurrence statistics, such as counts of how often a specific word occurs in a document. The latent Dirichlet allocation model (LDA) (Blei et al., 2002) can be seen as a proper generative version of the aspect model: each document/input vector is not represented as a set of labels for a particular vector in the training set, and there is a natural way to examine the probability of some unseen vector. MCVQ shares the ability of these models to associate multiple aspects with a given document, yet it achieves this in a slightly different manner, since the two approaches present different ways of generating documents. The

aspect and LDA models propose that each document—a list of exchangeable words—is generated by sampling an aspect, then sampling a word from the aspect, for each word in the document. Thus each occurrence of a word is associated with a single aspect, but different aspects can generate the same word. On the other hand MCVQ models the aggregate word counts of a document. That is, each data vector has a number of components D equal to the size of the vocabulary, and x_d^c indicates the number of times word d appears in document c. For each word in the vocabulary, its entire document frequency is generated according to the dictates of a stochastically-selected aspect (VQ). The stochastic selection leads to a posterior probability stipulating a soft mixture over aspects for each word.

Recently LDA and the aspect model have also been applied to images, by first representing each image as an exchangeable set of 'visual words'—interest points or image patches extracted from the image (Fei-Fei and Perona, 2005; Sivic et al., 2005; Fergus et al., 2005; Sudderth et al., 2005). By selecting as 'words' (or parts) image features that can be recognized regardless of the position or scale at which they appear, these models can be made invariant to the position of the target object in the training images. Since these models are designed for object detection and image categorization, they learn very different object representations than MCVQ/MCFA. First, parts are represented using invariant descriptors (typically SIFT descriptors, Lowe, 2004), which are useful for recognizing a part but provide little information about its actual appearance. Second, since interest points generally do not appear on all regions of the object, the learned parts are not sufficient for describing all aspects of its appearance. Thus, unlike MCVQ/MCFA, one cannot use these models to synthesize or repair a realistic image of the object.

The generative model of MCFA, and the EM algorithm for learning it, are related to the mixture of factor analyzers model (MFA) (Ghahramani, 1995). The distinction between the two is that in MFA a data point is generated entirely by one selected factor analyzer, while in MCFA data dimensions can be generated by different FAs. Thus MFA does not attempt to learn parts, rather it fits the data with a mixture of linear manifolds. MCFA also resembles a recently proposed method employing factor analysis to model the appearance and occlusion masks of moving 'sprites' in video (Frey et al., 2003).

6. Conclusion

In this paper we have proposed an approach to learning parts-based models of data, and provided details for a discrete appearance model (MCVQ) and a continuous appearance model (MCFA) for the latent factors.

This approach can be used to learn informative and intuitively appealing models of various kinds of vector-valued data, such face images and movie ratings. The parts-based models can be interpreted as a set of sparse basis vectors, with constraints on how they can be combined to generate a valid data vector. The sparsity is not assumed a priori, or forcibly encoded in the model. Rather it results naturally from our assumption that different parts choose their states independently.

When the independence assumption does not hold, we have shown that the model can be extended hierarchically, learning the dependencies between latent states. This permits modeling of data containing multiple categories with a single vocabulary of parts, in addition to clustering data based on the states used for each part in the generative process.

Considering the four potential advantages of parts-based models for objects in images, as outlined in the introduction, our approach has managed to realize two of these. Firstly, addressing

advantage 2., our approach naturally allows the generation of new images that are novel combinations of familiar parts (see Figure 13). Secondly, addressing advantage 4., we have shown that MCVQ is robust to partial occlusions in the training images (see Section 3.1.1, including Figure 7). Unfortunately, however, the approach we present for learning parts is not well-suited to handling pose variation and articulation of the target object in images. This is because in our models a part is essentially a group of like-minded pixels, thus is tied to specific dimensions of the input vector. As such it cannot handle variation in the spatial location of parts. One way to address this would be to incorporate the transformation invariances developed by Frey and Jojic (2003).

An important direction for future research is the problem of automatically discovering the number of parts present in the data. Possible methods for this include employing ideas from variational Bayesian modeling (Beal and Ghahramani, 2003), or infinite mixture models and Dirichlet process (Teh et al., 2005).

Acknowledgments

We would like to thank Sam Roweis and Brendan Frey for helpful discussions, as well as the many reviewers for their thoughtful suggestions. This research was supported by Communications and Information Technology Ontario (CITO), the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Ontario Graduate Scholarship Program (OGS).

Appendix A.

In this appendix we motivate, in more detail, our choice of approximate posterior for variational EM learning of MCVQ (and consequently for MCFA as well). This choice is somewhat unorthodox in that the generative model proposes a selection variable r_{dk}^c for each dimension of each training vector c, yet the approximate posterior (3) includes only one parameter g_{dk} for all the training vectors. There are two alternatives which might seem more natural: the number of variational parameters could be increased to match the generative model, or the generative model could be modified so that the selection of factors is made only once for all data vectors.

The simplest and most conventional variation approximation, the *fully-factorized mean-field method* (Ghahramani, 1995), proposes using a set of variational parameters g_{dk}^c for each training case c. This causes a number of changes to the EM updates. First of all the g_{dk}^c's, now updated in the E-step, depend only on a single training case c, while in the M-step their prior is computed by averaging:

$$g_{dk}^c \propto a_{dk} \exp\left(-\sum_j m_{kj}^c \varepsilon_{dkj}^c\right), \qquad a_{dk} = \frac{1}{C}\sum_c g_{dk}^c.$$

The remaining changes consist of replacing g_{dk} with g_{dk}^c in the update of m_{kj}^c, and replacing m_{kj}^c with $g_{dk}^c m_{kj}^c$ in the updates of μ_{dkj} and σ_{dkj}^2. Experimentally this results in a weak (high-entropy) prior, unable to discover any parts in the data. Adding a low-entropy hyper-prior to the a_{dk}'s, as proposed by Brand (1999), did not improve learning.

Another closely related model proposes only a single set of factor selections, r_{dk}, which are used to generate all of the data vectors. For this model, which we will call the *r-tied* model, equation (3) is the natural fully-factorized mean-field approximation. Working through the EM updates, the only

change from Section 2.1.1 is in the update for g_{dk}, which becomes

$$g_{dk} \propto a_{dk} \exp\left(-\sum_{cj} m_{kj}^c \varepsilon_{dkj}^c\right).$$

This update differs from (4) only in the omission of a factor of $\frac{1}{C}$ inside the exponential. Thus the new g_{dk}'s can be obtained from the old ones through exponentiating by the power C and renormalizing. This has the effect of pushing the low probability factor selections closer to zero and high probability selections closer to 1, resulting in new g_{dk}'s with lower entropy.

As a quantitative comparison, we trained MCVQ models using each of the three algorithms on the face images described in Section 3.1. For each trained model we computed the mean and standard deviation of the entropy of the a_{dk} parameters, as well as the sum-of-squares error reconstructing 100 held-out face images. A low mean entropy indicates that a near-binary association between data dimensions and factors was been learned, while a low reconstruction error shows that a good model of faces was obtained. The results appear in Table 4.

Learning Algorithm	Standard	Fully-Factorized	r-Tied
Average Entropy in a_{dk}	0.6751	1.9662	0.4365
Squared Reconstruction Error	1.5163e+04	1.9699e+04	1.5362e+04

Table 4: A quantitative comparison of alternative learning algorithms for MCVQ.

As expected, the entropy in the prior distribution over factor selection is lowest in the r-tied model, and highest in the fully-factorized model. The fully-factorized model also has the highest reconstruction error, while the standard model shows a slight advantage over the r-tied model. In practice, the fully-factorized model performs poorly, and is unable to discover any parts in the data. The standard and r-tied models often show similar performance, but the standard algorithm is usually qualitatively better at discovering parts. A possible explanation could be that the r-tied updates push the g_{dk} parameters too quickly to a low-entropy configuration during the early stages of learning.

In the future we plan to explore the possibility of combining these alternatives, in hope of being able to realize benefits provided by each. For instance, the parameters learned by a standard or r-tied MCVQ model could be used as an initialization for learning with the fully-factorized variational approximation. This approach has the potential to be able to discover parts, while still allowing some variation in parts between data (e.g., borderline pixels could be assigned to the nose in some face images, and to the upper-lip in others).

Code for MCVQ, which also implements all the alternatives described here, can be obtained at http://www.cs.toronto.edu/~dross/mcvq/.

References

C. Andrieu, N. de Freitas, A. Doucet, , and M.I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

M.J. Beal and Z. Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. In *Bayesian Statistics 7*, 2003.

I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet Allocation. In S. Becker T. Dietterich and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.

C. Boutilier, R.S. Zemel, and B. Marlin. Active collaborative filtering. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 98–106. Morgan Kaufmann Publishers, 2003.

M. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182, 1999.

Y. Cheng and G.M Church. Biclustering of Expression Data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 2000.

D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *Proc. CVPR*, 2005.

R. Fergus, L. Fei-Fei, P. Perona A., and Zisserman. Learning object categories from google's image search. In *Proceedings of the 2005 IEEE International Conference on Computer Vision*, 2005.

B. J. Frey, N. Jojic, and A. Kannan. Learning appearance and transparency manifolds of occluded objects in layers. In *Proc. CVPR*, 2003.

B.J. Frey and N. Jojic. Transformation-invariant clustering using the EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):1–17, 2003.

Z. Ghahramani. Factorial learning and the EM algorithm. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge, MA, 1995.

Z. Ghahramani and G.E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, 1996.

B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images. A.I. Memo 1687, Massachusetts Institute of Technology, May 2000.

G. Hinton and R.S. Zemel. Autoencoders, minimum description length, and Helmholtz free energy. In G. Tesauro J. D. Cowan and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann Publishers, San Mateo, CA, 1994.

G.E. Hinton, Z. Ghahramani, and Y.W. Teh. Learning to parse images. In S.A. Solla, T.K. Leen, and K.R. Muller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, Cambridge, MA, 2000.

T. Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

N. Jojic and Y. Caspi. Capturing image structure with probabilistic index maps. In *Proc. CVPR*, 2004.

N. Jojic and B.J. Frey. Learning flexible sprites in video layers. In *Proc. CVPR*, 2001.

D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, October 1999.

D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In T.K. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge, MA, 2001.

S. Li, X. Hou, and H. Zhang. Learning spatially localized, parts-based representation. In *Proc. CVPR*, 2001.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision,*, 60(2):91–110, 2004.

B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.

A.M. Martinez and R. Benavente. The AR face database. Technical Report 24, CVC, 1998.

B.W. Mel. Think positive to find parts. *Nature*, 401:759–760, October 1999.

B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academic Publishers, 1996.

MIT-CBCL. CBCL face database #1. MIT Center For Biological and Computation Learning, 2000. http://cbcl.mit.edu/.

A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001.

R.M. Neal and G.E. Hinton. A new view of the EM algorithm that justifies incremental and other algorithms. In M.I. Jordan, editor, *Learning and Inference in Graphical Models*. Kluwer Academic Publishers, 1998.

D.A. Ross and R.S. Zemel. Multiple cause vector quantization. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, MA, 2003.

S. Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems 10*. MIT Press, Cambridge, MA, 1997.

J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering object categories in image collections. Technical Report A. I. Memo 2005-005, Massachusetts Institute of Technology, 2005.

E.B. Sudderth, A. Torralba, W.T. Freeman, and A.S. Wilsky. Learning hierarchical models of scenes, objects, and parts. In *Proceedings of the 2005 IEEE International Conference on Computer Vision*, 2005.

Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Sharing clusters among related groups: Hierarchical Dirichlet processes. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

M. Weber, W. Einhäuser, M. Welling, and P. Perona. Viewpoint-invariant learning and detection of human heads. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

C. Williams and N. Adams. DTs: Dynamic trees. In M.J. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, Cambridge, MA, 1999.

J. Winn and N. Jojic. LOCUS: Learning object classes with unsupervised segmentation. In *Proc. IEEE Intl. Conf. on Computer Vision (ICCV)*, 2005.

R.S. Zemel. *A Minimum Description Length Framework for Unsupervised Learning*. PhD thesis, Dept. of Computer Science, University of Toronto, Toronto, Canada, 1993.

Journal of Machine Learning Research 7 (2006) 2399-2434 Submitted 4/05; Revised 5/06; Published 11/06

Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples

Mikhail Belkin MBELKIN@CSE.OHIO-STATE.EDU
Department of Computer Science and Engineering
The Ohio State University
2015 Neil Avenue, Dreese Labs 597
Columbus, OH 43210, USA

Partha Niyogi NIYOGI@CS.UCHICAGO.EDU
Departments of Computer Science and Statistics
University of Chicago
1100 E. 58th Street
Chicago, IL 60637, USA

Vikas Sindhwani VIKASS@CS.UCHICAGO.EDU
Department of Computer Science
University of Chicago
1100 E. 58th Street
Chicago, IL 60637, USA

Editor: Peter Bartlett

Abstract

We propose a family of learning algorithms based on a new form of regularization that allows us to exploit the geometry of the marginal distribution. We focus on a semi-supervised framework that incorporates labeled and unlabeled data in a general-purpose learner. Some transductive graph learning algorithms and standard methods including support vector machines and regularized least squares can be obtained as special cases. We use properties of reproducing kernel Hilbert spaces to prove new Representer theorems that provide theoretical basis for the algorithms. As a result (in contrast to purely graph-based approaches) we obtain a natural out-of-sample extension to novel examples and so are able to handle both transductive and truly semi-supervised settings. We present experimental evidence suggesting that our semi-supervised algorithms are able to use unlabeled data effectively. Finally we have a brief discussion of unsupervised and fully supervised learning within our general framework.

Keywords: semi-supervised learning, graph transduction, regularization, kernel methods, manifold learning, spectral graph theory, unlabeled data, support vector machines

1. Introduction

In this paper, we introduce a framework for data-dependent regularization that exploits the geometry of the probability distribution. While this framework allows us to approach the full range of learning problems from unsupervised to supervised (discussed in Sections 6.1 and 6.2 respectively), we focus on the problem of semi-supervised learning.

The problem of learning from labeled and unlabeled data (*semi-supervised* and *transductive* learning) has attracted considerable attention in recent years. Some recently proposed methods

include transductive SVM (Vapnik, 1998; Joachims, 1999), cotraining (Blum and Mitchell, 1998), and a variety of graph-based methods (Blum and Chawla, 2001; Chapelle et al., 2003; Szummer and Jaakkola, 2002; Kondor and Lafferty, 2002; Smola and Kondor, 2003; Zhou et al., 2004; Zhu et al., 2003, 2005; Kemp et al., 2004; Joachims, 1999; Belkin and Niyogi, 2003b). We also note the regularization based techniques of Corduneanu and Jaakkola (2003) and Bousquet et al. (2004). The latter reference is closest in spirit to the intuitions of our paper. We postpone the discussion of related algorithms and various connections until Section 4.5.

The idea of regularization has a rich mathematical history going back to Tikhonov (1963), where it is used for solving ill-posed inverse problems. Regularization is a key idea in the theory of splines (e.g., Wahba, 1990) and is widely used in machine learning (e.g., Evgeniou et al., 2000). Many machine learning algorithms, including support vector machines, can be interpreted as instances of regularization.

Our framework exploits the geometry of the probability distribution that generates the data and incorporates it as an additional regularization term. Hence, there are two regularization terms—one controlling the complexity of the classifier in the *ambient space* and the other controlling the complexity as measured by the *geometry* of the distribution. We consider in some detail the special case where this probability distribution is supported on a submanifold of the ambient space.

The points below highlight several aspects of the current paper:

1. Our general framework brings together three distinct concepts that have received some independent recent attention in machine learning:
   i. The first of these is the technology of *spectral graph theory* (see, e.g., Chung, 1997) that has been applied to a wide range of clustering and classification tasks over the last two decades. Such methods typically reduce to certain eigenvalue problems.
   ii. The second is the geometric point of view embodied in a class of algorithms that can be termed as *manifold learning*.[1] These methods attempt to use the geometry of the probability distribution by assuming that its support has the geometric structure of a Riemannian manifold.
   iii. The third important conceptual framework is the set of ideas surrounding regularization in Reproducing Kernel Hilbert Spaces (RKHS). This leads to the class of *kernel based algorithms* for classification and regression (e.g., Scholkopf and Smola, 2002; Wahba, 1990; Evgeniou et al., 2000).

   We show how these ideas can be brought together in a coherent and natural way to incorporate geometric structure in a kernel based regularization framework. As far as we know, these ideas have not been unified in a similar fashion before.

2. This general framework allows us to develop algorithms spanning the range from unsupervised to fully supervised learning.

   In this paper we primarily focus on the semi-supervised setting and present two families of algorithms: the Laplacian Regularized Least Squares (hereafter, LapRLS) and the Laplacian Support Vector Machines (hereafter LapSVM). These are natural extensions of RLS and SVM respectively. In addition, several recently proposed transductive methods (e.g., Zhu et al., 2003; Belkin and Niyogi, 2003b) are also seen to be special cases of this general approach.

1. See http://www.cse.msu.edu/∼lawhiu/manifold/ for a long list of references.

In the absence of labeled examples our framework results in new algorithms for unsupervised learning, which can be used both for data representation and clustering. These algorithms are related to spectral clustering and Laplacian Eigenmaps (Belkin and Niyogi, 2003a).

3. We elaborate on the RKHS foundations of our algorithms and show how geometric knowledge of the probability distribution may be incorporated in such a setting through an additional regularization penalty. In particular, a new Representer theorem provides a functional form of the solution when the distribution is known; its empirical version involves an expansion over labeled and unlabeled points when the distribution is unknown. These Representer theorems provide the basis for our algorithms.

4. Our framework with an ambiently defined RKHS and the associated Representer theorems result in a natural out-of-sample extension from the data set (labeled and unlabeled) to novel examples. This is in contrast to the variety of purely graph-based approaches that have been considered in the last few years. Such graph-based approaches work in a transductive setting and do not naturally extend to the semi-supervised case where novel test examples need to be classified (predicted). Also see Bengio et al. (2004) and Brand (2003) for some recent related work on out-of-sample extensions. We also note that a method similar to our regularized spectral clustering algorithm has been independently proposed in the context of graph inference in Vert and Yamanishi (2005).

The work presented here is based on the University of Chicago Technical Report TR-2004-05, a short version in the Proceedings of AI and Statistics 2005, Belkin et al. (2005) and Sindhwani (2004).

1.1 The Significance of Semi-Supervised Learning

From an engineering standpoint, it is clear that collecting labeled data is generally more involved than collecting unlabeled data. As a result, an approach to pattern recognition that is able to make better use of unlabeled data to improve recognition performance is of potentially great practical significance.

However, the significance of semi-supervised learning extends beyond purely utilitarian considerations. Arguably, most natural (human or animal) learning occurs in the semi-supervised regime. We live in a world where we are constantly exposed to a stream of natural stimuli. These stimuli comprise the unlabeled data that we have easy access to. For example, in phonological acquisition contexts, a child is exposed to many acoustic utterances. These utterances do not come with identifiable phonological markers. Corrective feedback is the main source of directly labeled examples. In many cases, a small amount of feedback is sufficient to allow the child to master the acoustic-to-phonetic mapping of any language.

The ability of humans to learn unsupervised concepts (e.g., learning clusters and categories of objects) suggests that unlabeled data can be usefully processed to learn natural invariances, to form categories, and to develop classifiers. In most pattern recognition tasks, humans have access only to a small number of labeled examples. Therefore the success of human learning in this "small sample" regime is plausibly due to effective utilization of the large amounts of unlabeled data to extract information that is useful for generalization.

Consequently, if we are to make progress in understanding how natural learning comes about, we need to think about the basis of semi-supervised learning. Figure 1 illustrates how unlabeled

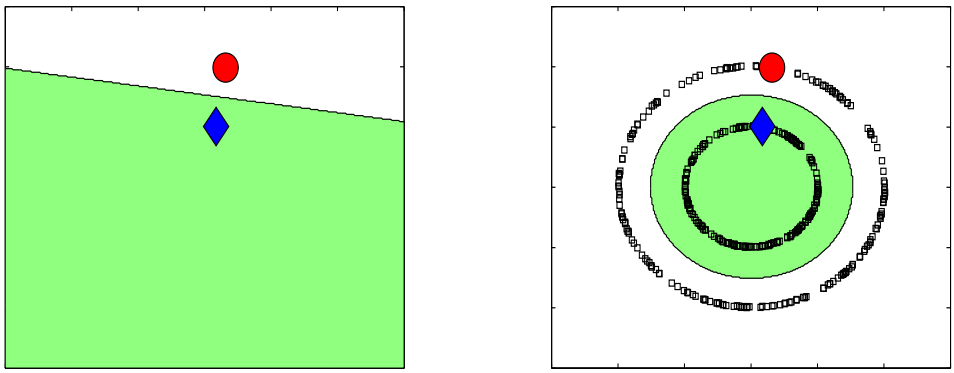


Figure 1: Unlabeled data and prior beliefs

examples may force us to restructure our hypotheses during learning. Imagine a situation where one is given two labeled examples—one positive and one negative—as shown in the left panel. If one is to induce a classifier on the basis of this, a natural choice would seem to be the linear separator as shown. Indeed, a variety of theoretical formalisms (Bayesian paradigms, regularization, minimum description length or structural risk minimization principles, and the like) have been constructed to rationalize such a choice. In most of these formalisms, one structures the set of one's hypothesis functions by a prior notion of simplicity and one may then justify why the linear separator is the simplest structure consistent with the data.

Now consider the situation where one is given additional unlabeled examples as shown in the right panel. We argue that it is self-evident that in the light of this new unlabeled set, one must re-evaluate one's prior notion of simplicity. The particular geometric structure of the marginal distribution suggests that the most natural classifier is now the circular one indicated in the right panel. Thus the geometry of the marginal distribution must be incorporated in our regularization principle to impose structure on the space of functions in nonparametric classification or regression. This is the intuition we formalize in the rest of the paper. The success of our approach depends on whether we can extract structure from the marginal distribution, and on the extent to which such structure may reveal the underlying truth.

1.2 Outline of the Paper

The paper is organized as follows: in Section 2, we develop the basic framework for semi-supervised learning where we ultimately formulate an objective function that can use both labeled and unlabeled data. The framework is developed in an RKHS setting and we state two kinds of Representer theorems describing the functional form of the solutions. In Section 3, we elaborate on the theoretical underpinnings of this framework and prove the Representer theorems of Section 2. While the Representer theorem for the finite sample case can be proved using standard orthogonality arguments, the Representer theorem for the known marginal distribution requires more subtle considerations. In Section 4, we derive the different algorithms for semi-supervised learning that arise out of our framework. Connections to related algorithms are stated. In Section 5, we describe experiments that evaluate the algorithms and demonstrate the usefulness of unlabeled data. In Section 6,

we consider the cases of fully supervised and unsupervised learning. In Section 7 we conclude this paper.

2. The Semi-Supervised Learning Framework

Recall the standard framework of learning from examples. There is a probability distribution P on $X \times \mathbb{R}$ according to which examples are generated for function learning. Labeled examples are (x,y) pairs generated according to P. Unlabeled examples are simply $x \in X$ drawn according to the marginal distribution $\mathcal{P}_X$ of P.

One might hope that knowledge of the marginal $\mathcal{P}_X$ can be exploited for better function learning (e.g., in classification or regression tasks). Of course, if there is no identifiable relation between $\mathcal{P}_X$ and the conditional $\mathcal{P}(y|x)$, the knowledge of $\mathcal{P}_X$ is unlikely to be of much use.

Therefore, we will make a specific assumption about the connection between the marginal and the conditional distributions. We will assume that if two points $x_1, x_2 \in X$ are *close* in the *intrinsic* geometry of $\mathcal{P}_X$, then the conditional distributions $\mathcal{P}(y|x_1)$ and $\mathcal{P}(y|x_2)$ are similar. In other words, the conditional probability distribution $\mathcal{P}(y|x)$ varies smoothly along the geodesics in the intrinsic geometry of $\mathcal{P}_X$.

We use these geometric intuitions to extend an established framework for function learning. A number of popular algorithms such as SVM, Ridge regression, splines, Radial Basis Functions may be broadly interpreted as regularization algorithms with different empirical cost functions and complexity measures in an appropriately chosen Reproducing Kernel Hilbert Space (RKHS).

For a Mercer kernel $K : X \times X \to \mathbb{R}$, there is an associated RKHS $\mathcal{H}_K$ of functions $X \to \mathbb{R}$ with the corresponding norm $\| \ \|_K$. Given a set of labeled examples (x_i, y_i), $i = 1, \ldots, l$ the standard framework estimates an unknown function by minimizing

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma \| f \|_K^2, \tag{1}$$

where V is some loss function, such as squared loss $(y_i - f(x_i))^2$ for RLS or the hinge loss function $\max[0, 1 - y_i f(x_i)]$ for SVM. Penalizing the RKHS norm imposes smoothness conditions on possible solutions. The classical Representer Theorem states that the solution to this minimization problem exists in $\mathcal{H}_K$ and can be written as

$$f^*(x) = \sum_{i=1}^{l} \alpha_i K(x_i, x).$$

Therefore, the problem is reduced to optimizing over the finite dimensional space of coefficients α_i, which is the algorithmic basis for SVM, regularized least squares and other regression and classification schemes.

We first consider the case when the marginal distribution is already known.

2.1 Marginal $\mathcal{P}_X$ is Known

Our goal is to extend this framework by incorporating additional information about the geometric structure of the marginal $\mathcal{P}_X$. We would like to ensure that the solution is smooth with respect to both the ambient space and the marginal distribution $\mathcal{P}_X$. To achieve that, we introduce an additional

regularizer:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \|f\|_I^2, \tag{2}$$

where $\|f\|_I^2$ is an appropriate penalty term that should reflect the intrinsic structure of $\mathcal{P}_X$. Intuitively, $\|f\|_I^2$ is a smoothness penalty corresponding to the probability distribution. For example, if the probability distribution is supported on a low-dimensional manifold, $\|f\|_I^2$ may penalize f along that manifold. γ_A controls the complexity of the function in the *ambient* space while γ_I controls the complexity of the function in the *intrinsic* geometry of $\mathcal{P}_X$. It turns out that one can derive an explicit functional form for the solution f^* as shown in the following theorem.

Theorem 1 *Assume that the penalty term $\|f\|_I$ is sufficiently smooth with respect to the RKHS norm $\|f\|_K$ (see Section 3.2 for the exact statement). Then the solution f^* to the optimization problem in Equation 2 above exists and admits the following representation*

$$f^*(x) = \sum_{i=1}^{l} \alpha_i K(x_i, x) + \int_{\mathcal{M}} \alpha(z) K(x, z)\, d\mathcal{P}_X(z) \tag{3}$$

where $\mathcal{M} = \text{supp}\{\mathcal{P}_X\}$ is the support of the marginal $\mathcal{P}_X$.

We postpone the proof and the formulation of smoothness conditions on the norm $\|\ \|_I$ until the next section.

The Representer Theorem above allows us to express the solution f^* directly in terms of the labeled data, the (ambient) kernel K, and the marginal $\mathcal{P}_X$. If $\mathcal{P}_X$ is unknown, we see that the solution may be expressed in terms of an empirical estimate of $\mathcal{P}_X$. Depending on the nature of this estimate, different approximations to the solution may be developed. In the next section, we consider a particular approximation scheme that leads to a simple algorithmic framework for learning from labeled and unlabeled data.

2.2 Marginal $\mathcal{P}_X$ Unknown

In most applications the marginal $\mathcal{P}_X$ is not known. Therefore we must attempt to get empirical estimates of $\mathcal{P}_X$ and $\|\ \|_I$. Note that in order to get such empirical estimates it is sufficient to have *unlabeled* examples.

A case of particular recent interest (for example, see Roweis and Saul, 2000; Tenenbaum et al., 2000; Belkin and Niyogi, 2003a; Donoho and Grimes, 2003; Coifman et al., 2005, for a discussion on dimensionality reduction) is when the support of $\mathcal{P}_X$ is a compact submanifold $\mathcal{M} \subset \mathbb{R}^n$. In that case, one natural choice for $\|f\|_I$ is $\int_{x \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 d\mathcal{P}_X(x)$, where $\nabla_{\mathcal{M}}$ is the *gradient* (see, for example Do Carmo, 1992, for an introduction to differential geometry) of f along the manifold $\mathcal{M}$ and the integral is taken over the marginal distribution.

The optimization problem becomes

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \int_{x \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 d\mathcal{P}_X(x).$$

The term $\int_{x \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 d\mathcal{P}_X(x)$ may be approximated on the basis of labeled and unlabeled data using the graph Laplacian associated to the data. While an extended discussion of these issues goes

beyond the scope of this paper, it can be shown that under certain conditions choosing exponential weights for the adjacency graph leads to convergence of the graph Laplacian to the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ (or its weighted version) on the manifold. See the Remarks below and Belkin (2003); Lafon (2004); Belkin and Niyogi (2005); Coifman et al. (2005); Hein et al. (2005) for details.

Thus, given a set of l labeled examples $\{(x_i, y_i)\}_{i=1}^{l}$ and a set of u unlabeled examples $\{x_j\}_{j=l+1}^{j=l+u}$, we consider the following optimization problem:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2} \sum_{i,j=1}^{l+u} (f(x_i) - f(x_j))^2 W_{ij},$$

$$= \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2} \mathbf{f}^T L \mathbf{f}. \tag{4}$$

where W_{ij} are edge weights in the data adjacency graph, $\mathbf{f} = [f(x_1), \ldots, f(x_{l+u})]^T$, and L is the graph Laplacian given by $L = D - W$. Here, the diagonal matrix D is given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$. The normalizing coefficient $\frac{1}{(u+l)^2}$ is the natural scale factor for the empirical estimate of the Laplace operator. We note than on a sparse adjacency graph it may be replaced by $\sum_{i,j=1}^{l+u} W_{ij}$.

The following version of the Representer Theorem shows that the minimizer has an expansion in terms of both labeled and unlabeled examples and is a key to our algorithms.

Theorem 2 *The minimizer of optimization problem 4 admits an expansion*

$$f^*(x) = \sum_{i=1}^{l+u} \alpha_i K(x_i, x) \tag{5}$$

in terms of the labeled and unlabeled examples.

The proof is a variation of the standard orthogonality argument and is presented in Section 3.4.

Remark 1: Several natural choices of $\| \|_I$ exist. Some examples are:

1. Iterated Laplacians $(\Delta_{\mathcal{M}})^k$. Differential operators $(\Delta_{\mathcal{M}})^k$ and their linear combinations provide a natural family of smoothness penalties.

   Recall that the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ can be defined as the divergence of the gradient vector field $\Delta_{\mathcal{M}} f = \operatorname{div}(\nabla_{\mathcal{M}} f)$ and is characterized by the equality

   $$\int_{x \in \mathcal{M}} f(x) \Delta_{\mathcal{M}} f(x) d\mu = \int_{x \in \mathcal{M}} \|\nabla_{\mathcal{M}} f(x)\|^2 d\mu.$$

   where μ is the standard measure (uniform distribution) on the Riemannian manifold. If μ is taken to be non-uniform, then the corresponding notion is the weighted Laplace-Beltrami operator (e.g., Grigor'yan, 2006).

2. Heat semigroup $e^{-t\Delta_{\mathcal{M}}}$ is a family of smoothing operators corresponding to the process of diffusion (Brownian motion) on the manifold. One can take $\|f\|_I^2 = \int_{\mathcal{M}} f e^{t\Delta_{\mathcal{M}}}(f) d\mathcal{P}_X$. We note that for small values of t the corresponding Green's function (the heat kernel of $\mathcal{M}$), which is close to a Gaussian in the geodesic coordinates, can also be approximated by a sharp Gaussian in the ambient space.

3. Squared norm of the Hessian (cf. Donoho and Grimes, 2003). While the Hessian $\mathbf{H}(f)$ (the matrix of second derivatives of f) generally depends on the coordinate system, it can be shown that the Frobenius norm (the sum of squared eigenvalues) of $\mathbf{H}$ is the same in any geodesic coordinate system and hence is invariantly defined for a Riemannian manifold $\mathcal{M}$. Using the Frobenius norm of $\mathbf{H}$ as a regularizer presents an intriguing generalization of thin-plate splines. We also note that $\Delta_{\mathcal{M}}(f) = \mathrm{tr}(\mathbf{H}(f))$.

Remark 2: Why not just use the intrinsic regularizer? Using ambient and intrinsic regularizers jointly is important for the following reasons:

1. We do not usually have access to $\mathcal{M}$ or the true underlying marginal distribution, just to data points sampled from it. Therefore regularization with respect only to the sampled manifold is ill-posed. By including an ambient term, the problem becomes well-posed.

2. There may be situations when regularization with respect to the ambient space yields a better solution, for example, when the manifold assumption does not hold (or holds to a lesser degree). Being able to trade off these two regularizers may be important in practice.

Remark 3: While we use the graph Laplacian for simplicity, the *normalized Laplacian*

$$\tilde{L} = D^{-1/2} L D^{-1/2}$$

can be used interchangeably in all our formulas. Using $\tilde{L}$ instead of L provides certain theoretical guarantees (see von Luxburg et al., 2004) and seems to perform as well or better in many practical tasks. In fact, we use $\tilde{L}$ in all our empirical studies in Section 5. The relation of $\tilde{L}$ to the weighted Laplace-Beltrami operator was discussed in Lafon (2004).

Remark 4: Note that a global kernel K restricted to $\mathcal{M}$ (denoted by $K_{\mathcal{M}}$) is also a kernel defined on $\mathcal{M}$ with an associated RKHS $\mathcal{H}_{\mathcal{M}}$ of functions $\mathcal{M} \to \mathbb{R}$. While this might suggest

$$\|f\|_I = \|f_{\mathcal{M}}\|_{K_{\mathcal{M}}}$$

($f_{\mathcal{M}}$ is f restricted to $\mathcal{M}$) as a reasonable choice for $\|f\|_I$, it turns out, that for the minimizer f^* of the corresponding optimization problem we get $\|f^*\|_I = \|f^*\|_K$, yielding the same solution as standard regularization, although with a different parameter γ. This observation follows from the restriction properties of RKHS discussed in the next section and is formally stated as Proposition 6. Therefore it is impossible to have an out-of-sample extension without two *different* measures of smoothness. On the other hand, a different ambient kernel restricted to $\mathcal{M}$ can potentially serve as the intrinsic regularization term. For example, a sharp Gaussian kernel can be used as an approximation to the heat kernel on $\mathcal{M}$. Thus one (sharper) kernel may be used in conjunction with unlabeled data to estimate the heat kernel on $\mathcal{M}$ and a wider kernel for inference.

3. Theoretical Underpinnings and Results

In this section we briefly review the theory of reproducing kernel Hilbert spaces and their connection to integral operators. We proceed to establish the Representer theorems from the previous section.

3.1 General Theory of RKHS

We start by recalling some basic properties of reproducing kernel Hilbert spaces (see the original work of Aronszajn, 1950; Cucker and Smale, 2002, for a nice discussion in the context of learning theory) and their connections to integral operators. We say that a Hilbert space $\mathcal{H}$ of functions $X \to \mathbb{R}$ has the *reproducing property*, if $\forall x \in X$ the evaluation functional $f \to f(x)$ is continuous. For the purposes of this discussion we will assume that X is compact. By the Riesz representation theorem it follows that for a given $x \in X$, there is a function $h_x \in \mathcal{H}$, s.t.

$$\forall f \in \mathcal{H} \quad \langle h_x, f \rangle_{\mathcal{H}} = f(x).$$

We can therefore define the corresponding *kernel function*

$$K(x,y) = \langle h_x, h_y \rangle_{\mathcal{H}}.$$

It follows that $h_x(y) = \langle h_x, h_y \rangle_{\mathcal{H}} = K(x,y)$ and thus $\langle K(x,\cdot), f \rangle = f(x)$. It is clear that $K(x,\cdot) \in \mathcal{H}$.

It is easy to see that $K(x,y)$ is a positive semi-definite kernel as defined below:

Definition: We say that $K(x,y)$, satisfying $K(x,y) = K(y,x)$, is a positive semi-definite kernel if given an arbitrary finite set of points $x_1, \ldots, x_n$, the corresponding $n \times n$ matrix K with $K_{ij} = K(x_i, x_j)$ is positive semi-definite.

Importantly, the converse is also true. Any positive semi-definite kernel $K(x,y)$ gives rise to an RKHS $\mathcal{H}_K$, which can be constructed by considering the space of finite linear combinations of kernels $\sum \alpha_i K(x_i,\cdot)$ and taking completion with respect to the inner product given by $\langle K(x,\cdot), K(y,\cdot) \rangle_{\mathcal{H}_K} = K(x,y)$. See Aronszajn (1950) for details.

We therefore see that reproducing kernel Hilbert spaces of functions on a space X are in *one-to-one correspondence* with positive semidefinite kernels on X.

It can be shown that if the space $\mathcal{H}_K$ is sufficiently rich, that is if for any distinct point $x_1, \ldots, x_n$ there is a function f, s.t. $f(x_1) = 1, f(x_i) = 0, i > 1$, then the corresponding matrix $K_{ij} = K(x_i, x_j)$ is strictly positive definite. For simplicity we will sometimes assume that our RKHS are rich (the corresponding kernels are sometimes called *universal*).

Notation: In what follows, we will use kernel K to denote inner products and norms in the corresponding Hilbert space $\mathcal{H}_K$, that is, we will write $\langle\ ,\ \rangle_K$, $\|\ \|_K$, instead of the more cumbersome $\langle\ ,\ \rangle_{\mathcal{H}_K}$, $\|\ \|_{\mathcal{H}_K}$.

We proceed to endow X with a measure μ (supported on all of X). The corresponding $\mathcal{L}^2_\mu$ Hilbert space inner product is given by

$$\langle f, g \rangle_\mu = \int_X f(x) g(x) d\mu.$$

We can now consider the integral operator L_K corresponding to the kernel K:

$$(L_K f)(x) = \int_X f(y) K(x,y)\, d\mu.$$

It is well-known that if X is a compact space, L_K is a compact operator and is self-adjoint with respect to $\mathcal{L}^2_\mu$. By the spectral theorem, its eigenfunctions $e_1(x), e_2(x), \ldots$, (scaled to norm 1) form an orthonormal basis of $\mathcal{L}^2_\mu$. The spectrum of the operator is discrete and the corresponding eigenvalues $\lambda_1, \lambda_2, \ldots$ are of finite multiplicity, $\lim_{i\to\infty} \lambda_i = 0$.

We see that

$$\langle K(x,\cdot), e_i(\cdot) \rangle_\mu = \lambda_i e_i(x).$$

and therefore $K(x,y) = \sum_i \lambda_i e_i(x) e_i(y)$. Writing a function f in that basis, we have $f = \sum a_i e_i(x)$ and $\langle K(x,\cdot), f(\cdot)\rangle_\mu = \sum_i \lambda_i a_i e_i(x)$.

It is not hard to show that the eigenfunctions e_i are in $\mathcal{H}_K$ (e.g., see the argument below). Thus we see that

$$e_j(x) = \langle K(x,\cdot), e_j(\cdot)\rangle_K = \sum_i \lambda_i e_i(x) \langle e_i, e_j\rangle_K.$$

Therefore $\langle e_i, e_j\rangle_K = 0$, if $i \neq j$, and $\langle e_i, e_i\rangle_K = \frac{1}{\lambda_i}$. On the other hand $\langle e_i, e_j\rangle_\mu = 0$, if $i \neq j$, and $\langle e_i, e_i\rangle_\mu = 1$.

This observation establishes a simple relationship between the Hilbert norms in $\mathcal{H}_K$ and $\mathcal{L}^2_\mu$. We also see that $f = \sum a_i e_i(x) \in \mathcal{H}_K$ if and only if $\sum \frac{a_i^2}{\lambda_i} < \infty$.

Consider now the operator $L_K^{1/2}$. It can be defined as the only positive definite self-adjoint operator, s.t. $L_K = L_K^{1/2} \circ L_K^{1/2}$. Assuming that the series $\tilde{K}(x,y) = \sum_i \sqrt{\lambda_i} e_i(x) e_i(y)$ converges, we can write

$$(L_K^{1/2} f)(x) = \int_X f(y) \tilde{K}(x,y)\, d\mu.$$

It is easy to check that $L_K^{1/2}$ is an isomorphism between $\mathcal{H}$ and $\mathcal{L}^2_\mu$, that is

$$\forall f, g \in \mathcal{H}_K \quad \langle f, g\rangle_\mu = \langle L_K^{1/2} f, L_K^{1/2} g\rangle_K.$$

Therefore $\mathcal{H}_K$ is the image of $L_K^{1/2}$ acting on $\mathcal{L}^2_\mu$.

Lemma 3 *A function $f(x) = \sum_i a_i e_i(x)$ can be represented as $f = L_K g$ for some g if and only if*

$$\sum_{i=1}^{\infty} \frac{a_i^2}{\lambda_i^2} < \infty. \tag{6}$$

Proof Suppose $f = L_K g$. Write $g(x) = \sum_i b_i e_i(x)$. We know that $g \in L^2_\mu$ if and only if $\sum_i b_i^2 < \infty$. Since $L_K(\sum_i b_i e_i) = \sum_i b_i \lambda_i e_i = \sum_i a_i e_i$, we obtain $a_i = b_i \lambda_i$. Therefore $\sum_{i=1}^{\infty} \frac{a_i^2}{\lambda_i^2} < \infty$.

Conversely, if the condition in the inequality 6 is satisfied, $f = L_k g$, where $g = \sum \frac{a_i}{\lambda_i} e_i$. ■

3.2 Proof of Theorems

Now let us recall the Equation 2:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \|f\|_I^2.$$

We have an RKHS $\mathcal{H}_K$ and the probability distribution μ which is supported on $\mathcal{M} \subset X$. We denote by $\mathcal{S}$ the linear space, which is the closure with respect to the RKHS norm of $\mathcal{H}_K$, of the linear span of kernels centered at points of $\mathcal{M}$:

$$\mathcal{S} = \overline{\operatorname{span}\{K(x,\cdot) \,|\, x \in \mathcal{M}\}}.$$

Notation. By the subscript $\mathcal{M}$ we will denote the restriction to $\mathcal{M}$. For example, by $\mathcal{S}_\mathcal{M}$ we denote functions in $\mathcal{S}$ restricted to the manifold $\mathcal{M}$. It can be shown (Aronszajn, 1950, p. 350) that the space $(\mathcal{H}_K)_\mathcal{M}$ of functions from $\mathcal{H}_K$ restricted to $\mathcal{M}$ is an RKHS with the kernel $K_\mathcal{M}$, in other words $(\mathcal{H}_K)_\mathcal{M} = \mathcal{H}_{K_\mathcal{M}}$.

Lemma 4 *The following properties of $\mathcal{S}$ hold:*

1. *$\mathcal{S}$ with the inner product induced by $\mathcal{H}_K$ is a Hilbert space.*
2. *$\mathcal{S}_\mathcal{M} = (\mathcal{H}_K)_\mathcal{M}$.*
3. *The orthogonal complement $\mathcal{S}^\perp$ to $\mathcal{S}$ in $\mathcal{H}_K$ consists of all functions vanishing on $\mathcal{M}$.*

Proof

1. From the definition of $\mathcal{S}$ it is clear by that $\mathcal{S}$ is a complete subspace of $\mathcal{H}_K$.

2. We give a convergence argument similar to the one found in Aronszajn (1950). Since $(\mathcal{H}_K)_\mathcal{M} = \mathcal{H}_{K_\mathcal{M}}$ any function $f_\mathcal{M}$ in it can be written as $f_\mathcal{M} = \lim_{n\to\infty} f_{\mathcal{M},n}$, where $f_{\mathcal{M},n} = \sum_i \alpha_{in} K_\mathcal{M}(x_{in},\cdot)$ is a sum of kernel functions.

Consider the corresponding sum $f_n = \sum_i \alpha_{in} K(x_{in},\cdot)$. From the definition of the norm we see that $\|f_n - f_k\|_K = \|f_{\mathcal{M},n} - f_{\mathcal{M},k}\|_{K_\mathcal{M}}$ and therefore f_n is a Cauchy sequence. Thus $f = \lim_{n\to\infty} f_n$ exists and its restriction to $\mathcal{M}$ must equal $f_\mathcal{M}$. This shows that $(\mathcal{H}_K)_\mathcal{M} \subset \mathcal{S}_\mathcal{M}$. The other direction follows by a similar argument.

3. Let $g \in \mathcal{S}^\perp$. By the reproducing property for any $x \in \mathcal{M}$, $g(x) = \langle K(x,\cdot), g(\cdot)\rangle_K = 0$ and therefore any function in $\mathcal{S}^\perp$ vanishes on $\mathcal{M}$. On the other hand, if g vanishes on $\mathcal{M}$ it is perpendicular to each $K(x,\cdot), x \in \mathcal{M}$ and is therefore perpendicular to the closure of their span $\mathcal{S}$.

■

Lemma 5 *Assume that the intrinsic norm is such that for any $f,g \in \mathcal{H}_K$, $(f-g)|_\mathcal{M} \equiv 0$ implies that $\|f\|_I = \|g\|_I$. Then assuming that the solution f^* of the optimization problem in Equation 2 exists, $f^* \in \mathcal{S}$.*

Proof Any $f \in \mathcal{H}_K$ can be written as $f = f_\mathcal{S} + f_\mathcal{S}^\perp$, where $f_\mathcal{S}$ is the projection of f to $\mathcal{S}$ and $f_\mathcal{S}^\perp$ is its orthogonal complement.

For any $x \in M$ we have $K(x,\cdot) \in \mathcal{S}$. By the previous Lemma $f_\mathcal{S}^\perp$ vanishes on $\mathcal{M}$. We have $f(x_i) = f_\mathcal{S}(x_i)\ \ \forall_i$ and by assumption $\|f_\mathcal{S}\|_I = \|f\|_I$.

On the other hand, $\|f\|_K^2 = \|f_\mathcal{S}\|_K^2 + \|f_\mathcal{S}^\perp\|_K^2$ and therefore $\|f\|_K \geq \|f_\mathcal{S}\|_K$. It follows that the minimizer f^* is in $\mathcal{S}$. ■

As a direct corollary of these consideration, we obtain the following

Proposition 6 *If $\|f\|_I = \|f\|_{K_\mathcal{M}}$ then the minimizer of Equation 2 is identical to that of the usual regularization problem (Equation 1) although with a different regularization parameter ($\lambda_A + \lambda_I$).*

We can now restrict our attention to the study of $\mathcal{S}$. While it is clear that the right-hand side of Equation 3 lies in $\mathcal{S}$, not every element in $\mathcal{S}$ can be written in that form. For example, $K(x,\cdot)$, where x is not one of the data points x_i cannot generally be written as

$$\sum_{i=1}^{l} \alpha_i K(x_i,x) + \int_\mathcal{M} \alpha(y) K(x,y)\, d\mu.$$

We will now assume that for $f \in \mathcal{S}$

$$\|f\|_I^2 = \langle f, Df \rangle_{\mathcal{L}_\mu^2}.$$

We usually assume that D is an appropriate smoothness penalty, such as an inverse integral operator or a differential operator, for example, $Df = \Delta_{\mathcal{M}} f$. The Representer theorem, however, holds under quite mild conditions on D:

Theorem 7 *Let $\|f\|_I^2 = \langle f, Df \rangle_{\mathcal{L}_\mu^2}$ where D is a bounded operator $D : \mathcal{S} \to \mathcal{L}^2_{\mathcal{P}_X}$. Then the solution f^* of the optimization problem in Equation 2 exists and can be written as*

$$f^*(x) = \sum_{i=1}^{l} \alpha_i K(x_i, x) + \int_{\mathcal{M}} \alpha(y) K(x, y)\, d\mathcal{P}_X(y). \tag{7}$$

Proof

For simplicity we will assume that the loss function V is differentiable. This condition can ultimately be eliminated by approximating a non-differentiable function appropriately and passing to the limit.

Put

$$H(f) = \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f(x_i)) + \gamma_A \|f\|_K^2 + \gamma_I \|f\|_I^2.$$

We first show that the solution to Equation 2, f^*, exists and by Lemma 5 belongs to $\mathcal{S}$. It follows easily from Cor. 10 and standard results about compact embeddings of Sobolev spaces (e.g., Adams, 1975) that a ball $\mathcal{B}_r \subset \mathcal{H}_K$, $\mathcal{B}_r = \{f \in \mathcal{S}, s.t.\ \|f\|_K \leq r\}$ is compact in $\mathcal{L}_X^\infty$. Therefore for any such ball the minimizer in that ball f_r^* must exist and belong to $\mathcal{B}_r$. On the other hand, by substituting the zero function

$$H(f_r^*) \leq H(0) = \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, 0).$$

If the loss is actually zero, then zero function is a solution, otherwise

$$\gamma_A \|f_r^*\|_K^2 < \sum_{i=1}^{l} V(x_i, y_i, 0),$$

and hence $f_r^* \in \mathcal{B}_r$, where

$$r = \sqrt{\frac{\sum_{i=1}^{l} V(x_i, y_i, 0)}{\gamma_A}}.$$

Therefore we cannot decrease $H(f^*)$ by increasing r beyond a certain point, which shows that $f^* = f_r^*$ with r as above, which completes the proof of existence. If V is convex, such solution will also be unique.

We proceed to derive the Equation 7. As before, let $e_1, e_2, \ldots$ be the basis associated to the integral operator $(L_K f)(x) = \int_{\mathcal{M}} f(y) K(x, y)\, d\mathcal{P}_X(y)$. Write $f^* = \sum_i a_i e_i(x)$. By substituting f^* into $H(f)$ we obtain:

$$H(f^*) = \frac{1}{l} \sum_{j=1}^{l} V(x_j, y_j, \sum_i a_i e_i(x_i)) + \gamma_A \|f^*\|_K^2 + \gamma_I \|f^*\|_I^2.$$

Assume that V is differentiable with respect to each a_k. We have $\|\sum_i a_i e_i(x)\|_K^2 = \sum_i \frac{a_i^2}{\lambda_i}$. Differentiating with respect to the coefficients a_i yields the following set of equations:

$$0 = \frac{\partial H(f^*)}{\partial a_k} = \frac{1}{l}\sum_{j=1}^{l} e_k(x_j)\partial_3 V(x_j, y_j, \sum_i a_i e_i) + 2\gamma_A \frac{a_k}{\lambda_k} + \gamma_I \langle Df, e_k\rangle + \gamma_I \langle f, De_k\rangle,$$

where $\partial_3 V$ denotes the derivative with respect to the third argument of V.

$\langle Df, e_k\rangle + \langle f, De_k\rangle = \langle (D+D^*)f, e_k\rangle$ and hence

$$a_k = -\frac{\lambda_k}{2\gamma_A l}\sum_{j=1}^{l} e_k(x_j)\partial_3 V(x_j, y_j, f^*) - \frac{\gamma_I}{2\gamma_A}\lambda_k \langle Df^* + D^* f^*, e_k\rangle.$$

Since $f^*(x) = \sum_k a_k e_k(x)$ and recalling that $K(x,y) = \sum_i \lambda_i e_i(x) e_i(y)$

$$f^*(x) = -\frac{1}{2\gamma_A l}\sum_k \sum_{j=1}^{l} \lambda_k e_k(x) e_k(x_j)\partial_3 V(x_j, y_j, f^*) - \frac{\gamma_I}{2\gamma_A}\sum_k \lambda_k \langle Df^* + D^* f^*, e_k\rangle e_k,$$

$$= -\frac{1}{2\gamma_A l}\sum_{j=1}^{l} K(x, x_j)\partial_3 V(x_j, y_j, f^*) - \frac{\gamma_I}{2\gamma_A}\sum_k \lambda_k \langle Df^* + D^* f^*, e_k\rangle e_k.$$

We see that the first summand is a sum of the kernel functions centered at data points. It remains to show that the second summand has an integral representation, that is, can be written as $\int_{\mathcal{M}} \alpha(y) K(x,y)\, d\mathcal{P}_X(y)$, which is equivalent to being in the image of L_K. To verify this we apply Lemma 3. We need that

$$\sum_k \frac{\lambda_k^2 \langle Df^* + D^* f^*, e_k\rangle^2}{\lambda_k^2} = \sum_k \langle Df^* + D^* f^*, e_k\rangle^2 < \infty.$$

Since D, its adjoint operator D^* and hence their sum are bounded the inequality above is satisfied for any function in $\mathcal{S}$. ■

3.3 Manifold Setting[2]

We now show that for the case when $\mathcal{M}$ is a manifold and D is a differential operator, such as the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$, the boundedness condition of Theorem 7 is satisfied. While we consider the case when the manifold has no boundary, the same argument goes through for manifold with boundary, with, for example, Dirichlet's boundary conditions (vanishing at the boundary). Thus the setting of Theorem 7 is very general, applying, among other things, to arbitrary differential operators on compact domains in Euclidean space.

Let $\mathcal{M}$ be a $\mathcal{C}^\infty$ manifold without boundary with an infinitely differentiable embedding in some ambient space X, D a differential operator with $\mathcal{C}^\infty$ coefficients and let μ, be the measure corresponding to some $\mathcal{C}^\infty$ nowhere vanishing volume form on $\mathcal{M}$. We assume that the kernel $K(x,y)$ is also infinitely differentiable.[3] As before for an operator A, A^* denotes the adjoint operator.

2. We thank Peter Constantin and Todd Dupont for help with this section.
3. While we have assumed that all objects are infinitely differentiable, it is not hard to specify the precise differentiability conditions. Roughly speaking, a degree k differential operator D is bounded as an operator $\mathcal{H}_K \to L^2_\mu$, if the kernel $K(x,y)$ has $2k$ derivatives.

Theorem 8 *Under the conditions above D is a bounded operator* $\mathcal{S} \to \mathcal{L}^2_\mu$.

Proof First note that it is enough to show that D is bounded on $\mathcal{H}_{K_\mathcal{M}}$, since D only depends on the restriction $f_\mathcal{M}$. As before, let $L_{K_\mathcal{M}}(f)(x) = \int_\mathcal{M} f(y)K_\mathcal{M}(x,y)\, d\mu$ is the integral operator associated to K_M. Note that D^* is also a differential operator of the same degree as D. The integral operator $L_{K_\mathcal{M}}$ is bounded (compact) from L^2_μ to any Sobolev space H^{sob}. Therefore the operator $L_{K_\mathcal{M}}D$ is also bounded. We therefore see that $DL_{K_\mathcal{M}}D^*$ is bounded $L^2_\mu \to L^2_\mu$. Therefore there is a constant C, s.t. $\langle DL_{K_\mathcal{M}}D^* f, f\rangle_{L^2_\mu} \leq C\|f\|_{L^2_\mu}$.

The square root $T = L^{1/2}_{K_\mathcal{M}}$ of the self-adjoint positive definite operator $L_{K_\mathcal{M}}$ is a self-adjoint positive definite operator as well. Thus $(DT)^* = TD^*$. By definition of the operator norm, for any $\varepsilon > 0$ there exists $f \in L^2_\mu, \|f\|_{L^2_\mu} \leq 1+\varepsilon$, such that

$$\|DT\|^2_{L^2_\mu} = \|TD^*\|^2_{L^2_\mu} \leq \langle TD^* f, TD^* f\rangle_{L^2_\mu} =$$

$$= \langle DLD^* f, f\rangle_{L^2_\mu} \leq \|DLD^*\|_{L^2_\mu}\|f\|^2_{L^2_\mu} \leq C(1+\varepsilon)^2.$$

Therefore the operator $DT : L^2_\mu \to L^2_\mu$ is bounded (and also $\|DT\|_{\mathcal{L}^2_\mu} \leq C$, since ε is arbitrary).

Now recall that T provides an isometry between L^2_μ and $\mathcal{H}_{K_\mathcal{M}}$. That means that for any $g \in \mathcal{H}_{K_\mathcal{M}}$ there is $f \in L^2_\mu$, such that $Tf = g$ and $\|f\|_{L^2_\mu} = \|g\|_{K_\mathcal{M}}$. Thus $\|Dg\|_{\mathcal{L}^2_\mu} = \|DTf\|_{\mathcal{L}^2_\mu} \leq C\|g\|_{K_\mathcal{M}}$, which shows that $T : \mathcal{H}_{K_\mathcal{M}} \to L^2_\mu$ is bounded and concludes the proof. ■

Since $\mathcal{S}$ is a subspace of $\mathcal{H}_K$ the main result follows immediately:

Corollary 9 *D is a bounded operator* $\mathcal{S} \to L^2_\mu$ *and the conditions of Theorem 7 hold.*

Before finishing the theoretical discussion we obtain a useful

Corollary 10 *The operator* $T = L^{1/2}_K$ *on* L^2_μ *is a bounded (and in fact compact) operator* $L^2_\mu \to H^{sob}$, *where* H^{sob} *is an arbitrary Sobolev space.*

Proof Follows from the fact that DT is bounded operator $L^2_\mu \to L^2_\mu$ for an arbitrary differential operator D and standard results on compact embeddings of Sobolev spaces (see, for example, Adams, 1975). ■

3.4 The Representer Theorem for the Empirical Case

In the case when $\mathcal{M}$ is unknown and sampled via labeled and unlabeled examples, the Laplace-Beltrami operator on $\mathcal{M}$ may be approximated by the Laplacian of the data adjacency graph (see Belkin, 2003; Bousquet et al., 2004, for some discussion). A regularizer based on the graph Laplacian leads to the optimization problem posed in Equation 4. We now provide a proof of Theorem 2 which states that the solution to this problem admits a representation in terms of an expansion over labeled and unlabeled points. The proof is based on a simple orthogonality argument (e.g., Scholkopf and Smola, 2002).

Proof *(Theorem 2)* Any function $f \in \mathcal{H}_K$ can be uniquely decomposed into a component $f_{||}$ in the linear subspace spanned by the kernel functions $\{K(x_i,\cdot)\}_{i=1}^{l+u}$, and a component $f_\perp$ orthogonal to it. Thus,

$$f = f_{||} + f_\perp = \sum_{i=1}^{l+u} \alpha_i K(x_i,\cdot) + f_\perp.$$

By the reproducing property, as the following arguments show, the evaluation of f on any data point x_j, $1 \le j \le l+u$ is independent of the orthogonal component $f_\perp$:

$$f(x_j) = \langle f, K(x_j,\cdot)\rangle = \langle \sum_{i=1}^{l+u} \alpha_i K(x_i,\cdot), K(x_j,\cdot)\rangle + \langle f_\perp, K(x_j,\cdot)\rangle.$$

Since the second term vanishes, and $\langle K(x_i,\cdot), K(x_j,\cdot)\rangle = K(x_i,x_j)$, it follows that $f(x_j) = \sum_{i=1}^{l+u} \alpha_i K(x_i,x_j)$. Thus, the empirical terms involving the loss function and the intrinsic norm in the optimization problem in Equation 4 depend only on the value of the coefficients $\{\alpha_i\}_{i=1}^{l+u}$ and the gram matrix of the kernel function.

Indeed, since the orthogonal component only increases the norm of f in $\mathcal{H}_K$:

$$\|f\|_K^2 = \|\sum_{i=1}^{l+u} \alpha_i K(x_i,\cdot)\|_K^2 + \|f_\perp\|_K^2 \ge \|\sum_{i=1}^{l+u} \alpha_i K(x_i,\cdot)\|_K^2.$$

It follows that the minimizer of problem 4 must have $f_\perp = 0$, and therefore admits a representation $f^*(\cdot) = \sum_{i=1}^{l+u} \alpha_i K(x_i,\cdot)$. ■

The simple form of the minimizer, given by this theorem, allows us to translate our extrinsic and intrinsic regularization framework into optimization problems over the finite dimensional space of coefficients $\{\alpha_i\}_{i=1}^{l+u}$, and invoke the machinery of kernel based algorithms. In the next section, we derive these algorithms, and explore their connections to other related work.

4. Algorithms

We now discuss standard regularization algorithms (RLS and SVM) and present their extensions (LapRLS and LapSVM respectively). These are obtained by solving the optimization problems posed in Equation 4) for different choices of cost function V and regularization parameters γ_A, γ_I. To fix notation, we assume we have l labeled examples $\{(x_i,y_i)\}_{i=1}^{l}$ and u unlabeled examples $\{x_j\}_{j=l+1}^{j=l+u}$. We use K interchangeably to denote the kernel function or the Gram matrix.

4.1 Regularized Least Squares

The regularized least squares algorithm is a fully supervised method where we solve:

$$\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} (y_i - f(x_i))^2 + \gamma \|f\|_K^2.$$

The classical Representer Theorem can be used to show that the solution is of the following form:

$$f^\star(x) = \sum_{i=1}^{l} \alpha_i^\star K(x,x_i).$$

Substituting this form in the problem above, we arrive at following convex differentiable objective function of the l-dimensional variable $\alpha = [\alpha_1 \dots \alpha_l]^T$:

$$\alpha^* = \text{argmin} \frac{1}{l}(Y - K\alpha)^T(Y - K\alpha) + \gamma\alpha^T K\alpha,$$

where K is the $l \times l$ gram matrix $K_{ij} = K(x_i, x_j)$ and Y is the label vector $Y = [y_1 \dots y_l]^T$.

The derivative of the objective function vanishes at the minimizer:

$$\frac{1}{l}(Y - K\alpha^*)^T(-K) + \gamma K\alpha^* = 0,$$

which leads to the following solution:

$$\alpha^* = (K + \gamma lI)^{-1}Y.$$

4.2 Laplacian Regularized Least Squares (LapRLS)

The Laplacian regularized least squares algorithm solves the optimization problem in Equation 4) with the squared loss function:

$$\min_{f \in \mathcal{H}_K} \frac{1}{l}\sum_{i=1}^{l}(y_i - f(x_i))^2 + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2}\mathbf{f}^T L\mathbf{f}.$$

As before, the Representer Theorem can be used to show that the solution is an expansion of kernel functions over both the labeled and the unlabeled data:

$$f^\star(x) = \sum_{i=1}^{l+u} \alpha_i^\star K(x, x_i).$$

Substituting this form in the equation above, as before, we arrive at a convex differentiable objective function of the $l+u$-dimensional variable $\alpha = [\alpha_1 \dots \alpha_{l+u}]^T$:

$$\alpha^* = \underset{\alpha \in \mathbb{R}^{l+u}}{\text{argmin}} \frac{1}{l}(Y - JK\alpha)^T(Y - JK\alpha) + \gamma_A\alpha^T K\alpha + \frac{\gamma_I}{(u+l)^2}\alpha^T KLK\alpha,$$

where K is the $(l+u) \times (l+u)$ Gram matrix over labeled and unlabeled points; Y is an $(l+u)$ dimensional label vector given by: $Y = [y_1, \dots, y_l, 0, \dots, 0]$ and J is an $(l+u) \times (l+u)$ diagonal matrix given by $J = \text{diag}(1, \dots, 1, 0, \dots, 0)$ with the first l diagonal entries as 1 and the rest 0.

The derivative of the objective function vanishes at the minimizer:

$$\frac{1}{l}(Y - JK\alpha)^T(-JK) + (\gamma_A K + \frac{\gamma_I l}{(u+l)^2}KLK)\alpha = 0,$$

which leads to the following solution:

$$\alpha^* = (JK + \gamma_A lI + \frac{\gamma_I l}{(u+l)^2}LK)^{-1}Y. \tag{8}$$

Note that when $\gamma_I = 0$, Equation 8) gives zero coefficients over unlabeled data, and the coefficients over the labeled data are exactly those for standard RLS.

4.3 Support Vector Machine Classification

Here we outline the SVM approach to binary classification problems. For SVMs, the following problem is solved:

$$\min_{f\in\mathcal{H}_K} \frac{1}{l}\sum_{i=1}^{l}(1-y_i f(x_i))_+ + \gamma\|f\|_K^2,$$

where the hinge loss is defined as: $(1-yf(x))_+ = \max(0,1-yf(x))$ and the labels $y_i \in \{-1,+1\}$.

Again, the solution is given by:

$$f^\star(x) = \sum_{i=1}^{l}\alpha_i^\star K(x,x_i). \tag{9}$$

Following SVM expositions, the above problem can be equivalently written as:

$$\begin{aligned} \min_{f\in\mathcal{H}_K,\xi_i\in\mathbb{R}} \frac{1}{l}\sum_{i=1}^{l}\xi_i + \gamma\|f\|_K^2 \\ \text{subject to: } y_i f(x_i) \geq 1-\xi_i \quad i=1,\ldots,l \\ \xi_i \geq 0 \quad i=1,\ldots,l. \end{aligned}$$

Using the Lagrange multipliers technique, and benefiting from strong duality, the above problem has a simpler quadratic dual program in the Lagrange multipliers $\beta = [\beta_1,\ldots,\beta_l]^T \in \mathbb{R}^l$:

$$\begin{aligned} \beta^\star &= \max_{\beta\in\mathbb{R}^l}\sum_{i=1}^{l}\beta_i - \frac{1}{2}\beta^T Q\beta \\ \text{subject to:} \quad & \sum_{i=1}^{l} y_i\beta_i = 0 \\ & 0 \leq \beta_i \leq \frac{1}{l} \quad i=1,\ldots,l. \end{aligned}$$

where the equality constraint arises due to an unregularized bias term that is often added to the sum in Equation 9, and the following notation is used:

$$\begin{aligned} Y &= \text{diag}(y_1,y_2,...,y_l), \\ Q &= Y\left(\frac{K}{2\gamma}\right)Y, \\ \alpha^\star &= \frac{Y\beta^\star}{2\gamma}. \end{aligned}$$

Here again, K is the gram matrix over labeled points. SVM practitioners may be familiar with a slightly different parameterization involving the C parameter: $C = \frac{1}{2\gamma l}$ is the weight on the hinge loss term (instead of using a weight γ on the norm term in the optimization problem). The C parameter appears as the upper bound (instead of $\frac{1}{l}$) on the values of β in the quadratic program. For additional details on the derivation and alternative formulations of SVMs, see Scholkopf and Smola (2002); Rifkin (2002).

4.4 Laplacian Support Vector Machines

By including the intrinsic smoothness penalty term, we can extend SVMs by solving the following problem:

$$\min_{f\in\mathcal{H}_K} \frac{1}{l}\sum_{i=1}^{l}(1-y_i f(x_i))_+ + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2}\mathbf{f}^T L\mathbf{f}.$$

By the representer theorem,as before, the solution to the problem above is given by:

$$f^\star(x) = \sum_{i=1}^{l+u}\alpha_i^\star K(x,x_i).$$

Often in SVM formulations, an unregularized bias term b is added to the above form. Again, the primal problem can be easily seen to be the following:

$$\begin{aligned}\min_{\alpha\in\mathbb{R}^{l+u},\xi\in\mathbb{R}^l} & \frac{1}{l}\sum_{i=1}^{l}\xi_i + \gamma_A\alpha^T K\alpha + \frac{\gamma_I}{(u+l)^2}\alpha^T KLK\alpha \\ \text{subject to: } & y_i\Big(\sum_{j=1}^{l+u}\alpha_j K(x_i,x_j)+b\Big) \geq 1-\xi_i, \quad i=1,\ldots,l \\ & \xi_i \geq 0 \quad i=1,\ldots,l.\end{aligned}$$

Introducing the Lagrangian, with β_i,ζ_i as Lagrange multipliers:

$$\begin{aligned}L(\alpha,\xi,b,\beta,\zeta) &= \frac{1}{l}\sum_{i=1}^{l}\xi_i + \frac{1}{2}\alpha^T(2\gamma_A K + 2\frac{\gamma_A}{(l+u)^2}KLK)\alpha \\ &\quad -\sum_{i=1}^{l}\beta_i\Big(y_i\Big(\sum_{j=1}^{l+u}\alpha_j K(x_i,x_j)+b\Big)-1+\xi_i\Big) - \sum_{i=1}^{l}\zeta_i\xi_i.\end{aligned}$$

Passing to the dual requires the following steps:

$$\begin{aligned}\frac{\partial L}{\partial b} = 0 &\implies \sum_{i=1}^{l}\beta_i y_i = 0, \\ \frac{\partial L}{\partial \xi_i} = 0 &\implies \frac{1}{l}-\beta_i-\zeta_i = 0, \\ &\implies 0\leq\beta_i\leq\frac{1}{l} \quad (\xi_i,\zeta_i \text{ are non-negative}).\end{aligned}$$

Using above identities, we formulate a reduced Lagrangian:

$$\begin{aligned}L^R(\alpha,\beta) &= \frac{1}{2}\alpha^T(2\gamma_A K + 2\frac{\gamma_I}{(u+l)^2}KLK)\alpha - \sum_{i=1}^{l}\beta_i\Big(y_i\sum_{j=1}^{l+u}\alpha_j K(x_i,x_j)-1\Big), \\ &= \frac{1}{2}\alpha^T(2\gamma_A K + 2\frac{\gamma_I}{(u+l)^2}KLK)\alpha - \alpha^T KJ^T Y\beta + \sum_{i=1}^{l}\beta_i,\end{aligned}$$

where $J = [I \quad 0]$ is an $l \times (l+u)$ matrix with I as the $l \times l$ identity matrix (assuming the first l points are labeled) and $Y = \text{diag}(y_1, y_2, ..., y_l)$.

Taking derivative of the reduced Lagrangian with respect to α:

$$\frac{\partial L^R}{\partial \alpha} = (2\gamma_A K + 2\frac{\gamma_I}{(u+l)^2} KLK)\alpha - KJ^T Y\beta.$$

This implies:

$$\alpha = (2\gamma_A I + 2\frac{\gamma_I}{(u+l)^2} LK)^{-1} J^T Y\beta^\star. \tag{10}$$

Note that the relationship between α and β is no longer as simple as the SVM algorithm. In particular, the $(l+u)$ expansion coefficients are obtained by solving a linear system involving the l dual variables that will appear in the SVM dual problem.

Substituting back in the reduced Lagrangian we get:

$$\beta^* = \max_{\beta \in \mathbb{R}^l} \sum_{i=1}^{l} \beta_i - \frac{1}{2}\beta^T Q\beta \tag{11}$$

$$\text{subject to:} \quad \sum_{i=1}^{l} \beta_i y_i = 0$$

$$0 \leq \beta_i \leq \frac{1}{l} \; i = 1, \ldots, l \tag{12}$$

where

$$Q = YJK(2\gamma_A I + 2\frac{\gamma_I}{(l+u)^2} LK)^{-1} J^T Y.$$

Laplacian SVMs can be implemented by using a standard SVM solver with the quadratic form induced by the above matrix, and using the solution to obtain the expansion coefficients by solving the linear system in Equation 10.

Note that when $\gamma_I = 0$, the SVM QP and Equations 11 and 10, give zero expansion coefficients over the unlabeled data. The expansion coefficients over the labeled data and the Q matrix are as in standard SVM, in this case.

The manifold regularization algorithms are summarized in the Table 1.

Efficiency Issues: It is worth noting that our algorithms compute the inverse of a dense Gram matrix which leads to $O((l+u)^3)$ complexity. This may be impractical for large data sets. In the case of linear kernels, instead of using Equation 5, we can directly write $f^\star(x) = w^T x$ and solve for the weight vector w using a primal optimization method. This is much more efficient when the data is low-dimensional. For highly sparse data sets, for example, in text categorization problems, effective conjugate gradient schemes can be used in a large scale implementation, as outlined in Sindhwani et al. (2006). For the non-linear case, one may obtain approximate solutions (e.g., using greedy, matching pursuit techniques) where the optimization problem is solved over the span of a small set of basis functions instead of using the full representation in Equation 5. We note these directions for future work. In section 5, we evaluate the empirical performance of our algorithms with exact computations as outlined in Table 1 with non-linear kernels. For other recent work addressing

	Manifold Regularization algorithms
Input:	l labeled examples $\{(x_i,y_i)\}_{i=1}^{l}$, u unlabeled examples $\{x_j\}_{j=l+1}^{l+u}$
Output:	Estimated function $f:\mathbb{R}^n \to \mathbb{R}$
Step 1	► Construct data adjacency graph with $(l+u)$ nodes using, for example, k nearest neighbors or a graph kernel. Choose edge weights W_{ij}, for example, binary weights or heat kernel weights $W_{ij} = e^{-\|x_i - x_j\|^2/4t}$.
Step 2	► Choose a kernel function $K(x,y)$. Compute the Gram matrix $K_{ij} = K(x_i,x_j)$.
Step 3	► Compute graph Laplacian matrix: $L = D - W$ where D is a diagonal matrix given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$.
Step 4	► Choose γ_A and γ_I.
Step 5	► Compute α^* using Equation 8 for squared loss (Laplacian RLS) or using Equations 11 and 10 together with the SVM QP solver for soft margin loss (Laplacian SVM).
Step 6	► Output function $f^*(x) = \sum_{i=1}^{l+u} \alpha_i^* K(x_i,x)$.

Table 1: A summary of the algorithms

scalability issues in semi-supervised learning, see, example, Tsang and Kwok. (2005); Bengio et al. (2004).

4.5 Related Work and Connections to Other Algorithms

In this section we survey various approaches to semi-supervised and transductive learning and highlight connections of manifold regularization to other algorithms.

Transductive SVM (TSVM) (Vapnik, 1998; Joachims, 1999): TSVMs are based on the following optimization principle:

$$f^* = \underset{\substack{f \in \mathcal{H}_K \\ y_{l+1},\ldots y_{l+u}}}{\text{argmin}} \; C\sum_{i=1}^{l}(1-y_i f(x_i))_+ + C^*\sum_{i=l+1}^{l+u}(1-y_i f(x_i))_+ + \|f\|_K^2,$$

which proposes a joint optimization of the SVM objective function over binary-valued labels on the unlabeled data and functions in the RKHS. Here, C, C^* are parameters that control the relative hinge-loss over labeled and unlabeled sets. The joint optimization is implemented in Joachims (1999) by first using an inductive SVM to label the unlabeled data and then iteratively solving SVM quadratic programs, at each step switching labels to improve the objective function. However this procedure is susceptible to local minima and requires an unknown, possibly large number of label switches before converging. Note that even though TSVM were inspired by transductive inference, they do provide an out-of-sample extension.

Semi-Supervised SVMs (S^3VM) (Bennett and Demiriz, 1999; Fung and Mangasarian, 2001): S^3VM incorporate unlabeled data by including the minimum hinge-loss for the two choices of

labels for each unlabeled example. This is formulated as a mixed-integer program for linear SVMs in Bennett and Demiriz (1999) and is found to be intractable for large amounts of unlabeled data. Fung and Mangasarian (2001) reformulate this approach as a concave minimization problem which is solved by a successive linear approximation algorithm. The presentation of these algorithms is restricted to the linear case.

Measure-Based Regularization (Bousquet et al., 2004): The conceptual framework of this work is closest to our approach. The authors consider a gradient based regularizer that penalizes variations of the function more in high density regions and less in low density regions leading to the following optimization principle:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{l} V(f(x_i), y_i) + \gamma \int_X \langle \nabla f(x), \nabla f(x) \rangle p(x) dx,$$

where p is the density of the marginal distribution P_X. The authors observe that it is not straightforward to find a kernel for arbitrary densities p, whose associated RKHS norm is

$$\int \langle \nabla f(x), \nabla f(x) \rangle p(x) dx.$$

Thus, in the absence of a representer theorem, the authors propose to perform minimization of the regularized loss on a fixed set of basis functions chosen apriori, that is, $\mathcal{F} = \{\sum_{i=1}^{q} \alpha_i \phi_i\}$. For the hinge loss, this paper derives an SVM quadratic program in the coefficients $\{\alpha_i\}_{i=1}^{q}$ whose Q matrix is calculated by computing q^2 integrals over gradients of the basis functions. However the algorithm does not demonstrate performance improvements in real world experiments. It is also worth noting that while Bousquet et al. (2004) use the gradient $\nabla f(x)$ in the ambient space, we use the gradient over a submanifold $\nabla_{\mathcal{M}} f$ for penalizing the function. In a situation where the data truly lies on or near a submanifold $\mathcal{M}$, the difference between these two penalizers can be significant since smoothness in the normal direction to the data manifold is irrelevant to classification or regression.

Graph-Based Approaches See, for example, Blum and Chawla (2001); Chapelle et al. (2003); Szummer and Jaakkola (2002); Zhou et al. (2004); Zhu et al. (2003, 2005); Kemp et al. (2004); Joachims (2003); Belkin and Niyogi (2003b): A variety of graph-based methods have been proposed for transductive inference. However, these methods do not provide an out-of-sample extension. In Zhu et al. (2003), nearest neighbor labeling for test examples is proposed once unlabeled examples have been labeled by transductive learning. In Chapelle et al. (2003), test points are approximately represented as a linear combination of training and unlabeled points in the feature space induced by the kernel. For graph regularization and label propagation see (Smola and Kondor, 2003; Belkin et al., 2004; Zhu et al., 2003). Smola and Kondor (2003) discusses the construction of a canonical family of graph regularizers based on the graph Laplacian. Zhu et al. (2005) presents a non-parametric construction of graph regularizers.

Manifold regularization provides natural out-of-sample extensions to several graph-based approaches. These connections are summarized in Table 2.

We also note the recent work (Delalleau et al., 2005) on out-of-sample extensions for semi-supervised learning where an induction formula is derived by assuming that the addition of a test point to the graph does not change the transductive solution over the unlabeled data.

Cotraining (Blum and Mitchell, 1998): The cotraining algorithm was developed to integrate abundance of unlabeled data with availability of multiple sources of information in domains like web-page classification. Weak learners are trained on labeled examples and their predictions on

subsets of unlabeled examples are used to mutually expand the training set. Note that this setting may not be applicable in several cases of practical interest where one does not have access to multiple information sources.

Bayesian Techniques See, for example, Nigam et al. (2000); Seeger (2001); Corduneanu and Jaakkola (2003). An early application of semi-supervised learning to Text classification appeared in Nigam et al. (2000) where a combination of EM algorithm and Naive-Bayes classification is proposed to incorporate unlabeled data. Seeger (2001) provides a detailed overview of Bayesian frameworks for semi-supervised learning. The recent work in Corduneanu and Jaakkola (2003) formulates a new information-theoretic principle to develop a regularizer for conditional log-likelihood.

Parameters	Corresponding algorithms (square loss or hinge loss)
$\gamma_A \geq 0 \;\; \gamma_I \geq 0$	Manifold Regularization
$\gamma_A \geq 0 \;\; \gamma_I = 0$	Standard Regularization (RLS or SVM)
$\gamma_A \to 0 \;\; \gamma_I > 0$	Out-of-sample extension for Graph Regularization (RLS or SVM)
$\gamma_A \to 0 \;\; \gamma_I \to 0$ $\gamma_I \gg \gamma_A$	Out-of-sample extension for Label Propagation (RLS or SVM)
$\gamma_A \to 0 \;\; \gamma_I = 0$	Hard margin SVM or Interpolated RLS

Table 2: Connections of manifold regularization to other algorithms

5. Experiments

We performed experiments on a synthetic data set and three real world classification problems arising in visual and speech recognition, and text categorization. Comparisons are made with inductive methods (SVM, RLS). We also compare Laplacian SVM with transductive SVM. All software and data sets used for these experiments will be made available at:
http://www.cs.uchicago.edu/∼vikass/manifoldregularization.html.

For further experimental benchmark studies and comparisons with numerous other methods, we refer the reader to Chapelle et al. (2006); Sindhwani et al. (2006, 2005).

5.1 Synthetic Data: Two Moons Data Set

The two moons data set is shown in Figure 2. The data set contains 200 examples with only 1 labeled example for each class. Also shown are the decision surfaces of Laplacian SVM for increasing values of the intrinsic regularization parameter γ_I. When $\gamma_I = 0$, Laplacian SVM disregards unlabeled data and returns the SVM decision boundary which is fixed by the location of the two labeled points. As γ_I is increased, the intrinsic regularizer incorporates unlabeled data and causes the decision surface to appropriately adjust according to the geometry of the two classes. In Figure 3, the best decision surfaces across a wide range of parameter settings are also shown for SVM, transductive SVM and Laplacian SVM. Figure 3 demonstrates how TSVM fails to find the optimal solution, probably since it gets stuck in a local minimum. The Laplacian SVM decision boundary seems to be intuitively most satisfying.

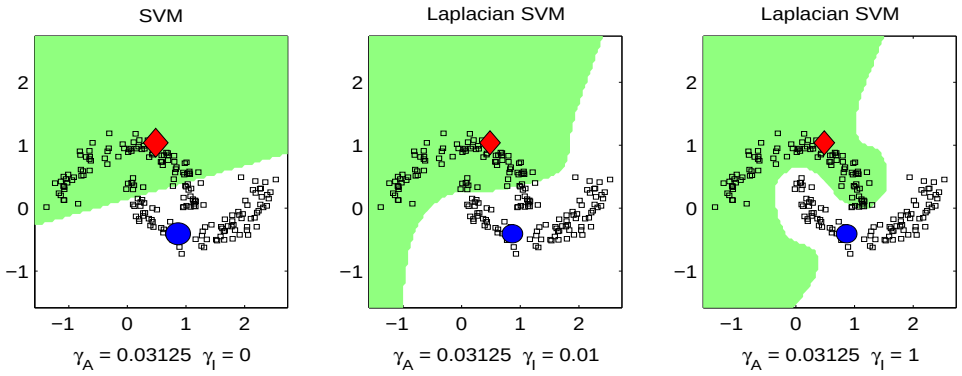


Figure 2: Laplacian SVM with RBF kernels for various values of γ_I. Labeled points are shown in color, other points are unlabeled.

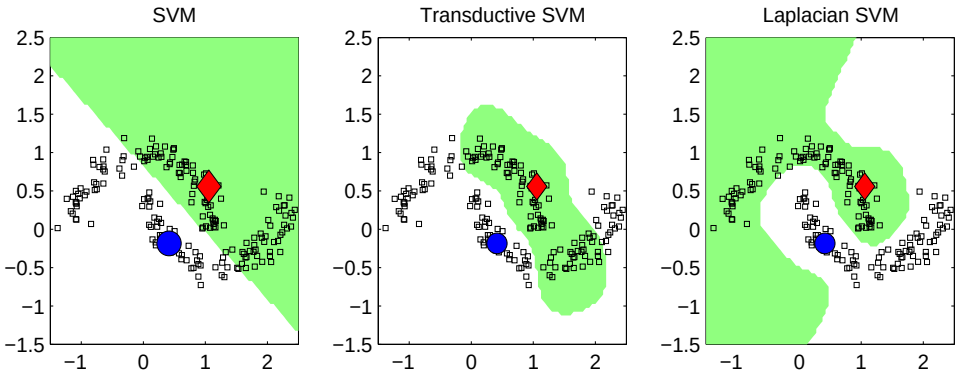


Figure 3: Two Moons data set: Best decision surfaces using RBF kernels for SVM, TSVM and Laplacian SVM. Labeled points are shown in color, other points are unlabeled.

5.2 Handwritten Digit Recognition

In this set of experiments we applied Laplacian SVM and Laplacian RLS algorithms to 45 binary classification problems that arise in pairwise classification of handwritten digits. The first 400 images for each digit in the USPS training set (preprocessed using PCA to 100 dimensions) were taken to form the training set. The remaining images formed the test set. 2 images for each class were randomly labeled (l=2) and the rest were left unlabeled (u=398). Following Scholkopf et al. (1995), we chose to train classifiers with polynomial kernels of degree 3, and set the weight on the regularization term for inductive methods as $\gamma l = 0.05 (C = 10)$. For manifold regularization, we chose to split the same weight in the ratio $1:9$ so that $\gamma_A l = 0.005$, $\frac{\gamma_I l}{(u+l)^2} = 0.045$. The observations reported in this section hold consistently across a wide choice of parameters.

In Figure 4, we compare the error rates of manifold regularization algorithms, inductive classifiers and TSVM, at the break-even points in the precision-recall curves for the 45 binary classi-

fication problems. These results are averaged over 10 random choices of labeled examples. The following comments can be made: (a) manifold regularization results in significant improvements over inductive classification, for both RLS and SVM, and either compares well or significantly outperforms TSVM across the 45 classification problems. Note that TSVM solves multiple quadratic programs in the size of the labeled and unlabeled sets whereas LapSVM solves a single QP (Equation 11) in the size of the labeled set, followed by a linear system (Equation 10). This resulted in substantially faster training times for LapSVM in this experiment. (b) Scatter plots of performance on test and unlabeled data sets, in the bottom row of Figure 4, confirm that the out-of-sample extension is good for both LapRLS and LapSVM. (c) Also shown, in the rightmost scatter plot in the bottom row of Figure 4, are standard deviation of error rates obtained by LapSVM and TSVM. We found LapSVM to be significantly more stable than the inductive methods and TSVM, with respect to choice of the labeled data. In Figure 5, we demonstrate the benefit of unlabeled data as a function of the number of labeled examples.

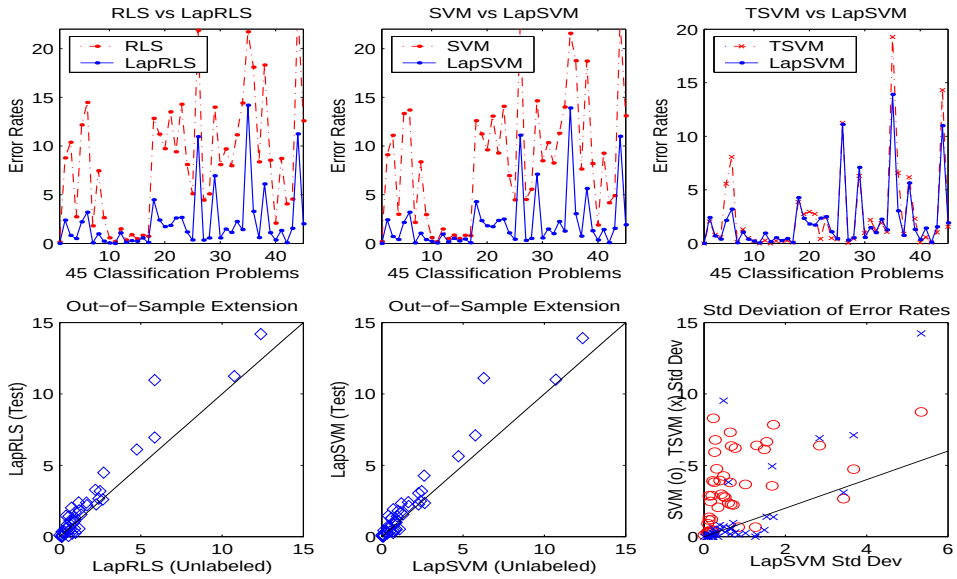


Figure 4: USPS Experiment: (Top row) Error rates at precision-recall break-even points for 45 binary classification problems. (Bottom row) Scatter plots of error rates on test and unlabeled data for Laplacian RLS, Laplacian SVM; and standard deviations in test errors of Laplacian SVM and TSVM.

Method	SVM	TSVM	LapSVM	RLS	LapRLS
Error	23.6	26.5	12.7	23.6	12.7

Table 3: USPS Experiment: one-versus-rest multiclass error rates

We also performed one-vs-rest multiclass experiments on the USPS test set with $l = 50$ and $u = 1957$ with 10 random splits as provided by Chapelle and Zien (2005). The mean error rates

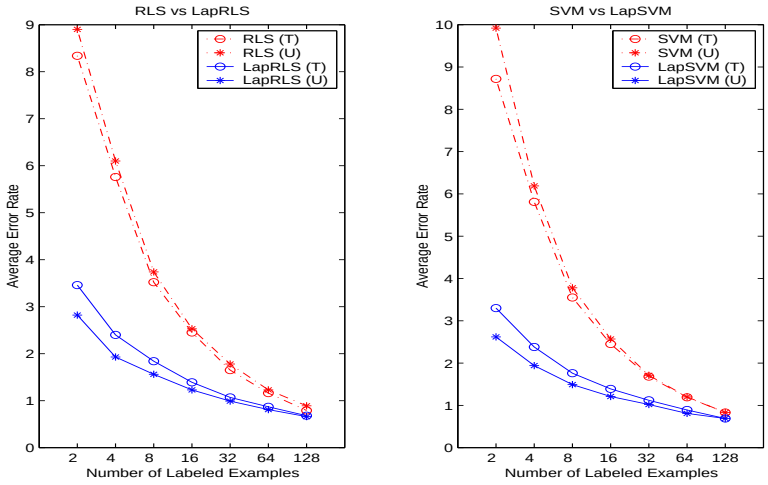


Figure 5: USPS Experiment: mean error rate at precision-recall break-even points as a function of number of labeled points (T: test set, U: unlabeled set)

in predicting labels of unlabeled data are reported in Table 3. In this experiment, TSVM actually performs worse than the SVM baseline probably since local minima problems become severe in a multi-class setting. For several other experimental observations and comparisons on this data set, see Sindhwani et al. (2005).

5.3 Spoken Letter Recognition

This experiment was performed on the Isolet database of letters of the English alphabet spoken in isolation (available from the UCI machine learning repository). The data set contains utterances of 150 subjects who spoke the name of each letter of the English alphabet twice. The speakers are grouped into 5 sets of 30 speakers each, referred to as isolet1 through isolet5. For the purposes of this experiment, we chose to train on the first 30 speakers (isolet1) forming a training set of 1560 examples, and test on isolet5 containing 1559 examples (1 utterance is missing in the database due to poor recording). We considered the task of classifying the first 13 letters of the English alphabet from the last 13. We considered 30 binary classification problems corresponding to 30 splits of the training data where all 52 utterances of one speaker were labeled and all the rest were left unlabeled. The test set is composed of entirely new speakers, forming the separate group isolet5.

We chose to train with RBF kernels of width $\sigma = 10$ (this was the best value among several settings with respect to 5-fold cross-validation error rates for the fully supervised problem using standard SVM). For SVM and RLS we set $\gamma l = 0.05$ ($C = 10$) (this was the best value among several settings with respect to mean error rates over the 30 splits). For Laplacian RLS and Laplacian SVM we set $\gamma_A l = \frac{\gamma_I l}{(u+l)^2} = 0.005$.

In Figure 6, we compare these algorithms. The following comments can be made: (a) LapSVM and LapRLS make significant performance improvements over inductive methods and TSVM, for predictions on unlabeled speakers that come from the same group as the labeled speaker, over all choices of the labeled speaker. (b) On Isolet5 which comprises of a separate group of speakers,

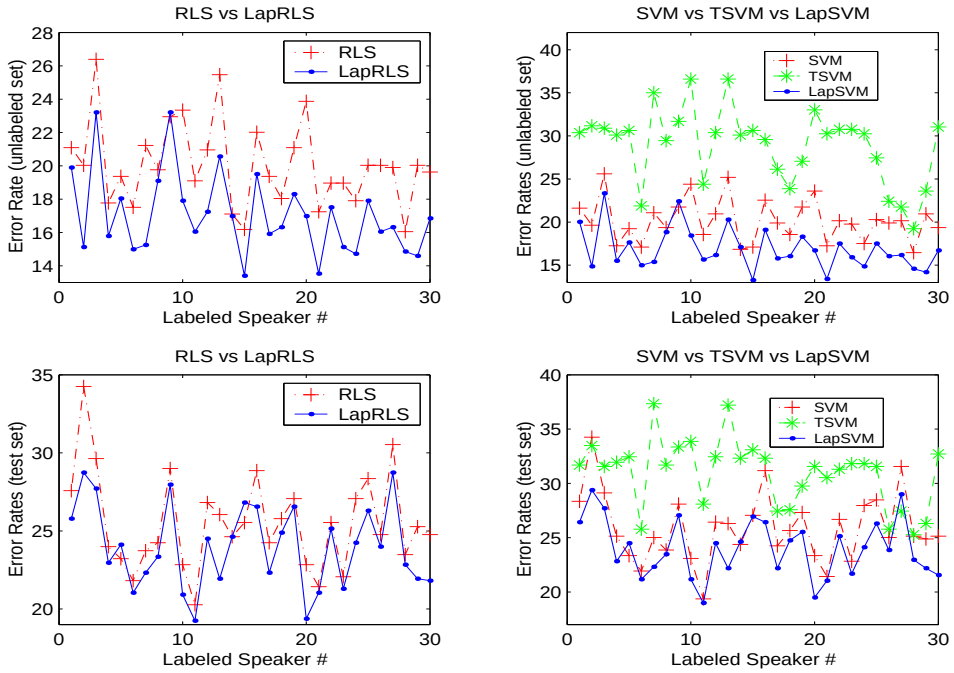


Figure 6: Isolet Experiment - Error Rates at precision-recall break-even points of 30 binary classification problems

performance improvements are smaller but consistent over the choice of the labeled speaker. This can be expected since there appears to be a systematic bias that affects all algorithms, in favor of same-group speakers. To test this hypothesis, we performed another experiment in which the training and test utterances are both drawn from Isolet1. Here, the second utterance of each letter for each of the 30 speakers in Isolet1 was taken away to form the test set containing 780 examples. The training set consisted of the first utterances for each letter. As before, we considered 30 binary classification problems arising when all utterances of one speaker are labeled and other training speakers are left unlabeled. The scatter plots in Figure 7 confirm our hypothesis, and show high correlation between in-sample and out-of-sample performance of our algorithms in this experiment. It is encouraging to note performance improvements with unlabeled data in Experiment 1 where the test data comes from a slightly different distribution. This robustness is often desirable in real-world applications.

In Table 4 we report mean error rates over the 30 splits from one-vs-rest 26-class experiments on this data set. The parameters were held fixed as in the 2-class setting. The failure of TSVM in producing reasonable results on this data set has also been observed in Joachims (2003). With LapSVM and LapRLS we obtain around 3 to 4% improvement over their supervised counterparts.

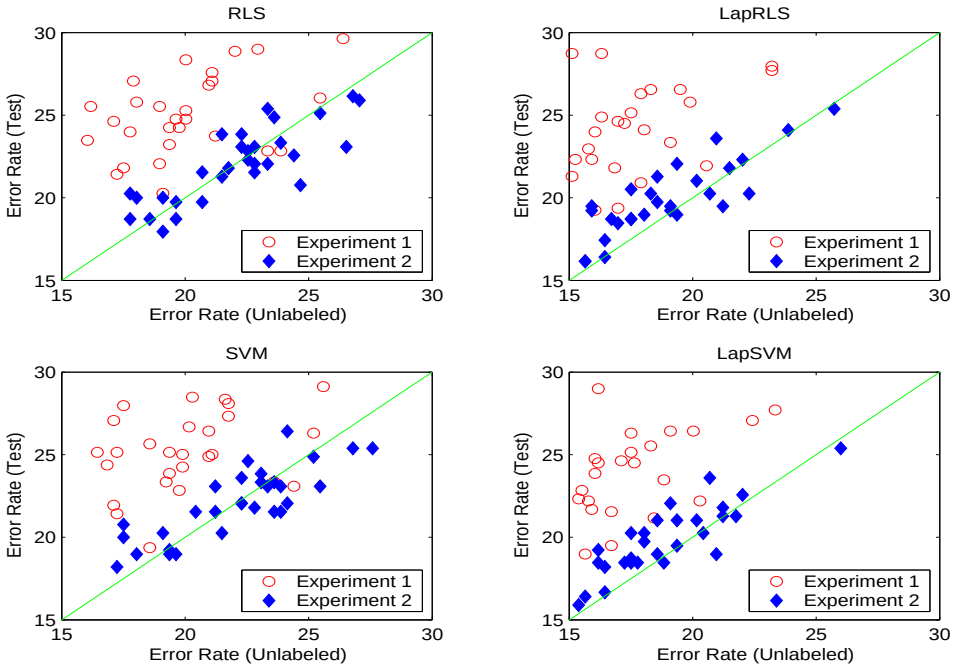


Figure 7: Isolet Experiment - Error Rates at precision-recall break-even points on test set versus unlabeled set. In Experiment 1, the training data comes from Isolet 1 and the test data comes from Isolet5; in Experiment 2, both training and test sets come from Isolet1.

Method	SVM	TSVM	LapSVM	RLS	LapRLS
Error (unlabeled)	28.6	46.6	24.5	28.3	24.1
Error (test)	36.9	43.3	33.7	36.3	33.3

Table 4: Isolet: one-versus-rest multiclass error rates

5.4 Text Categorization

We performed Text Categorization experiments on the WebKB data set which consists of 1051 web pages collected from Computer Science department web-sites of various universities. The task is to classify these web pages into two categories: *course* or *non-course*. We considered learning classifiers using only textual content of the web pages, ignoring link information. A bag-of-word vector space representation for documents is built using the the top 3000 words (skipping HTML headers) having highest mutual information with the class variable, followed by TFIDF mapping.[4] Feature vectors are normalized to unit length. 9 documents were found to contain none of these words and were removed from the data set.

4. TFIDF stands for Term Frequency Inverse Document Frequency. It is a common document preprocessing procedure, which combines the number of occurrences of a given term with the number of documents containing it.

For the first experiment, we ran LapRLS and LapSVM in a transductive setting, with 12 randomly labeled examples (3 course and 9 non-course) and the rest unlabeled. In Table 5, we report the precision and error rates at the precision-recall break-even point averaged over 100 realizations of the data, and include results reported in Joachims (2003) for spectral graph transduction, and the cotraining algorithm (Blum and Mitchell, 1998) for comparison. We used 15 nearest neighbor graphs, weighted by cosine distances and used iterated Laplacians of degree 3. For inductive methods, $\gamma_A l$ was set to 0.01 for RLS and 1.00 for SVM. For LapRLS and LapSVM, γ_A was set as in inductive methods, with $\frac{\gamma_I l}{(l+u)^2} = 100\gamma_A l$. These parameters were chosen based on a simple grid search for best performance over the first 5 realizations of the data. Linear kernels and cosine distances were used since these have found wide-spread applications in text classification problems, for example, in Dumais et al. (1998).

Method	PRBEP	Error
k-NN	73.2	13.3
SGT	86.2	6.2
Naive-Bayes	—	12.9
Cotraining	—	6.20
SVM	76.39 (5.6)	10.41 (2.5)
TSVM	88.15 (1.0)	5.22 (0.5)
LapSVM	87.73 (2.3)	5.41 (1.0)
RLS	73.49 (6.2)	11.68 (2.7)
LapRLS	86.37 (3.1)	5.99 (1.4)

Table 5: Precision and Error Rates at the Precision-Recall Break-even Points of supervised and transductive algorithms.

Since the exact data sets on which these algorithms were run, somewhat differ in preprocessing, preparation and experimental protocol, these results are only meant to suggest that manifold regularization algorithms perform similar to state-of-the-art methods for transductive inference in text classification problems. The following comments can be made: (a) transductive categorization with LapSVM and LapRLS leads to significant improvements over inductive categorization with SVM and RLS. (b) Joachims (2003) reports 91.4% precision-recall break-even point, and 4.6% error rate for TSVM. Results for TSVM reported in the table were obtained when we ran the TSVM implementation using SVM-Light software on this particular data set. The average training time for TSVM was found to be more than 10 times slower than for LapSVM. (c) The cotraining results were obtained on unseen test data sets utilizing additional hyperlink information, which was excluded in our experiments. This additional information is known to improve performance, as demonstrated in Joachims (2003) and Blum and Mitchell (1998).

In the next experiment, we randomly split the WebKB data into a test set of 263 examples and a training set of 779 examples. We noted the performance of inductive and semi-supervised classifiers on unlabeled and test sets as a function of the number of labeled examples in the training set. The performance measure is the precision-recall break-even point (PRBEP), averaged over 100 random

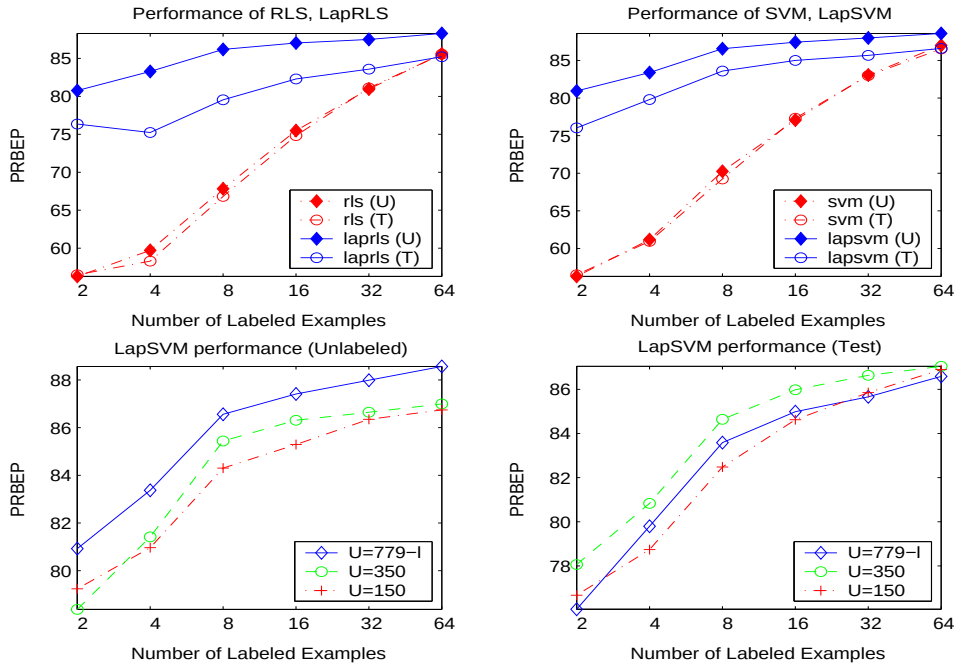


Figure 8: WebKb Text Classification Experiment: The top panel presents performance in terms of precision-recall break-even points (PRBEP) of RLS,SVM,Laplacian RLS and Laplacian SVM as a function of number of labeled examples, on test (marked as T) set and unlabeled set (marked as U and of size 779-number of labeled examples). The bottom panel presents performance curves of Laplacian SVM for different number of unlabeled points.

data splits. Results are presented in the top panel of Figure 8. The benefit of unlabeled data can be seen by comparing the performance curves of inductive and semi-supervised classifiers.

We also performed experiments with different sizes of the training set, keeping a randomly chosen test set of 263 examples. The bottom panel in Figure 8 presents the quality of transduction and semi-supervised learning with Laplacian SVM (Laplacian RLS performed similarly) as a function of the number of labeled examples for different amounts of unlabeled data. We find that transduction improves with increasing unlabeled data. We expect this to be true for test set performance as well, but do not observe this consistently possibly since we use a fixed set of parameters that become suboptimal as unlabeled data is increased. The optimal choice of the regularization parameters depends on the amount of labeled and unlabeled data, and should be adjusted by the model selection protocol accordingly.

6. Unsupervised and Fully Supervised Cases

While the previous discussion concentrated on the semi-supervised case, our framework covers both unsupervised and fully supervised cases as well. We briefly discuss each in turn.

6.1 Unsupervised Learning: Clustering and Data Representation

In the unsupervised case one is given a collection of unlabeled data points $x_1, \ldots, x_u$. Our basic algorithmic framework embodied in the optimization problem in Equation 2 has three terms: (i) fit to labeled data, (ii) extrinsic regularization and (iii) intrinsic regularization. Since no labeled data is available, the first term does not arise anymore. Therefore we are left with the following optimization problem:

$$\min_{f \in \mathcal{H}_K} \gamma_A \|f\|_K^2 + \gamma_I \|f\|_I^2$$

Of course, only the ratio $\gamma = \frac{\gamma_A}{\gamma_I}$ matters. As before $\|f\|_I^2$ can be approximated using the unlabeled data. Choosing $\|f\|_I^2 = \int_{\mathcal{M}} \langle \nabla_{\mathcal{M}} f, \nabla_{\mathcal{M}} f \rangle$ and approximating it by the empirical Laplacian, we are left with the following optimization problem:

$$f^* = \operatorname*{argmin}_{\substack{\sum_i f(x_i)=0;\ \sum_i f(x_i)^2=1 \\ f \in \mathcal{H}_K}} \gamma \|f\|_K^2 + \sum_{i \sim j} (f(x_i) - f(x_j))^2. \tag{13}$$

Note that to avoid degenerate solutions we need to impose some additional conditions (cf. Belkin and Niyogi, 2003a). It turns out that a version of Representer theorem still holds showing that the solution to Equation 13 admits a representation of the form

$$f^* = \sum_{i=1}^{u} \alpha_i K(x_i, \cdot).$$

By substituting back in Equation 13, we come up with the following optimization problem:

$$\alpha = \operatorname*{argmin}_{\substack{\mathbf{1}^T K \alpha = 0 \\ \alpha^T K^2 \alpha = 1}} \gamma \|f\|_K^2 + \sum_{i \sim j} (f(x_i) - f(x_j))^2,$$

where $\mathbf{1}$ is the vector of all ones and $\alpha = (\alpha_1, \ldots, \alpha_u)$ and K is the corresponding Gram matrix.

Letting P be the projection onto the subspace of $\mathbb{R}^u$ orthogonal to $K\mathbf{1}$, one obtains the solution for the constrained quadratic problem, which is given by the generalized eigenvalue problem

$$P(\gamma K + KLK)P\mathbf{v} = \lambda PK^2 P\mathbf{v}. \tag{14}$$

The final solution is given by $\alpha = P\mathbf{v}$, where $\mathbf{v}$ is the eigenvector corresponding to the smallest eigenvalue.

Remark 1: The framework for clustering sketched above provides a method for regularized spectral clustering, where γ controls the smoothness of the resulting function in the ambient space. We also obtain a natural out-of-sample extension for clustering points not in the original data set. Figures 9,10 show results of this method on two two-dimensional clustering problems. Unlike recent work (Bengio et al., 2004; Brand, 2003) on out-of-sample extensions, our method is based on a Representer theorem for RKHS.

Remark 2: By taking multiple eigenvectors of the system in Equation 14 we obtain a natural regularized out-of-sample extension of Laplacian Eigenmaps. This leads to new method for dimensionality reduction and data representation. Further study of this approach is a direction of future research. We note that a similar algorithm has been independently proposed in Vert and Yamanishi (2005) in the context of supervised graph inference. A relevant discussion is also presented in Ham et al. (2005) on the interpretation of several geometric dimensionality reduction techniques as kernel methods.

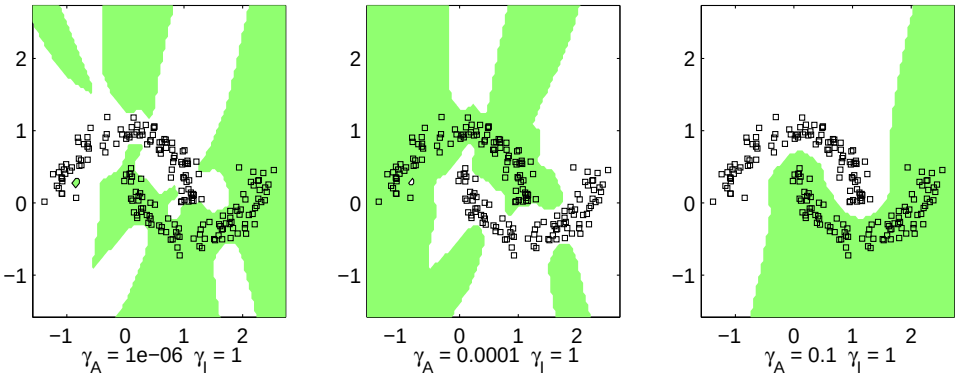


Figure 9: Two Moons data set: Regularized clustering

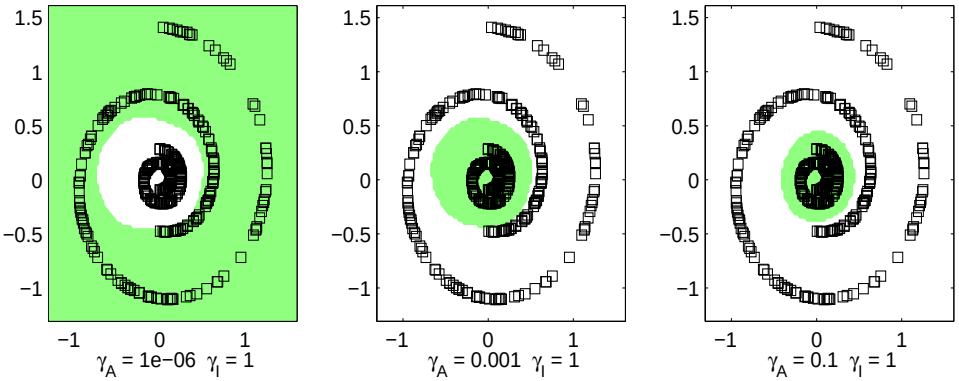


Figure 10: Two Spirals data set: Regularized clustering

6.2 Fully Supervised Learning

The fully supervised case represents the other end of the spectrum of learning. Since standard supervised algorithms (SVM and RLS) are special cases of manifold regularization, our framework is also able to deal with a labeled data set containing no unlabeled examples. Additionally, manifold regularization can augment supervised learning with intrinsic regularization, possibly in a class-dependent manner, which suggests the following algorithm:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \| f \|_K^2 + \frac{\gamma_I^+}{(u+l)^2} \mathbf{f}_+^T L_+ \mathbf{f}_+ + \frac{\gamma_I^-}{(u+l)^2} \mathbf{f}_-^T L_- \mathbf{f}_- .$$

Here we introduce two intrinsic regularization parameters γ_I^+, γ_I^- and regularize separately for the two classes: $\mathbf{f}_+$, $\mathbf{f}_-$ are the vectors of evaluations of the function f, and L_+, L_- are the graph Laplacians, on positive and negative examples respectively. The solution to the above problem for

RLS and SVM can be obtained by replacing $\gamma_I L$ by the block-diagonal matrix $\begin{pmatrix} \gamma_I^+ L_+ & 0 \\ 0 & \gamma_I^- L_- \end{pmatrix}$ in the manifold regularization formulas given in Section 4.

Detailed experimental study of this approach to supervised learning is left for future work.

7. Conclusions and Further Directions

We have a provided a novel framework for data-dependent geometric regularization. It is based on a new Representer theorem that provides a basis for several algorithms for unsupervised, semi-supervised and fully supervised learning. This framework brings together ideas from the theory of regularization in reproducing kernel Hilbert spaces, manifold learning and spectral methods.

There are several directions of future research:

1. Convergence and generalization error: The crucial issue of dependence of generalization error on the number of labeled and unlabeled examples is still very poorly understood. Some very preliminary steps in that direction have been taken in Belkin et al. (2004).

2. Model selection: Model selection involves choosing appropriate values for the extrinsic and intrinsic regularization parameters. We do not as yet have a good understanding of how to choose these parameters. More systematic procedures need to be developed.

3. Efficient algorithms: The naive implementations of our algorithms have cubic complexity in the number of labeled and unlabeled examples, which is restrictive for large scale real-world applications. Scalability issues need to be addressed.

4. Additional structure: In this paper we have shown how to incorporate the geometric structure of the marginal distribution into the regularization framework. We believe that this framework will extend to other structures that may constrain the learning task and bring about effective learnability. One important example of such structure is invariance under certain classes of natural transformations, such as invariance under lighting conditions in vision. Some ideas are presented in Sindhwani (2004).

Acknowledgments

We are grateful to Marc Coram, Steve Smale and Peter Bickel for intellectual support and to NSF funding for financial support. We would like to acknowledge the Toyota Technological Institute for its support for this work. We also thank the anonymous reviewers for helping to improve the paper.

References

R.A. Adams. *Sobolev spaces*. Academic Press New York, 1975.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

M. Belkin. *Problems of Learning on Manifolds*. PhD thesis, The University of Chicago, 2003.

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. *COLT*, 2004.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003a.

M. Belkin and P. Niyogi. Using manifold structure for partially labeled classification. *Advances in Neural Information Processing Systems*, 15:929–936, 2003b.

M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Proc. of COLT*, 2005.

M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

Y. Bengio, J.F. Paiement, P. Vincent, and O. Delalleau. Out-of-sample extensions for LLE, isomap, MDS, Eigenmaps, and spectral clustering. *Advances in Neural Information Processing Systems*, 16, 2004.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 11:368–374, 1999.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. *Proc. 18th International Conf. on Machine Learning*, pages 19–26, 2001.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

O. Bousquet, O. Chapelle, and M. Hein. Measure based regularization. *Advances in Neural Information Processing Systems*, 16, 2004.

M. Brand. Continuous nonlinear dimensionality reduction by kernel eigenmaps. *Int. Joint Conf. Artif. Intel*, 2003.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-supervised Learning*. MIT Press, 2006.

O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems*, 15:585–592, 2003.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005.

F.R.K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

RR Coifman, S. Lafon, AB Lee, M. Maggioni, B. Nadler, F. Warner, and SW Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005.

A. Corduneanu and T. Jaakkola. On information regularization. *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence*, 2003.

F. Cucker and S. Smale. On the mathematical foundations of learning. *American Mathematical Society*, 39(1):1–49, 2002.

O. Delalleau, Y. Bengio, and N. Le Roux. Efficient non-parametric function induction in semi-supervised learning. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

M.P. Do Carmo. *Riemannian Geometry*. Birkhauser, 1992.

D.L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. *Proceedings of the Seventh International Conference on Information and Knowledge Management*, 11:16, 1998.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

G. Fung and O.L. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15(1):99–05, 2001.

A. Grigor'yan. Heat kernels on weighted manifolds and applications. *Cont. Math, 398, 93-191*, 2006.

J. Ham, D.D. Lee, and L.K. Saul. Semisupervised alignment of manifolds. *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence, Z. Ghahramani and R. Cowell, Eds*, 10:120–127, 2005.

M. Hein, J.Y. Audibert, and U. von Luxburg. From graphs to manifolds-weak and strong pointwise consistency of graph Laplacians. *Proceedings of the 18th Conference on Learning Theory (COLT)*, pages 470–485, 2005.

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

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

C. Kemp, T.L. Griffiths, S. Stromsten, and J.B. Tenenbaum. Semi-supervised learning with trees. *Advances in Neural Information Processing Systems*, 16, 2004.

R.I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. *Proc. 19th International Conf. on Machine Learning*, 2002.

S. Lafon. *Diffusion Maps and Geometric Harmonics*. PhD thesis, Yale University, 2004.

K. Nigam, A.K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2):103–134, 2000.

R.M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusets Institute of Technology, 2002.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323, 2000.

B. Scholkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. *Proceedings, First International Conference on Knowledge Discovery & Data Mining, Menlo Park*, 1995.

B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press Cambridge, Mass, 2002.

M. Seeger. Learning with labeled and unlabeled data. *Inst. for Adaptive and Neural Computation, technical report, Univ. of Edinburgh*, 2001.

V. Sindhwani. Kernel machines for semi-supervised learning. Master's thesis, The University of Chicago, 2004.

V. Sindhwani, M. Belkin, and P. Niyogi. The geometric basis of semi-supervised learning. In O. Chapelle, A. Zien, and B. Schölkopf, editors, *Semi-supervised Learning*, chapter 12, pages 217–235. MIT Press, 2006.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings, Twenty Second International Conference on Machine Learning*, 2005.

A. Smola and R. Kondor. Kernels and regularization on graphs. *Conference on Learning Theory, COLT/KW*, 2003.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. *Advances in Neural Information Processing Systems*, 14:945–952, 2002.

J.B. Tenenbaum, V. Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

A.N. Tikhonov. Regularization of incorrectly posed problems. *Sov. Math. Dokl*, 4:1624–1627, 1963.

I. W. Tsang and J. T. Kwok. Very large scale manifold regularization using core vector machines. *NIPS 2005 Workshop on Large Scale Kernel Machines*, 2005.

V.N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

J.P. Vert and Y. Yamanishi. Supervised graph inference. *Advances in Neural Information Processing Systems*, 17:1433–1440, 2005.

U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *Max Planck Institute for Biological Cybernetics Technical Report TR*, 134, 2004.

G. Wahba. *Spline models for observational data*. Society for Industrial and Applied Mathematics Philadelphia, Pa, 1990.

D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. *Advances in Neural Information Processing Systems*, 17, 2005.

Journal of Machine Learning Research 7 (2006) 2435-2447 Submitted 11/05; Revised 6/06; Published 10/06

Consistency of Multiclass Empirical Risk Minimization Methods Based on Convex Loss

Di-Rong Chen DRCHEN@BUAA.EDU.CN
Department of Mathematics, and LMIB
Beijing University of Aeronautics and Astronautics
Beijing 100083, P. R. CHINA

Tao Sun ST.126@163.COM.CN
Department of Mathematics
Beijing University of Aeronautics and Astronautics
Beijing 100083, P. R. CHINA

Editor: Peter Bartlett

Abstract

The consistency of classification algorithm plays a central role in statistical learning theory. A consistent algorithm guarantees us that taking more samples essentially suffices to roughly reconstruct the unknown distribution. We consider the consistency of ERM scheme over classes of combinations of very simple rules (base classifiers) in multiclass classification. Our approach is, under some mild conditions, to establish a quantitative relationship between classification errors and convex risks. In comparison with the related previous work, the feature of our result is that the conditions are mainly expressed in terms of the differences between some values of the convex function.

Keywords: multiclass classification, classifier, consistency, empirical risk minimization, constrained comparison method, Tsybakov noise condition

1. Introduction

We consider the consistency of empirical risk minimization (ERM) algorithm in multiclass classification.

Given an input vector $x \in \mathcal{X} \subseteq \mathbb{R}^d$, we would like to predict its corresponding label $y \in \{1,2,\ldots,K\}$. A classifier f is a function defined on $\mathcal{X}$ with values in $\{1,2,\ldots,K\}$. The quality of this classifier can be measured by the classification error

$$\mathcal{R}(f) = \mathbb{E}_{X,Y} I_{\{f(X)\neq Y\}},$$

where I_A is the characteristic function of set A, and X,Y are drawn from an unknown underlying distribution D. It is clear that $\mathcal{R}(f) = \mathbb{P}\{Y \neq f(X)\}$. If we know the conditional density $\mathbb{P}\{Y = c|X = x\}$, then the classifier ϕ_B given by

$$\phi_B(x) := \arg \max_{c\in\{1,2,\ldots,K\}} \mathbb{P}\{Y = c|X = x\},$$

referred to as Bayes rule, minimizes $\mathcal{R}(f)$ over all classifiers: $\mathcal{R}(\phi_B) = \inf \mathcal{R}(f)$. Henceforth, let $\mathcal{R}^*$ stand for the number $\inf \mathcal{R}(f)$. However, the conditional density is unknown in practice.

Instead, we are given n samples $\{(X_1,Y_1),\ldots,(X_n,Y_n)\}$ of independent random variable drawn from the underlying distribution D. The goal of statistical learning is to find a classifier based on the samples and a pre-chosen set $\mathcal{F}$ of vector functions with K-components. For this purpose, a very successful method used in binary classification is to solve a minimization problem of a risk based on a convex loss ϕ. Main examples of ϕ include the exponential loss $\phi(x)=e^{-x}$ used in AdaBoost, the logit loss $\phi(x)=\ln(1+e^{-x})$ and the hinge loss $\phi(x)=(1-x)_+$ used in support vector machine, where $(u)_+=\max\{0,u\}$ for a number $u\in\mathbb{R}$.

Probably since one can solve a multiclass classification problem $(K>2)$ by solving several binary classification problems, there are much fewer studies on multiclass classification algorithms based directly on minimizing empirical risk with convex loss. Recently, Zhang (2004b) proposes a natural version of EMR scheme in solving a multiclass problem:

$$\hat{\mathbf{f}}=\arg\min_{\mathbf{f}\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^{n}\Psi_{Y_i}(\mathbf{f}(X_i)), \tag{1}$$

where Ψ_c is a mapping from $\mathbb{R}^K$ to $\mathbb{R}$, which is usually constructed by some convex loss function ϕ. In the following, we use bold symbols such as $\mathbf{f}$ and $\mathbf{q}$ to denote vectors, and f_c and q_c to denote their c-th component. We also use $\mathbf{f}(\cdot)$ to denote a vector function. Once obtaining $\hat{\mathbf{f}}$, we have a classifier $C(\hat{\mathbf{f}})$, where $C(\mathbf{f})$ is defined by

$$C(\mathbf{f})(x)=\arg\max_c f_c(x),\qquad \forall\mathbf{f}=(f_1(x),\ldots,f_K(x)).$$

A natural question is how close the optimal Bayes error $\mathcal{R}^*$ can be approximately reached by $\mathcal{R}(C(\hat{\mathbf{f}}))$. A very desirable property is the consistency of algorithm: the excess error $\mathcal{R}(C(\hat{\mathbf{f}}))-\mathcal{R}^*\to 0$ in some sense, as the size n of samples increases to ∞. A consistent algorithm guarantees us that taking more samples essentially suffices to roughly reconstruct the unknown distribution. A good learning algorithm should be consistent.

In recent years, a large part of research has been focused on classifiers which base their decision on a certain combination of (base) classifiers. Suppose that $\mathcal{H}$ is a set of classifiers and λ is a positive number. Let $\mathcal{F}=\mathcal{F}_\lambda$ be the following set of vector functions

$$\mathcal{F}_\lambda=\Big\{\mathbf{f}=\Big(\sum_{j=1}^{J}\beta_j T_c(h_j(\cdot))\Big)_{c=1}^{K}:\beta_j>0,h_j\in\mathcal{H},J=1,2,\ldots,\sum_{j=1}^{J}\beta_j=\lambda\Big\},$$

where $T_c, c=1,\ldots,K$, are functions defined on $\{1,2,\ldots,K\}$ by

$$T_c(h)=\begin{cases}K-1, & \text{if } h=c,\\ -1 & \text{if } h\neq c.\end{cases}$$

A classifier $C(\mathbf{f})$ with $\mathbf{f}\in\mathcal{F}_\lambda$ may be thought as one that, upon observing x, takes a weighted vote of classifiers $h_1,\ldots,h_J$, using weights $\beta_1,\ldots,\beta_J$.

For $K=2$, the vector function $\mathbf{f}=(f_1,f_2)\in\mathcal{F}_\lambda$ satisfies $f_1+f_2=0$. Therefore $\mathcal{F}_\lambda$ is usually regarded as the set of functions $f=\sum_{j=1}^{J}\beta_j T_1(h_j(\cdot)),h_j\in\mathcal{H},\sum_{j=1}^{J}\beta_j=\lambda$. In different versions of boosting, bagging and arcing algorithms, the output classifiers are constructed by weighted voting schemes. Their consistency is established in Lugosi and Vayatis (2004) under the assumption that the Bayes classifier can be approximated by $\mathcal{F}_\lambda$ and $\mathcal{H}$ has a finite VC dimension.

The computational feasibility of schemes (1) has been recognized all along. Moreover, in binary classification, as revealed recently in binary classification problem, a striking feature of ERM (1) using a convex loss is that one can upper bound the excess error by the excess $\{\Psi_c\}_c$-risk $\mathcal{E}(\hat{\mathbf{f}}) - \mathcal{E}^*$, where $\mathcal{E}(\mathbf{f}) = \mathbb{E}_{X,Y}\Psi_Y(\mathbf{f}(X))$ is the expectation of $\Psi_Y(\mathbf{f}(X))$, referred to as the $\{\Psi_c\}$-risk, and $\mathcal{E}^*$ is the infimum $\inf_{\mathbf{f}} \mathcal{E}(\mathbf{f})$ of $\mathcal{E}(\mathbf{f})$ over an appropriate set (not restricted to $\mathcal{F}$). Consequently, we have a very important implication relation (e.g., Bartlett et al., 2005; Lugosi and Vayatis, 2004; Chen et al., 2004; Zhang, 2004a)

$$\mathcal{E}(\hat{\mathbf{f}}) \to \mathcal{E}^* \Rightarrow \mathcal{R}(C(\hat{\mathbf{f}})) \to \mathcal{R}^*.$$

The notion of classification calibrated in Bartlett et al. (2005) is extended to multiclass classification problem and is used to characterize above implication in Tewari and Bartlett (2005). Such an implication is also established under the so called infinite-sample-consistency (ISC) condition on $\{\Psi_c\}_c$ (see Zhang, 2004b). Moreover, an quantitative relation between the excess error and the excess $\{\Psi_c\}$-risk is obtained for One-versus-All method in Zhang (2004b).

In this paper we consider the constrained comparison method in multiclass classification problem. One of our goals is to generalize the results of consistency for weighted voting schemes in Lugosi and Vayatis (2004) to multiclass case. We first establish an inequality concerning with the excess error and the excess $\{\Psi_c\}$-risk. The inequality is interesting in its own right.

The paper is organized as following. In Section 2, we upper bound the excess error by the excess $\{\Psi_c\}$-risk under some mild conditions. In comparison with the previous work, our conditions are mainly expressed in terms of the differences between some values of function ϕ. On the other hand, the sufficient conditions ensuring the quantitative relationships, even in case $K = 2$, are expressed previously in terms of the infimum $\inf_{\mathbf{f}} \mathbb{E}(\Psi_Y(\mathbf{f}(X))|X = x)$. In Section 3, we apply the results in Section 2 to establish a consistency result in multiclass case, similar to that of Lugosi and Vayatis (2004).

2. Bounding Classification Error by Convexity

In this section, we upper bound, under some conditions on convex loss ϕ, the excess classification error $\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^*$ by the excess $\{\Psi_c\}$-risk $\mathcal{E}(\mathbf{f}) - \mathcal{E}^*$ for the constrained comparison method. Similar result is established for the One-versus-All method (see Zhang, 2004b). The two methods are different: in the One-versus-All method, one can deal with each component of the vector function separately. The conditions and proofs here are different from those in Zhang (2004b). Moreover, a tighter upper bound is given under Tsybakov noise condition.

Recall that $\mathbb{P}\{Y = c|X = x\}$ is the conditional probability. Let

$$\mathbf{q}(x) = (q_c(x))_{c=1}^K, \qquad q_c(x) = \mathbb{P}\{Y = c|X = x\}.$$

Suppose that ϕ is a convex function on $\mathbb{R}$. The constrained comparison method proposed in Zhang (2004b) uses Ψ_c below.

$$\Psi_c(\mathbf{f}) = \sum_{k=1,k\neq c}^{K} \phi(-f_k), \qquad \mathbf{f} \in \Omega := \Big\{\mathbf{f} \in \mathbb{R}^K : \sum_c f_c = 0\Big\}.$$

Then the risk $\mathcal{E}(\mathbf{f})$ may by expressed as

$$\mathcal{E}(\mathbf{f}) = \mathbb{E}_X W(\mathbf{q}(X), \mathbf{f}(X)), \tag{2}$$

with $W(\mathbf{q},\mathbf{f}) = \sum_{c=1}^{K}(1-q_c)\phi(-f_c)$.

Note that we use q_c to denote the c-th component of a K-dimensional vector $\mathbf{q} \in \Lambda_K$, where Λ_K is the set of possible conditional probability vectors:

$$\Lambda_K = \Big\{ q \in \mathbb{R}^K : \sum_{c=1}^{K} q_c = 1, q_c \geq 0 \Big\}.$$

Denote by $\mathcal{B}$ the set of all K-dimensional vectors of Borel measurable functions on $\mathcal{X}$ and $\mathcal{B}_\Omega = \{\mathbf{f} \in \mathcal{B} : \forall x \in \mathcal{X}, \mathbf{f}(x) \in \Omega\}$. Let $\mathcal{E}^* = \inf_{\mathbf{f}\in\mathcal{B}_\Omega} \mathcal{E}(\mathbf{f})$.

For any $\mathbf{q} \in \Lambda_K$, let $W^*(\mathbf{q}) := \inf_{\mathbf{f}\in\Omega} W(\mathbf{q},\mathbf{f})$. It is easily seen that

$$\mathcal{E}^* = \mathbb{E}W^*(\mathbf{q}(X)).$$

Lemma 2.1 *Assume that ϕ is a decreasing and convex function on $\mathbb{R}$. Let $W(\mathbf{q},\mathbf{f})$ be given as above. Suppose that $\mathbf{q} \in \Lambda_K$ and $\mathbf{f} \in \mathcal{B}_\Omega$ satisfy that there are i, j such that $q_i < q_j$ and $f_j < f_i$. Then $W(\mathbf{q},\mathbf{f}') \leq W(\mathbf{q},\mathbf{f})$, where $\mathbf{f}' = (f_1', \cdots, f_K')$ is given by $f_i' = f_j' = \frac{f_i+f_j}{2}$, and $f_c' = f_c$, $c \neq i, j$.*

Proof. Without loss of generally, we can assume that $q_1 < q_2$ and $f_2 < f_1$. Then

$$\frac{f_1+f_2}{2} \leq \frac{(1-q_1)f_1 + (1-q_2)f_2}{2-q_1-q_2}.$$

By assumption, we have

$$\begin{aligned}(2-q_1-q_2)\phi\Big(-\tfrac{f_1+f_2}{2}\Big) &\leq (2-q_1-q_2)\phi\Big(-\tfrac{(1-q_1)f_1+(1-q_2)f_2}{2-q_1-q_2}\Big)\\ &\leq (1-q_1)\phi(-f_1) + (1-q_2)\phi(-f_2).\end{aligned}$$

Therefore the proof is complete by

$$\begin{aligned} & W(\mathbf{q},\mathbf{f}) - W(\mathbf{q},\mathbf{f}')\\ = \;& (1-q_1)\phi(-f_1) + (1-q_2)\phi(-f_2) - (2-q_1-q_2)\phi(-\tfrac{f_1+f_2}{2}) \geq 0.\end{aligned}$$

■

Lemma 2.2 *Assume that ϕ is a decreasing and convex function on $\mathbb{R}$. Suppose that there exist positive constants $k > 0$ and $\alpha \geq 1$ such that for any $\mathbf{q} \in \Lambda_K$,*

$$k(q_j - q_i)^\alpha \leq W^*(\mathbf{q}') - W^*(\mathbf{q}), \tag{3}$$

where $j = \arg\max_c q_c$ and $q_i < q_j$, and $\mathbf{q}'$ is given by $\mathbf{q}' = (q_1', \cdots, q_K')$, where $q_i' = q_j' = \frac{q_i+q_j}{2}$, and $q_c' = q_c, c \neq i, j$. Then for any $\mathbf{f} \in \mathcal{B}_\Omega$,

$$k(\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^*) \leq \mathcal{E}(\mathbf{f}) - \mathcal{E}^*)^{\frac{1}{\alpha}}.$$

Proof. Recall that $q_c(x)$ is the conditional probability $\mathbb{P}\{Y = c|X = x\}$. For any $\mathbf{f}$ we have by definition of $\mathcal{R}(C(\mathbf{f}))$

$$\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^* = \int_{\mathcal{X}} \Big(q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x) \Big) d\rho_X. \tag{4}$$

Let $\mathbf{q}(x) = (q_c(x))_{c=1}^K$. Also by (2)

$$\mathcal{E}(\mathbf{f}) - \mathcal{E}^* = \int_{\mathcal{X}} \Big(W(\mathbf{q}(\mathbf{x}), \mathbf{f}(\mathbf{x})) - W^*(\mathbf{q}(x)) \Big) d\rho_X. \tag{5}$$

Let $x \in \mathcal{X}$ be given such that $q_{C(\mathbf{f})(x)}(x) \neq q_{\phi_B(x)}(x)$. Denote $j = \phi_B(x)$ and $i = C(\mathbf{f})(x)$. We regard $\mathbf{q}(x)$, $\mathbf{f}(x)$ and $\mathbf{f}'(x)$ as $\mathbf{q}$, $\mathbf{f}$ and $\mathbf{f}'$ in Lemma 2.1 respectively.

By assumption, we have $W(\mathbf{q}(x), \mathbf{f}'(x)) = W(\mathbf{q}'(x), \mathbf{f}'(x)) \geq W^*(\mathbf{q}'(x))$. It follows from Lemma 2.1 that $W^*(\mathbf{q}'(x)) \leq W(\mathbf{q}(x), \mathbf{f}(x))$. Therefore by (3)

$$k(q_j(x) - q_i(x))^\alpha \leq W(\mathbf{q}(x), \mathbf{f}(x)) - W^*(\mathbf{q}(x)).$$

Integrating the above inequality over the set $\mathcal{X}' = \{x \in \mathcal{X} : C(\mathbf{f})(x) \neq \phi_B(x)\}$, we have

$$k \int_{\mathcal{X}'} \Big(q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x) \Big)^\alpha d\rho_X \leq \int_{\mathcal{X}'} \Big(W(\mathbf{q}(\mathbf{x}), \mathbf{f}(\mathbf{x})) - W^*(\mathbf{q}(x)) \Big) d\rho_X.$$

By Hölder inequality, for $\alpha \geq 1$

$$\Big(\int_{\mathcal{X}'} \Big(q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x) \Big) d\rho_X \Big)^\alpha \leq \int_{\mathcal{X}'} \Big(q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x) \Big)^\alpha d\rho_X.$$

Then we have the desired inequality by the definition of $\mathcal{X}'$, (4) and (5). The proof is complete. ■

In the following we impose some conditions on ϕ.

Assumption 2.3 *1. ϕ is a differentiable, convex and decreasing function on $\mathbb{R}$ such that $\lim_{x \to +\infty} \phi(x) = 0$ and $\lim_{x \to -\infty} \phi(x) = +\infty$.*

2. For any $\mathbf{q} = (q_c)_{c=1}^K \in \Lambda_K$ with all $q_c < 1, c \in \{1, \ldots, K\}$, there is a minimizer $\mathbf{f}^* = (f_c^*)_{c=1}^K$ of $W(\mathbf{q}, \mathbf{f})$. Moreover, ϕ is twice differentiable at points $-f_c^*, c = 1, \ldots, K$, and $\phi''(-f_c^*) > 0, c \in \{1, \ldots, K\}$.

For any $\mathbf{q} = (q_c)_{c=1}^K$, let $j = \arg\max_c q_c$ and $i \in \{1, \ldots, K\}$ with $q_i < q_j$. We introduce $\mathbf{q}^t = (q_c^t)_{c=1}^K \in \Lambda_K$ for $0 \leq t \leq \frac{q_j - q_i}{2}$ as following.

$$q_i^t = q_i + t,\ q_j^t = q_j - t, \text{ and } q_c^t = q_c,\ c \neq i, j.$$

Clearly, $q_c^t < 1$ for $0 < t < \frac{q_j - q_i}{2}$ and any $1 \leq c \leq K$. Therefore, for any t, there is a $\mathbf{f}^{t,*} = (f_c^{t,*})_{c=1}^K$ minimizing $W(\mathbf{q}^t, \mathbf{f})$, that is, $W^*(\mathbf{q}^t) = W(\mathbf{q}^t, \mathbf{f}^{t,*})$.

Under a condition weaker than Assumption 2.3, Zhang (2004b) proves that the excess error $\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^*$ is small whenever the excess $\{\Psi\}$-risk $\mathcal{E}(\mathbf{f}) - \mathcal{E}^*$ is small. Our goal however is, under Assumption 2.3, to establish an inequality between the above two quantities. We give a sufficient condition for (3) in terms of the differences between any pair of $\phi(-f_c^{t,*}), c \in \{1, \ldots, K\}$.

Theorem 2.4 *Assume that ϕ satisfies Assumption 2.3. Suppose that there exist positive constants $k_1 > 0$ and $\beta \geq 0$ such that for any $\mathbf{q} \in \Lambda_K$,*

$$k_1(q_j - q_i - 2t)^\beta \leq \phi(-f_j^{t,*}) - \phi(-f_i^{t,*}), \qquad 0 < t < \frac{q_j - q_i}{2}, \tag{6}$$

whenever $j = \arg\max_c q_c$, $q_i < q_j$ *and* $\mathbf{f}^{t,*} = (f_c^{t,*})_{c=1}^K$ *is a minimizer of* $W(\mathbf{q}^t, \mathbf{f})$. *Then for any vector* $\mathbf{f} \in \mathcal{B}_\Omega$,

$$\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^* \leq \frac{2(\beta+1)}{k_1} (\mathcal{E}(\mathbf{f}) - \mathcal{E}^*)^{\frac{1}{\beta+1}}.$$

Proof. We establish condition (3) with $\alpha = \beta + 1$ and $k = \frac{k_1}{2(\beta+1)}$. As above, let $\mathbf{f}^{t,*} = (f_c^{t,*})_{c=1}^K$ be the minimizer of $W(\mathbf{q}^t, \mathbf{f})$. The first-order optimality condition is the set of equations

$$(1 - q_c^t)\phi'(-f_c^{t,*}) = \mu, \qquad c = 1, \ldots, K,$$

where μ, independent of c, is the Lagrangian multiplier. Assumption 2.3 implies that $f_{c,t}$ is differentiable with respect to $t, c = 1, \ldots, K$. Moreover, the constraint $\sum_{c=1}^K f_c^{t,*} = 0 \, (\forall t \in (0, \frac{q_2 - q_1}{2}))$ yields $\sum_{c=1}^K \frac{d f_c^{t,*}}{dt} = 0$. Consequently,

$$\begin{aligned} & \frac{dW^*(\mathbf{q}^t)}{dt} \\ = \; & \phi(-f_j^{t,*}) - \phi(-f_i^{t,*}) - \sum_{c=1}^K (1 - q_c^t)\phi'(-f_c^{t,*}) \frac{d f_c^{t,*}}{dt} \\ = \; & \phi(-f_j^{t,*}) - \phi(-f_i^{t,*}). \end{aligned}$$

Therefore, we have by (6)

$$\frac{dW^*(\mathbf{q}^t)}{dt} \geq k_1 (q_j - q_i - 2t)^\beta, \qquad 0 < t < \frac{q_j - q_i}{2}.$$

Integrating the above inequality over $[0, \frac{q_j - q_i}{2}]$ gives (3) with $\alpha = \beta + 1$ and $k = \frac{k_1}{2(\beta+1)}$. Our conclusion follows from Lemma 2.2. The proof is complete. ■

We consider the exponential loss as the first example.

Example 2.5 *Let* $\phi(x) = e^{-x}$. *Then for any vector* $\mathbf{f} \in \mathcal{B}_\Omega$, *we have*

$$\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^* \leq \frac{4\sqrt[K]{K-1}}{\sqrt[K]{(\frac{2K}{2K-1})^{2K-2}}} \sqrt{\mathcal{E}(\mathbf{f}) - \mathcal{E}^*}.$$

Proof. For $\mathbf{q} = (q_c)_{c=1}^K \in \Lambda_K$ with all $q_c < 1$, the unique minimizer $\mathbf{f}^* = (f_c^*)_{c=1}^K$ is determined by $(1 - q_c)\exp(f_c^*) = \mu, c = 1, \cdots, K$, with μ the Lagrangian multiplier. Assumption 2.3 holds for ϕ. By $\sum_c f_c^* = 0$ we have $\mu = \sqrt[K]{\prod_{c=1}^K (1 - q_c)}$. Therefore

$$\phi(-f_k^*) = \frac{\sqrt[K]{\prod_{c=1}^K (1 - q_c)}}{1 - q_k}, \qquad k = 1, \ldots, K.$$

Let $j = \arg\max_c q_c$ and $i \in \{1, \ldots, K\}$ such that $q_i < q_j$. Recall that $\mathbf{q}^t$ and $\mathbf{f}^{t,*}$ be defined as before. We apply the above equality and obtain

$$\phi(-f_j^{t,*}) - \phi(-f_i^{t,*}) = \frac{\sqrt[K]{\prod_{c \neq i,j} (1 - q_c)}}{((1 - q_j + t)(1 - q_i - t))^{\frac{K-1}{K}}} (q_j - q_i - 2t).$$

If $K=2$, $\prod_{c\neq i,j}(1-q_c)$ is understood as 1. If $K>2$, the $K-2$ nonnegative numbers $q_c, c\neq i,j$, may be arranged in the decreasing order, so that it is easily seen that they are not larger than $\frac{1}{2},\ldots,\frac{1}{K-1}$ respectively. Therefore

$$\prod_{c\neq i,j}(1-q_c)\geq\prod_{c=2}^{K-1}(1-\frac{1}{c})=\frac{1}{K-1}.$$

On the other hand, $(1-q_j+t)(1-q_i-t)\leq(1-\frac{q_i+q_j}{2})^2$ for $0\leq t\leq\frac{q_i-q_j}{2}$. Note $\frac{q_i+q_j}{2}\geq\frac{q_j}{2}\geq\frac{1}{2K}$. Consequently,

$$\phi(-f_j^{t,*})-\phi(-f_i^{t,*})\geq\frac{\sqrt[K]{(\frac{2K}{2K-1})^{2K-2}}}{\sqrt[K]{K-1}}(q_j-q_i-2t).$$

This is (6) with $\beta=1$ and $k_1=\frac{\sqrt[K]{(\frac{2K}{2K-1})^{2K-2}}}{\sqrt[K]{K-1}}$. The conclusion follows from Theorem 2.4. ■

Let $p\geq 1$ and $\phi(x)=(\frac{1}{K-1}-x)_+^p$, where $(x)_+=\max\{x,0\}$. The resulting risk is just the one used in p-norm Support vector machine (SVM). Chen and Xiang (2004) have established the inequality for $p=1$

$$\frac{\mathcal{R}(C(\mathbf{f}))-\mathcal{R}^*}{K-1}\leq\mathcal{E}(\mathbf{f})-\mathcal{E}^*.$$

Example 2.6 *Let $\phi(x)=(\frac{1}{K-1}-x)_+^2$. Then for any vector $\mathbf{f}\in\mathcal{B}_\Omega$, we have*

$$\mathcal{R}(C(\mathbf{f}))-\mathcal{R}^*\leq\frac{4(\frac{K-1}{K})^2}{k_2}\sqrt{\mathcal{E}(\mathbf{f})-\mathcal{E}^*},$$

where $k_2=2(\frac{2K-1}{2K})^2+(\frac{2K-1}{2K})^4\left(\left(\frac{1}{2}\right)^2+\cdots+\left(\frac{K-2}{K-1}\right)^2\right)$ for $K>2$ and $k_2=\frac{1}{8}$ for $K=2$.

Proof. For $\mathbf{q}=(q_c)_{c=1}^k$ with all $q_c<1$, by the method of Lagrange multiplier we conclude that the minimizer $\mathbf{f}^*=(f_c^*)_{c=1}^K$ satisfies $-f_c^*<\frac{1}{K-1}, c\in\{1,\ldots,K\}$. Thus, Assumption 2.3 is satisfied by ϕ. Moreover, we have

$$\phi(-f_k^*)=\left(\frac{K}{K-1}\right)^2\frac{1}{(1-q_k)^2\sum_{c=1}^K\frac{1}{(1-q_c)^2}},\qquad k=1,\ldots,K.$$

Let $j=\arg\max_c q_c$ and $i\in\{1,\ldots,K\}$ such that $q_i<q_j$. Moreover, $\mathbf{q}^t$ and $\mathbf{f}^{t,*}$ are defined as before. An application of the above equality to $\mathbf{q}^t$ and $\mathbf{f}^{t,*}$ yields

$$\begin{aligned}&\phi(-f_j^{t,*})-\phi(-f_i^{t,*})\\ =\;&\left(\frac{K}{K-1}\right)^2\frac{(q_j-q_i-2t)(2-q_i-q_j)}{(1-q_i-t)^2(1-q_j+t)^2\left(\frac{1}{(1-q_i+t)^2}+\frac{1}{(1-q_j+t)^2}+\sum_{c\neq i,j}\frac{1}{(1-q_c)^2}\right)},\end{aligned}$$

where $\sum_{c\neq i,j}\frac{1}{(1-q_c)^2}$ is understood as 0 for $K=2$. It is easily seen that

$$\begin{aligned}&(1-q_i-t)^2(1-q_j+t)^2\left(\frac{1}{(1-q_i+t)^2}+\frac{1}{(1-q_j+t)^2}\right)\\ \leq\;&2(1-\frac{q_j+q_i}{2})^2\leq 2(\frac{2K-1}{2K})^2,\qquad\qquad\forall t\in[0,\frac{q_j-q_i}{2}],\end{aligned}$$

where the second inequality holds by $1/K \le q_j$.

As in Example 2.5, again we arrange $q_c, c \neq i, j$, in decreasing order so that they are not larger than $\frac{1}{2}, \ldots, \frac{1}{K-1}$ respectively. It follows that, for $0 \le t \le \frac{q_j - q_i}{2}$,

$$(1-q_i-t)^2(1-q_j+t)^2 \sum_{c\neq i,j} \frac{1}{(1-q_c)^2} \le \Big(\frac{2K-1}{2K}\Big)^4 \Big(\Big(\frac{1}{2}\Big)^2 + \cdots + \Big(\frac{K-2}{K-1}\Big)^2\Big).$$

Therefore, the condition (6) holds with $\beta = 1$ and $k_1 = \Big(\frac{K}{K-1}\Big)^2 k_2$. The conclusion follows from Theorem 2.4. ■

Remark 2.7 *For $\phi(x) = (\frac{1}{K-1} - x)_+^p$ with $p > 1$, we can also apply Theorem 2.4 and get an inequality $\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^* \le k'\sqrt{\mathcal{E}(\mathbf{f}) - \mathcal{E}^*}$, where k' is a constant. The argument is similar to that of Example 2.8. We point out that $-f_c^* < \frac{1}{K-1}$ for any $\mathbf{q} = (q_c)_{c=1}^K$ with all $q_c < 1, c = 1, \ldots, K$, which ensures that ϕ satisfies Assumption 2.3. Moreover, by simple computation,*

$$\phi(-f_k^*) = \Big(\frac{K}{K-1}\Big)^{\frac{p}{p-1}} \frac{1}{\sum_{c=1}^K \Big(\frac{1-q_k}{1-q_c}\Big)^{\frac{p}{p-1}}}, \qquad k = 1, \ldots, K.$$

The bounds in Lemma 2.2 and Theorem 2.4 may be improved under the so-called Tsybakov noise condition. For any $x \in \mathcal{X}$, let

$$m(x) = q_{\phi_B(x)}(x) - \max\{q_i(x) : q_i(x) < q_{\phi_B(x)}(x), i = 1 \ldots, K\}$$

if the set $\{q_i(x) : q_i(x) < q_{\phi_B(x)}(x), i = 1 \ldots, K\}$ is not empty, and $m(x) = 0$ otherwise.

Definition 2.8 *Let $s \in [0,1]$. We say that $\mathbb{P}$ satisfies Tsybakov noise condition with exponent s, if there is a constant c such that*

$$\mathbb{P}\{X \in \mathcal{X} : 0 < m(X) < t\} \le c t^{\frac{s}{1-s}}, \qquad 0 < t \le 1.$$

As in binary classification (see Bartlett and Mendelson, 2002), Tsybakov noise condition with exponent s implies that there is a constant c such that, for any $\mathbf{f} \in \mathcal{B}_\Omega$,

$$\mathbb{P}\{x : x \in \mathcal{X}, q_{\phi_B(x)}(x) \neq q_{C(\mathbf{f})(x)}(x)\} \le c(\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^*)^s. \tag{7}$$

In fact, Tsybakov noise condition and (4) tell us

$$\begin{aligned} & \mathcal{R}(C(\mathbf{f})) - \mathcal{R}^* \\ \ge & \int_{\mathcal{X}} \Big(q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x)\Big) I_{\{t \le m(x)\}} d\rho_X \\ \ge & \; t\Big(\mathbb{P}\{x : x \in \mathcal{X}, q_{\phi_B(x)}(x) \neq q_{C(\mathbf{f})(x)}(x)\} - c t^{\frac{s}{1-s}}\Big). \end{aligned}$$

Minimizing the last term over t establishes (7).

Theorem 2.9 *Suppose that $\mathbb{P}$ satisfies Tsybakov noise condition with exponent s. If the conditions of Lemma 2.2 are satisfied, then for any vector $\mathbf{f} \in \mathcal{B}_\Omega$ we have*

$$\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^* \le k_\phi (\mathcal{E}(\mathbf{f}) - \mathcal{E}^*)^{\frac{1}{\alpha-(\alpha-1)s}}, \tag{8}$$

where k_ϕ is a constant.

Consequently, under Tsybakov noise condition with exponent s and conditions of Theorem 2.4, we have for any vector $\mathbf{f} \in \mathcal{B}_\Omega$

$$\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^* \le k_\phi (\mathcal{E}(\mathbf{f}) - \mathcal{E}^*)^{\frac{1}{\beta+1-\beta s}}.$$

Proof. For $\mathbf{f} \in \mathcal{B}_\Omega$ and $t \in (0,1]$ set $\mathcal{X}_1 = \{x : x \in \mathcal{X}, 0 < q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x) < t\}$ and $\mathcal{X}_2 = \{x : x \in \mathcal{X}, t \le q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x)\}$. Clearly, $\mathcal{X}_1 \subseteq \{x : x \in \mathcal{X}, q_{\phi_B(x)}(x) \ne q_{C(\mathbf{f})(x)}(x)\}$, which implies $\mathbb{P}(\mathcal{X}_1) \le c(\mathcal{R}(C(\mathbf{f})(x)) - \mathcal{R}^*)^s$ by (7). On the other hand,

$$\begin{aligned} & \int_{\mathcal{X}_2} \Big(q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x) \Big) d\rho_X \\ \le \;& t^{-\alpha+1} \int_{\mathcal{X}} \Big(q_{\phi_B(x)}(x) - q_{C(\mathbf{f})(x)}(x) \Big)^\alpha d\rho_X \\ \le \;& \frac{1}{kt^{\alpha-1}} (\mathcal{E}(\mathbf{f}) - \mathcal{E}^*), \end{aligned}$$

where the last inequality follows from the proof of Lemma 2.2. Therefore we have by (4) that

$$\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^* \le tc(\mathcal{R}(C(\mathbf{f})) - \mathcal{R}^*)^s + \frac{1}{kt^{\alpha-1}} (\mathcal{E}(\mathbf{f}) - \mathcal{E}^*).$$

Minimizing the right hand side of above inequality over $t \in (0,1]$ yields the inequality (8) for some constant c_ϕ.

As a consequence, the second conclusion follows from Theorem 2.4 and (8) with $\alpha = \beta + 1$. The proof is complete. ■

3. Consistency of Weighted Voting Schemes

In this section, we consider the consistency of weight voting schemes by the results of section 2.

Recall that $\mathcal{B}_\Omega$ is given in Section 2. It is easily seen that, for any set $\mathcal{H}$ of classifiers, $\mathcal{F}_\lambda \subset \mathcal{B}_\Omega$.

Assumption 3.1 *Recall that $\mathcal{E}^*$ is defined in Section 2. Suppose that the set $\mathcal{H}$ of classifiers satisfies*

$$\lim_{\lambda \to \infty} \inf_{\mathbf{f} \in \mathcal{F}_\lambda} \mathcal{E}(\mathbf{f}) = \mathcal{E}^*.$$

The notion of VC dimension plays an important role in classification (see Devroye et al., 1996; Vapnik, 1998). Recall that for a collection $\mathcal{A}$ of some sets A, the VC dimension $V_\mathcal{A}$ of $\mathcal{A}$ is defined to be the largest number d, when exists, such that $\mathcal{A}$ shatters a set of some d points (see Devroye et al., 1996). If there exists no such an integer d we define $V_\mathcal{A} = \infty$.

With n samples $\{(X_i, Y_i)\}_{i=1}^n \subset Z^n$, the empirical $\{\Psi_c\}$-risk $\mathcal{E}_n(\mathbf{f})$ of a vector function $\mathbf{f}$ is defined by

$$\mathcal{E}_n(\mathbf{f}) = \frac{1}{n} \sum_{i=1}^n \Psi_{Y_i}(\mathbf{f}(X_i)).$$

Clearly, $\mathcal{E}(\mathbf{f}) = \mathbb{E}_{Z^n} \mathcal{E}_n(\mathbf{f})$.

Lemma 3.2 *Suppose that ϕ satisfies the condition 1 of Assumption 2.3. Moreover, suppose that, for any $c \in \{1,\ldots,K\}$, the collection $\mathcal{A}_c$ of all sets*

$$\{(x,c) : h(x) \neq c\}, \qquad h \in \mathcal{H},$$

has a finite VC dimension $V_{\mathcal{A}_c}$. Then for any n and $\lambda > 0$ we have

$$\mathbb{E} \sup_{\mathbf{f} \in \mathcal{F}_\lambda} |\mathcal{E}(\mathbf{f}) - \mathcal{E}_n(\mathbf{f})| \leq 4K^2\lambda |\phi'(-\lambda(K-1))| \sqrt{\frac{2V \ln(4n+2)}{n}}, \tag{9}$$

where $V = \max_{1 \leq c \leq K} V_{\mathcal{A}_c}$. Also, for any $\delta > 0$, with probability at least $1-\delta$,

$$\begin{aligned} & \sup_{\mathbf{f} \in \mathcal{F}_\lambda} |\mathcal{E}(\mathbf{f}) - \mathcal{E}_n(\mathbf{f})| \\ \leq \;& 4K^2\lambda |\phi'(-\lambda(K-1))| \sqrt{\frac{2V \ln(4n+2)}{n}} + 2\exp\Big(\frac{-n\delta^2}{2(K-1)^2\phi^2(-\lambda K)}\Big). \end{aligned} \tag{10}$$

Proof. The proof is similar to that of Lugosi and Vayatis (2004) Lemma 2. Let $\sigma_1,\ldots,\sigma_n$ be the independent symmetric sign variables, that is,

$$\mathbb{P}\{\sigma_i = -1\} = \mathbb{P}\{\sigma_i = 1\} = \frac{1}{2}.$$

Then, by a standard symmetrization argument,

$$\mathbb{E} \sup_{\mathbf{f} \in \mathcal{F}_\lambda} |\mathcal{E}(\mathbf{f}) - \mathcal{E}_n(\mathbf{f})| \leq 2\mathbb{E} \sup_{\mathbf{f} \in \mathcal{F}_\lambda} \Big| \frac{1}{n} \sum_{i=1}^n \sigma_i (\Psi_{Y_i}(\mathbf{f}(X_i)) - (K-1)\phi(0)) \Big|.$$

On the other hand, it is easily seen that

$$\begin{aligned} & \sup_{\mathbf{f} \in \mathcal{F}_\lambda} \Big| \frac{1}{n} \sum_{i=1}^n \sigma_i (\Psi_{Y_i}(\mathbf{f}(X_i)) - (K-1)\phi(0)) \Big| \\ = \;& \sup_{\mathbf{f} \in \mathcal{F}_1} \Big| \frac{1}{n} \sum_{i=1}^n \sigma_i \sum_{c=1, c \neq Y_i}^K (\phi(-f_c(X_i)) - \phi(0)) \Big| \\ \leq \;& \sum_{c=1}^K \sup_{\mathbf{f} \in \mathcal{F}_1} \Big| \frac{1}{n} \sum_{i=1}^n \sigma_i (\phi(-\lambda f_c(X_i)) - \phi(0)) \Big|, \end{aligned}$$

where the equality holds by the definition of $\mathcal{F}_\lambda$.

For any $c \in \{1,\ldots,K\}$, let $g(t) = \phi(-\lambda t) - \phi(0), t \in [-1, K-1]$. Then $g(0) = 0$, and g satisfies Lipschitz condition with Lipschitz constant $L = -\lambda\phi'(-\lambda(K-1))$. We appeal to the "contraction principle" to conclude for any $c \in \{1,\ldots,K\}$

$$\mathbb{E} \sup_{\mathbf{f} \in \mathcal{F}_1} \Big| \frac{1}{n} \sum_{i=1}^n \sigma_i (\phi(-\lambda f_c(X_i)) - \phi(0)) \Big| \leq 2L\mathbb{E} \sup_{\mathbf{f} \in \mathcal{F}_1} \Big| \frac{1}{n} \sum_{i=1}^n \sigma_i f_c(X_i) \Big|,$$

and consequently,

$$\mathbb{E} \sup_{\mathbf{f} \in \mathcal{F}_\lambda} \Big| \frac{1}{n} \sum_{i=1}^n \sigma_i (\Psi_{Y_i}(\mathbf{f}(X_i)) - 1) \Big| \leq 2L \sum_{c=1}^K \mathbb{E} \sup_{\mathbf{f} \in \mathcal{F}_1} \Big| \frac{1}{n} \sum_{i=1}^n \sigma_i f_c(X_i) \Big|. \tag{11}$$

Since any $f_c = \sum_j \alpha_j T_c(h_j)$ is a convex combination of $T_c(h_j)$ with $h_j \in \mathcal{H}$, it follows that

$$\sup_{\mathbf{f}\in\mathcal{F}_1} \Big| \frac{1}{n}\sum_{i=1}^n \sigma_i f_c(X_i) \Big| = \sup_{h\in\mathcal{H}} \Big| \frac{1}{n}\sum_{i=1}^n \sigma_i T_c(h(X_i)) \Big|. \tag{12}$$

With $X_i, i = 1,\ldots,n$, fixed, $\sum_{i=1}^n \sigma_i T_c(h(X_i))$ is a sum of n independent zero mean random variables bounded between -1 and $K-1$. The coefficients satisfy $T_c(h(X_i)) = K-1-KI_{\{h(X_i)\neq c\}}$. By a version of the Vapnik-Chervonenkis inequality we conclude

$$\mathbb{E}\sup_{h\in\mathcal{H}} \Big| \frac{1}{n}\sum_{i=1}^n \sigma_i T_c(h(X_i)) \Big| \le (K-1)\sqrt{\frac{2V_{\mathcal{A}_c}\ln(4n+2)}{n}}, \qquad c = 1,\ldots,K.$$

The details are referred to Lugosi and Vayatis (2004). Summing the last inequalities for $c = 1,\ldots,K$ and appealing to (11) and (12) we prove (9).

It is easily seen that the random variable $\sup_{\mathbf{f}\in\mathcal{F}_\lambda} |\mathcal{E}(\mathbf{f}) - \mathcal{E}_n(\mathbf{f})|$ satisfies the bounded difference assumption with constant $c_i = 2(K-1)\phi(-\lambda K)/n, 1 \le i \le n$. Now inequality (10) follows from (9) and McDiarmid's bounded difference inequality (see Lugosi, 2002; McDiarmid, 1989). The proof is complete. ■

We are in a position to establish the consistency.

Theorem 3.3 *Suppose that the condition of Theorem 2.4 hold for ϕ and that $\mathcal{H}$ satisfies $V_{\mathcal{A}_c} < \infty$ for $c = 1,\ldots,K$. Choose λ_n such that $\lambda_n \to \infty$ and $\lambda_n\phi'(-\lambda_n(K-1))\sqrt{\frac{\ln n}{n}} \to 0$ as $n \to \infty$. Assume that, for any n samples $\{(X_1,Y_1),\ldots,(X_n,Y_n)\}$, there exists an $\hat{\mathbf{f}}_n \in \mathcal{F}_{\lambda_n}$ such that*

$$\mathcal{E}_n(\hat{\mathbf{f}}_n) \le \inf_{\mathbf{f}\in\mathcal{F}_{\lambda_n}} \mathcal{E}_n(\mathbf{f}) + \varepsilon_n, \tag{13}$$

where ε_n is a sequence of positive numbers converging to zero. Then under Assumption 3.1, we have the consistency

$$\lim_{n\to\infty} \mathbb{E}\mathcal{R}(C(\hat{\mathbf{f}}_n)) = \mathcal{R}^*.$$

Proof. Denote by $\mathbf{f}_{\lambda_n}$ an element of $\mathcal{F}_{\lambda_n}$ which minimizes $\mathcal{E}(\mathbf{f})$. By (13) we have

$$\begin{aligned} & \mathcal{E}(\hat{\mathbf{f}}_n) - \mathcal{E}(\mathbf{f}_{\lambda_n}) \\ = \;& \mathcal{E}(\hat{\mathbf{f}}_n) - \mathcal{E}_n(\hat{\mathbf{f}}_n) + \mathcal{E}_n(\hat{\mathbf{f}}_n) - \mathcal{E}_n(\mathbf{f}_{\lambda_n}) + \mathcal{E}_n(\mathbf{f}_{\lambda_n}) - \mathcal{E}(\mathbf{f}_{\lambda_n}) \\ \le \;& 2\sup_{\mathbf{f}\in\mathcal{F}_{\lambda_n}} |\mathcal{E}(\mathbf{f}) - \mathcal{E}_n(\mathbf{f})| + \varepsilon_n. \end{aligned}$$

Therefore,

$$\mathbb{E}\mathcal{E}(\hat{\mathbf{f}}_n) \le 2\mathbb{E}\sup_{\mathbf{f}\in\mathcal{F}_{\lambda_n}} |\mathcal{E}(\mathbf{f}) - \mathcal{E}_n(\mathbf{f})| + \mathcal{E}(\mathbf{f}_{\lambda_n}) + \varepsilon_n.$$

With our choice of λ_n, the first term on the right-hand side converges to zero by (9). Also $\mathcal{E}(\mathbf{f}_{\lambda_n}) \to \mathcal{E}^*$ by Assumption 3.1. Thus we have $\mathbb{E}\mathcal{E}(\hat{\mathbf{f}}_n) \to \mathcal{E}^*$. The proof is complete by Theorem 2.4 and the inequality

$$\mathbb{E}(\mathcal{E}(\hat{\mathbf{f}}_n) - \mathcal{E}^*)^{\frac{1}{\beta+1}} \le (\mathbb{E}\mathcal{E}(\hat{\mathbf{f}}_n) - \mathcal{E}^*)^{\frac{1}{\beta+1}}.$$

■

Example 3.4 *The most important choice of ϕ in Theorem 3.3 is $\phi(x) = e^{-x}$. In this case, we thus choose λ_n such that*

$$\lambda_n \to \infty \qquad \text{and} \qquad \lambda_n e^{\lambda_n (K-1)} \sqrt{\frac{\ln n}{n}} \to 0.$$

If the set $\mathcal{H}$ has a finite VC dimension and, for any samples $\{(X_i, Y_i)\}_{i=1}^n$, (13) holds, then we have the consistency stated in Theorem 3.3.

Acknowledgments

The corresponding author is Chen. Sun's current address is Department of Information and Compute Science, Shengli College, China University of Petroleum. The authors thank Prof. P.L. Bartlett, Prof. D.X. Zhou and anonymous referees for their valuable suggestions which improve the paper significantly. The work is supported in part by NSF of China under grant 10571010.

References

P. L. Bartlett, M. I. Jordan and J. D. Mcauliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

O. Bousquet, S. Boucheron and G. Lugosi. Theory of classification: a survey of recent advances. *ESAIM: Probability and Statistics*, 9:323–375. 2005.

D. R. Chen, Q. Wu, Y. Ying and D. X. Zhou. Support vector machine soft margin classifier: error analysis. *J. Machine Learning Reseach*, 5: 1143–1175, 2004.

D. R. Chen and D. H. Xiang. The consistency of multicategory support vector machines. *Adv in Comput. Math.*, 24: 155–169, 2006.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer–Verlag, New York, 1996.

G. Lugosi. *Pattern classification and learning theory, Principles of Nonparametric Learning.* Springer, Wien, New York, pp 1–56, 2002.

G. Lugosi and N. Vayatis. On the Bayes-risk consistency of regularized boosting metheds. *Ann. Statist.*,32: 30–55, 2004.

C. McDiarmid. On the method of bounded differences. In *Surveys on Combinatorics*, 148–188, Combridge University Press, 1989.

A. Tewari and P. L. Bartlett: On the consistency of multiclass classification methods, In *Proc. 18th International Conference on Computational Learning Theory*, pages 143–157, 2005.

V. N. Vapnik. *Statistical Learning Theory.* John Wiley and sons, New York, 1998.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32: 56–134, 2004a.

T. Zhang. Statistical analysis of some multiclass large margin classification method. *Journal of Machine Learning Research*, 5: 1225–1251, 2004b.

Journal of Machine Learning Research 7 (2006) 2449-2480 Submitted 6/06; Revised 10/06; Published 11/06

Bounds for the Loss in Probability of Correct Classification Under Model Based Approximation

Magnus Ekdahl MAEKD@MAI.LIU.SE
Timo Koski TIKOS@MAI.LIU.SE
Matematiska Institutionen
Linköpings Universitet
581 83 Linköping, Sweden

Editor: Michael Jordan

Abstract

In many pattern recognition/classification problem the true class conditional model and class probabilities are approximated for reasons of reducing complexity and/or of statistical estimation. The approximated classifier is expected to have worse performance, here measured by the probability of correct classification. We present an analysis valid in general, and easily computable formulas for estimating the degradation in probability of correct classification when compared to the optimal classifier. An example of an approximation is the Naïve Bayes classifier. We show that the performance of the Naïve Bayes depends on the degree of functional dependence between the features and labels. We provide a sufficient condition for zero loss of performance, too.

Keywords: Bayesian networks, naïve Bayes, plug-in classifier, Kolmogorov distance of variation, variational learning

1. Introduction

Classification procedures based on probability models are widely used in data mining and machine learning (Hand et al., 2001), since such models often lead to effective algorithms and modularity in computation and have a conceptual foundation in statistical learning theory. For tractable computation and learning these models may still in many cases require steps of approximation by less complex model families (Jordan et al., 1999).

By classification we mean procedures that group items represented by a feature vector into different predefined classes. We consider classification procedures based on class conditional probabilities that belong to a model family that does not necessarily contain the true probability distribution, and analyze how the probability of correct classification is affected.

One straightforward procedure is known as a plug-in function. By this we refer to the formal operation performed by the optimal classifier based on Bayes' formula of posterior probabilities of classes, but now plugging in the modeling or approximate class conditional densities as well as approximated class probabilities. There are still a lot of unresolved issues concerning the effects of plug-in functions in the context of classification with high-dimensional feature vectors.

A well known plug-in procedure in classification is modeling by independence, which is usually called the 'Naïve Bayes' classifier. We will review, extend and sharpen the theoretical justification for this procedure while connecting it to the general approximation theory. Friedman (1997) studies also the Naïve Bayes, when the optimal classifier is estimated from training data. He shows that the

bias and variance components of the estimation error affect classification error in a different way under the Gaussian approximation than the error in the estimated probabilities. This can help Naïve Bayes to perform better than expected in case the variance of the estimates of posterior probabilities is low. Our analysis in the sequel will not involve the variance − bias decomposition.

Bayesian networks is a widely used class of models for probabilistic reasoning and for classification, see for example (Korb and Nicholson, 2004; Friedman et al., 1997). As the network topologies increase in size and complexity, the run-time complexity of probabilistic inference and classification procedures becomes prohibitive. In general, exact inference on Bayesian networks is known to be NP-hard (Cooper, 1990). One way of approximating or simplifying the model is to enforce additional conditional independencies or by removing edges in the graph, see van Engelen (1997) and the references therein.

Here we analyze a simplification of Bayesian networks by a strategy of approximating factors of the joint probability, and give a bound for the probability of correct classification under the ensuing plug-in function. This corresponds to some degree to the general heuristics in the work by Lewis (1959); Brown (1959); Chow and Liu (1968); Ku and Kullback (1969), who developed the idea of approximating multivariate discrete probability distributions by a product of lower order marginal distributions. The set of marginal distributions applied needs not be the full set of margins of some order, the requirements are that the product is an extension of the lower order distributions which are compatible.

2. Organization

We will start by introducing notation and basic definitions in Section 3. Section 4 provides rationales and examples of approximating models and plug-in classifiers. These will be used to illustrate the mathematical results in the following sections. Section 5 introduces results about the degradation of classifier performance with respect to the optimal probability of correct classification. The results are phrased in terms of a distance between probabilities known as the Kolmogorov variation distance. There are several well known bounds for the Kolmogorov variation distance by other distances between probability measures, quoted in Section 5, which in many examples yield explicit and computable bounds for the plug-in classifier performance. We give also a novel bound that connects the work to variational learning theory (Jordan et al., 1999). Section 6 gives a rule for potential reduction of the number of dimensions needed for evaluating the degradations, and presents more easily computable bounds. Section 7 discusses the Naïve Bayes classifier by sharpening a bound for Naïve Bayes and connecting it to one of the general approximation bounds in Section 6. Section 8 gives sufficient conditions on the margin (explained later) between two classes, which is used to generalize the possible problems where Naïve Bayes can be argued as optimal.

3. Notation, Bayes and Plug-In Classifiers

Let $(\Omega, \mathcal{F}, P)$ be a probability space, such that (C, X) is a $\mathcal{F}$-measurable stochastic variable, s.v. Let $X = (X_i)_{i=1}^d$, that is X is d-dimensional. When denoting a sample (observation) of X with no missing components we use x, that is $x = (x_i)_{i=1}^d$ (x can be called a feature vector). When referring to the range of X we use $\mathcal{X}$, which for completeness of presentation is assumed to be a Borel space (Schervish, 1995). This assumption is needed to justify the use of results such as the Fubini theorem and the existence of conditional densities.

In the context of classification a sample x is assumed to have a source, one of a family of entities called classes or labels, denoted by c, which is regarded as an outcome of the random variable C. In classification C has range $\mathcal{C} = \{1,\ldots,k\}$, that is, k is the number of classes. We assume, as is common in much of classification theory, that the space of labels $\{1,\ldots,k\}$ is without relevant additional structure except whether two labels are equal or not. In order to resolve ties, it may, on the other hand, be useful to think of the labels as ordered by $1 < 2 < \ldots < k$.

Definition 1 *A classifier is a measurable function $\hat{c} : \mathcal{X} \to \mathcal{C}$ such that given x, $\hat{c}(x)$ is an estimate of c.*

In classification we do not deal directly with the whole sample (c,x), but the class c is a hidden variable. Hence we will deal with the class conditional probability. In

$$P(X \in A|C = c) = \int_A f(x|c)d\mu(x)$$

we call $f(x|c)$ the conditional density of a sample x given that the random variable C equals the label c with respect to the σ-finite measure μ. We assume in other words that μ dominates, see Schervish (1995), the probability measure $P(\cdot|C = c)$ for every c, that is, the same measure μ can be used for all $P(X \cdot |C = c)$ to define the corresponding class conditional density $f(x|c)$. The assumption of domination justifies the validity of a number of formulas of distances between probability measures. $P(c)$ is the short notation for the marginal probability $P(C = c)$. We also encounter $P(c|x)$, the probability of the class c given the sample x. $P(c|x)$ is used to define a classifier which is the cornerstone of probabilistic classification. For example, the procedure known as proportional prediction chooses the label for x by drawing c from the probability mass function $P(c|x)$ (Goodman and Kruskal, 1954). We study only deterministic classifiers $\hat{c}$.

Definition 2 *Bayes classifier for a sample x is*

$$\hat{c}_B(x) = \arg\max_{c\in\mathcal{C}} P(c|x).$$

Ties are resolved in some fixed manner, for example, by taking $\hat{c}_B(x)$ the smallest of the tied labels in (the ordered) $\mathcal{C}$.

The posterior $P(c|x)$ can be modeled directly ('the diagnostic paradigm') but this may often involve difficult computations (Ripley, 1996). Bayes' formula gives effectively

$$\hat{c}_B(x) = \arg\max_{c\in\mathcal{C}} f(x|c)P(c). \tag{1}$$

Thus we base Bayes classifier on $f(x|c)$ as well as on $P(c)$, the prior probability (or the prevalence) of class c. In essence $f(x|c)$ allows us to think of each class as generating x.

We evaluate the performance of a classifier by the probability of correct classification and assess the effect of approximating $f(x|c)$ by $\hat{f}(x|c)$ and $P(c)$ by $\hat{P}(c)$.

Definition 3 *For a classifier $\hat{c}(X)$ the probability of correct classification is $P(\hat{c}(X) = C)$.*

There is a good reason for using Bayes classifier (Definition 2), since for every $\hat{c}(X)$ it holds that

$$P(\hat{c}(X) = C) \leqslant P(\hat{c}_B(X) = C).$$

A simple way of constructing a classifier given $\hat{f}(x|c)$ and $\hat{P}(c)$ is to use these to replace the respective target probabilities in (1).

Definition 4 *$\hat{c}_{\hat{B}}(x)$ is a plug-in classifier with respect to the pair $\left(\hat{f}(x|c), \hat{P}(c)\right)$ if it is defined by*

$$\hat{c}_{\hat{B}}(x) = \arg\max_{c \in C} \hat{f}(x|c)\hat{P}(c). \tag{2}$$

Ties are resolved as in Definition 2.

The question studied here can now be stated as that of computing or bounding the difference

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C).$$

It is for many P difficult or even impossible to compute explicitly $P(\hat{c}_B(X) = C)$. Hence there exists a literature for bounding the optimal probability of error, $P_e^* = 1 - P(\hat{c}_B(X) = C)$. If we set $P_e = 1 - P(\hat{c}_{\hat{B}}(X) = C)$, then

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) = P_e - P_e^*.$$

This can be bounded downwards by, for example, the upper bounds for P_e^* in Bhattacharyya and Toussaint (1982). We shall not pursue the lower bounds for $P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C)$ any further.

4. Examples of Plug-In Approximations of the Bayes Classifier

As outlined in the introduction, there are several reasons for approximating $f(x|c)$ in classification. These include the problem of digitally storing probability tables, the topic introduced in Lewis (1959), and the complexity, or even infeasibility, of computing $\hat{c}_B(x)$. Therefore we could call f the target density and $\hat{f}$ the tractable density (Wainwright and Jordan, 2003). In block transmission systems a tractable density is found for fast computation of the signal classifier (detector) (Kaleh, 1995). In this section we present some examples of plug-in classifiers motivated by these considerations in pattern recognition and detection.

Example 1 *We consider $\mathcal{X} = \{0,1\}^d$ known as the binary hypercube in d dimensions. For the binary hypercube we need in general $2^d - 1$ parameters to specify each class conditional probability mass function. Hence we may encounter a difficulty with storing of the tables of probabilities.*

There are several canonical representations of the generic probability distribution on $\mathcal{X}$ and of the $2^d - 1$ parameters. Examples of these are given in Bahadur (1961b), Devroye et al. (1996), Ott and Kronmal (1976), and Teugels (1990). We recapitulate the representation by Bahadur (1961b) in the form given by Brunk and Pierce (1974). Let $f(x|c)$ be a probability mass function on $\mathcal{X}$ such that $f(x|c) > 0$ for all $x \in \mathcal{X}$. Let

$$f_{ic} = \sum_{x \in \mathcal{X}, x_i = 1} f(x|c), \quad y_{ic} = y_{ic}(x) = \frac{x_i - f_{ic}}{\sqrt{f_{ic}(1 - f_{ic})}}. \tag{3}$$

Let $\mathbf{w} = (w_1, w_2, \ldots, w_d) \in \{0,1\}^d$ *be a binary vector of zeros and ones. Then we denote by* $U_{\mathbf{w},c}(x)$ *products of a subset of* $y_{1c}, \ldots, y_{dc}$

$$U_{\mathbf{w},c}(x) = \prod_{i=1}^{d} y_{ic}(x)^{w_i}, \quad U_{\mathbf{0},c}(x) = 1.$$

We set

$$f_1(x|c) = \prod_{i=1}^{d} f(x_i|c) = \prod_{i=1}^{d} f_{ic}^{x_i}(1-f_{ic})^{1-x_i}. \tag{4}$$

Hence $f_1(x|c)$ *is another probability mass function, which is positive on* $\{0,1\}^d$*. Its marginal distributions coincide with those of* $f(x|c)$*. With respect to* f_1 *any binary random vector* $X = (X_i)_{i=1}^d$ *consists of independent components* X_i*.*

We shall next regard the set V *of real-valued functions on* $\{0,1\}^d$ *as a vector space of dimension* 2^d*. Let us equip* V *with the scalar product defined for* $\phi \in V, \psi \in V$ *as*

$$(\phi, \psi) = \sum_{x \in \{0,1\}^d} \phi(x)\,\psi(x)\, f_1(x|c). \tag{5}$$

Next we show that the functions $\{U_{\mathbf{w},c}(x)\}_{\mathbf{w}\in\{0,1\}^d}$ *constitute an orthonormal basis with respect to this scalar product. In fact*

$$(U_{\mathbf{w},c}, U_{\mathbf{w}^*,c}) = \sum_{x\in\{0,1\}^d} U_{\mathbf{w},c}(x) U_{\mathbf{w}^*,c}(x)\, f_1(x|c) =$$

$$= \sum_{x\in\{0,1\}^d} \prod_{i=1}^{d} y_{ic}(x)^{w_i} y_{ic}(x)^{w_i^*}\, f_1(x|c).$$

The sum in the right hand side is nothing but the expectation

$$E_{f_1}\left[\prod_{i=1}^{d} y_{ic}(X)^{w_i} y_{ic}(X)^{w_i^*}\right] = \prod_{i=1}^{d} E_{f_1}\left[y_{ic}(X)^{w_i} y_{ic}(X)^{w_i^*}\right], \tag{6}$$

where we used the aforementioned independence of the components of X *under* f_1*, which yields the independence of the* $y_{ic}(X)$ *as defined by (3), too. We now show that the product in (6) equals zero, if* $\mathbf{w} \neq \mathbf{w}^*$*. In this case there is at least one* i *such that* $w_i \neq w_i^*$*, and for this* i *we get*

$$E_{f_1}\left[y_{ic}(X)^{w_i} y_{ic}(X)^{w_i^*}\right] = E_{f_1}\left[y_{ic}(X)\right] = \frac{E_{f_1}[X_i] - f_{ic}}{\sqrt{f_{ic}(1-f_{ic})}} = 0,$$

since by the definitions above $E_{f_1}[X_i] = 1 \cdot P_1(X_i = 1) = f_{ic}$*. Hence the whole product in (6) is zero, and we have shown that* $(U_{\mathbf{w},c}, U_{\mathbf{w}^*,c}) = 0$*, if* $\mathbf{w} \neq \mathbf{w}^*$*. If* $\mathbf{w} = \mathbf{w}^*$*, then we get from (6) that*

$$(U_{\mathbf{w},c}, U_{\mathbf{w},c}) = \prod_{i=1:w_i=1} E_{f_1}\left[y_{ic}(X)^2\right].$$

Here

$$E_{f_1}\left[y_{ic}(X)^2\right] = \frac{1}{f_{ic}(1-f_{ic})} E_{f_1}\left[(X_i - f_{ic})^2\right]$$

But since X_i is a binary random variable (or, a Bernoulli random variable) with respect to f_1, we have

$$E_{f_1}\left[(X_i - f_{ic})^2\right] = f_{ic} - f_{ic}^2 = f_{ic}(1 - f_{ic}).$$

Hence $(U_{\mathbf{w},c}, U_{\mathbf{w},c}) = 1$, and we have shown that $\{U_{\mathbf{w},c}(x)\}_{\mathbf{w}\in\{0,1\}^d}$ is an orthonormal set in V with respect to the scalar product in (5). Since the number of functions in $\{U_{\mathbf{w},c}(x)\}_{\mathbf{w}\in\{0,1\}^d}$ equals the dimension of V (=2^d), $\{U_{\mathbf{w},c}(x)\}_{\mathbf{w}\in\{0,1\}^d}$ must be an orthonormal basis in V.

Hence every function $\phi \in V$ has a unique expansion in terms of the 2^d coordinates $(\phi, U_{\mathbf{w},c})$ with respect to this basis written as

$$\phi(x) = \sum_{\mathbf{w}\in\{0,1\}^d} (\phi, U_{\mathbf{w},c}) U_{\mathbf{w},c}(x). \tag{7}$$

If we take $\phi(x) = f(x|c)/f_1(x|c)$ we obtain

$$\left(\frac{f}{f_1}, U_{\mathbf{w},c}\right) = \sum_{x\in\{0,1\}^d} f(x|c) U_{\mathbf{w},c}(x) = E_f\left(U_{\mathbf{w},c}(X)\right).$$

In other words the coordinate $\left(\frac{f}{f_1}, U_{\mathbf{w},c}\right)$ equals the expectation of $U_{\mathbf{w},c}(X)$ w.r.t. to the probability mass function $f(x|c)$. For this we introduce the standard notation

$$\beta_{\mathbf{w},c} = E_f\left(U_{\mathbf{w},c}(X)\right). \tag{8}$$

By substitution of (8) in (7) we obtain

$$\frac{f(x|c)}{f_1(x|c)} = \sum_{\mathbf{w}\in\{0,1\}^d} \beta_{\mathbf{w},c} U_{\mathbf{w},c}(x).$$

This gives us the the (Bahadur-Lazarsfeld) representation of any positive probability mass function $f(x|c)$ on $\{0,1\}^d$ as

$$f(x|c) = f_1(x|c) f_{c,\text{interactions}}(x), \tag{9}$$

where we have written

$$f_{c,\text{interactions}}(x) = \sum_{\mathbf{w}\in\{0,1\}^d} \beta_{\mathbf{w},c} U_{\mathbf{w},c}(x). \tag{10}$$

The rank $R(\mathbf{w})$ of the polynomial $U_{\mathbf{w},c}$ is defined as

$$R(\mathbf{w}) = \sum_{i=1}^{d} w_i.$$

Here $\beta_{\mathbf{0},c} = 1$, and if $R(\mathbf{w}) = 1$, then $\beta_{\mathbf{w},c} = 0$. The probability mass function $f_1(x|c)$ in (4) is known as the first order term. For $R(\mathbf{w}) = 2$ the coefficients $\{\beta_{\mathbf{w}}\}$ are correlations. We can think of the coefficients β as interactions of order $R(\mathbf{w})$ minus one.

One can define a family of probability mass functions called kth order Bahadur distributions as the set of all probabilities on the binary hypercube in d dimensions such that $\beta_{\mathbf{w}} = 0$ for $R(\mathbf{w}) > k$. Anoulova et al. (1996) prove, simplifying their statement, that there is an algorithm that, given enough samples, computes for any $\varepsilon > 0$ a plug-in classifier $\hat{c}_{\hat{B}}(x)$ such that $P(\hat{c}_B(X) = C) -$

$P(\hat{c}_{\hat{B}}(X) = C) \leqslant \varepsilon$, *when the conditional distributions of* $X|C$ *are in the class of kth order Bahadur distributions.*

If we expand $\log \frac{f(x|c)}{f_1(x|c)}$ *with respect to the basis* $\{U_{\mathbf{w},c}(x)\}_{\mathbf{w}\in\{0,1\}^d}$ *we obtain the following canonical form*

$$f(x|c) = f_1(x|c)e^{\sum_{\mathbf{w}\in\{0,1\}^d} \alpha_{\mathbf{w},c}U_{\mathbf{w},c}(x)}, \tag{11}$$

where it follows similarly as above that

$$\alpha_{\mathbf{w},c} = E_{f_1}\left[\log \frac{f(X|c)}{f_1(X|c)} \cdot U_{\mathbf{w},c}(X)\right]. \tag{12}$$

The two canonical forms (10) and (11) above are of interest in the sequel for defining structures of approximations and for evaluating the effect of a plug-in classifier on probability of correct classification. First, the plug-in classifier

$$\hat{c}_{\hat{B}}(x) = \arg\max_{c\in C} f_1(x|c)\hat{P}(c)$$

is an instance of the Naïve Bayes procedure to be treated in more generality in Section 7 below. In the setting of the binary hypercube the Naïve Bayes is said to take into account only the first order term. A survey of the Naïve Bayes in supervised and unsupervised learning of bacterial taxonomies using binary features is found in Gyllenberg and Koski (2001). Further structures of plug-in classifiers can be defined by adding sets of higher order interactions to the first order term. Examples of this are found in Bahadur (1961a), Chow and Liu (1968), Moore (1973), and Ott and Kronmal (1976). Here the trade-off is between the additional complexity and the more accurate statistical description, and, as it will turn out in the sequel, higher probability of correct classification with the plug-in classifier.

A successful empirical application of the Bahadur representation in classification or diagnosis of six diseases using eleven features or symptoms is reported in Scheinck (1972). The underlying requirement $f(x|c) > 0$ *for all* $x \in \{0,1\}^{11}$ *is possibly overlooked in Scheinck (1972).*

In some of the contributions referred to in the above the approximating structure is not necessarily a probability, since an arbitrary truncation of a representation of a probability mass function with respect to a basis is not always a probability mass function.

In case the support of f *is a true subset of* $\mathcal{X} = \{0,1\}^d$*, a canonical representation (an interpolator) of* f *has been reported in Pistone et al. (2001). This is based on the monomials* $\prod_{i=1}^d x_i^{w_i}$ *and the properties of Gröbner bases.*

Example 2 *One model of intersymbol interference (ISI) channels in digital communication theory, see Kaleh (1995); Barbosa (1989), can be formulated as observing a* $d \times 1$ *vector* X *with the class conditional normal distribution*

$$X|C = \mathbf{b} \sim N(H\mathbf{b}, \Sigma),$$

where $\mathbf{b}$ *is* $N \times 1$ *vector such that* $b_i \in \{-1,+1\}$*, and* Σ *is a positive definite* $d \times d$ *matrix, and* H *represents a linear, time-invariant and causal ISI channel by the* $d \times N$ *matrix*

$$H = \begin{pmatrix} h_0 & 0 & \dots & 0 \\ h_1 & h_0 & \ddots & \vdots \\ \vdots & h_1 & \ddots & 0 \\ h_{L-1} & \vdots & \ddots & h_0 \\ 0 & h_{L-1} & \ddots & h_1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h_{L-1} \end{pmatrix}.$$

Here L *is the length of the channel memory, if* $h_0 \neq 0$ *and* $h_{L-1} \neq 0$*. Hence* $d = L+N-1$*. The set of labels* $\mathcal{C}$ *equals in this case a subset of* $\{-1,+1\}^N$*.* $\mathcal{C}$ *might be called a codebook. If all* $\mathbf{b}$ *are equally likely a priori, we have (the optimal detector)*

$$\hat{c}_B(x) = \arg\min_{\mathbf{b}\in\mathcal{C}} \|\Sigma^{-1}(x - H\mathbf{b})\|^2,$$

where $\|x\| = \sqrt{x^T x}$*.*

A suboptimal detector may be introduced, for example, for the purpose of reducing run time complexity, see Barbosa (1989), by a $d \times N$ *matrix* M *of the same structure as* H*, but with a shorter memory and the plug-in classifier*

$$\hat{c}_{\hat{B}}(x) = \arg\min_{\mathbf{b}\in\mathcal{C}} \|\Sigma^{-1}(x - M\mathbf{b})\|^2.$$

Here explicit expressions for both $P(\hat{c}_B(X) = C)$ *and* $P(\hat{c}_{\hat{B}}(X) = C)$ *are readily found, and the question of developing techniques for estimating the loss of performance incurred by the introduction of the suboptimal detector has been studied extensively for a number of designs of the matrices* M*.*

5. A Performance Bound

There are several representations of the exact difference $P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C)$.
For typographical and readability reasons we will use the notation $\hat{c}_B(x) = b$ as well as $\hat{c}_{\hat{B}}(x) = \hat{b}$.
We can write $P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C)$ as

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) = \int_{\{\hat{b}\neq b\}} \left(P(b) f(x|b) - P(\hat{b}) f(x|\hat{b})\right) d\mu(x). \tag{13}$$

We may also re-write this as

$$= \int_{\{\hat{b}\neq b\}} \left(P(b) f(x|b) - \hat{P}(b) \hat{f}(x|b)\right) d\mu(x) - \int_{\{\hat{b}\neq b\}} \left(P(\hat{b}) f(x|\hat{b}) - \hat{P}(\hat{b}) \hat{f}(x|\hat{b})\right) d\mu(x)$$

$$- \int_{\{\hat{b}\neq b\}} \left(\hat{P}(\hat{b}) \hat{f}(x|\hat{b}) - \hat{P}(b) \hat{f}(x|b)\right) d\mu(x) \tag{14}$$

since

$$= \int_{\{\hat{b}\neq b\}} P(b) f(x|b) d\mu(x) - \int_{\{\hat{b}\neq b\}} \hat{P}(b) \hat{f}(x|b) d\mu(x)$$

$$-\int_{\{\hat{b}\neq b\}} P(\hat{b})\,f(x|\hat{b})d\mu(x)+\int_{\{\hat{b}\neq b\}} \hat{P}(\hat{b})\,\hat{f}(x|\hat{b})d\mu(x)$$

$$-\int_{\{\hat{b}\neq b\}} \hat{P}(\hat{b})\,\hat{f}(x|\hat{b})d\mu(x)+\int_{\{\hat{b}\neq b\}} \hat{P}(b)\,\hat{f}(x|b)d\mu(x),$$

where 4 integrals cancel each other and (13) is formed. The difficulty with these expressions is to find the set $\{x|\hat{c}_B(x)\neq\hat{c}_{\hat{B}}(x)\}$ and to compute the integrals above.

We give next a first upper bound for $P(\hat{c}_B(X)=C)-P(\hat{c}_{\hat{B}}(X)=C)$. The result for the specific case of $k=2$ is presented in Ryzin (1966). When $k\geqslant 2$ the result in Theorem 5 can basically be found inside a proof in Glick (1972). A proof is included here for completeness and readability. For the specific approximation, where $\hat{f}$ and $\hat{P}$ are the maximum likelihood estimators, and samples are discrete, rates of convergence are provided in Glick (1973), as the sample size increases to infinity.

Theorem 5

$$P(\hat{c}_B(X)=C)-P(\hat{c}_{\hat{B}}(X)=C)\leqslant E_{\hat{f}\hat{P}}\left|\frac{f(X|C)P(C)}{\hat{f}(X|C)\hat{P}(C)}-1\right|. \tag{15}$$

Proof From (14) $P(\hat{c}_B(X)=C)-P(\hat{c}_{\hat{B}}(X)=C)=$

$$\int_{\{x|\hat{b}\neq b\}}\left(P(b)\,f(x|b)-\hat{P}(b)\,\hat{f}(x|b)\right)d\mu(x)$$

$$-\int_{\{x|\hat{b}\neq b\}}\left(P(\hat{b})\,f(x|\hat{b})-\hat{P}(\hat{b})\,\hat{f}(x|\hat{b})\right)d\mu(x)-\int_{\{x|\hat{b}\neq b\}}\left(\hat{f}(x|\hat{b})P(\hat{b})-\hat{f}(x|b)P(b)\right)d\mu(x).$$

Definition 4 implies that $\hat{f}(x|\hat{b})\hat{P}(\hat{b})\geqslant\hat{f}(x|b)\hat{P}(b)$, hence

$$\leqslant\int_{\{x|\hat{b}\neq b\}}\left(P(b)\,f(x|b)-\hat{P}(b)\,\hat{f}(x|b)\right)d\mu(x)-\int_{\{x|\hat{b}\neq b\}}\left(P(\hat{b})\,f(x|\hat{b})-\hat{P}(\hat{b})\,\hat{f}(x|\hat{b})\right)d\mu(x).$$

To simplify further $\int a-e\leqslant|\int a-e|\leqslant|\int a|+|\int e|\leqslant\int|a|+\int|e|$ is used, resulting in

$$\leqslant\int_{\{x|\hat{b}\neq b\}}\left|P(b)\,f(x|b)-\hat{P}(b)\,\hat{f}(x|b)\right|d\mu(x)+\int_{\{x|\hat{b}\neq b\}}\left|P(\hat{b})\,f(x|\hat{b})-\hat{P}(\hat{b})\,\hat{f}(x|\hat{b})\right|d\mu(x).$$

Then divide into cases where b as well as $\hat{b}$ are constant (they both depend on x)

$$=\sum_{c=1}^{k}\left[\int_{\left\{x|b\neq\hat{b}\bigcap b=c\right\}}\left|P(c)\,f(x|c)-\hat{P}(c)\,\hat{f}(x|c)\right|d\mu(x)\right.$$

$$\left.+\int_{\left\{x|b\neq\hat{b}\bigcap\hat{b}=c\right\}}\left|P(c)\,f(x|c)-\hat{P}(c)\,\hat{f}(x|c)\right|d\mu(x)\right].$$

Now $b\neq\hat{b}\bigcap b=c$ and $b\neq\hat{b}\bigcap\hat{b}=c$ are disjoint sets so we can write both integrals as one integral,

$$=\sum_{c=1}^{k}\int_{\left\{x|b\neq\hat{b}\bigcap\left(b=c\bigcup\hat{b}=c\right)\right\}}\left|P(c)\,f(x|c)-\hat{P}(c)\,\hat{f}(x|c)\right|d\mu(x)$$

We want an approximation that does not depend on $b, \hat{b}$, such as

$$\leqslant \sum_{c=1}^{k} \int_{\mathcal{X}} \left| P(c) f(x|c) - \hat{P}(c) \hat{f}(x|c) \right| d\mu(x) = \sum_{c=1}^{k} \int_{\mathcal{X}} \hat{P}(c) \hat{f}(x|c) \left| \frac{P(c) f(x|c)}{\hat{P}(c) \hat{f}(x|c)} - 1 \right| d\mu(x).$$

■

The right hand side of the inequality (15) is, when multiplied by the factor $1/2$, an instance of what is being called the Kolmogorov distance of variation, see for example Ali and Silvey (1966). We shall resort to this terminology in order to be able to refer concisely to the quantity in the right hand side of (15) (or of (16) below). The basic mathematical properties of this distance are found in Strasser (1985). Probabilistically the size of the quantity $P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C)$ in (15) is thus interpreted as the expected dispersion of $\frac{f(X|C)P(C)}{\hat{f}(X|C)\hat{P}(C)}$ around unity with respect to the approximating distribution $\hat{f}\hat{P}$.

The result above is the starting point of our development of approximations of probability to find plug-in classifiers for Bayesian networks and in particular to evaluate Naïve Bayes. We note that

$$E_{\hat{f}\hat{P}} \left| \frac{f(X|C)P(C)}{\hat{f}(X|C)\hat{P}(C)} - 1 \right| = \sum_{c=1}^{k} \int_{\mathcal{X}} \hat{f}(x|c)\hat{P}(c) \left| \frac{f(x|c)P(c)}{\hat{f}(x|c)\hat{P}(c)} - 1 \right| d\mu(x)$$

$$= \sum_{c=1}^{k} \int_{\mathcal{X}} \left| f(x|c)P(c) - \hat{f}(x|c)\hat{P}(c) \right| d\mu(x), \tag{16}$$

which is the bound in (15) written as in Ryzin (1966) and Glick (1972). We shall, next present examples of evaluating the bound directly.

Example 3 *Let again as in Example 1 take $\mathcal{X}$ as the binary hypercube in d dimensions. We assume that the true density (with respect to the counting measure μ) $f(x|c) > 0$ for all $x \in \mathcal{X}$ and $P(c) = \hat{P}(c)$. When we approximate this density by its first order term $f_1(x|c)$ in (4) we get from (9) and (10) that*

$$\left| f(x|c)P(c) - \hat{f}(x|c)\hat{P}(c) \right| = P(c) f_1(x|c) \left| \sum_{\mathbf{w} \neq \mathbf{0}} \beta_{\mathbf{w},c} U_{\mathbf{w},c}(x) \right|.$$

In words, here the bound in Theorem 5 expresses the deterioration of the classifier performance by means of a sum of all interactions of order higher than one. Since the measure μ is the counting measure we have the bound

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant \sum_{c=1}^{k} P(c) \sum_{x \in \mathcal{X}} f_1(x|c) \left| \sum_{\mathbf{w} \neq \mathbf{0}} \beta_{\mathbf{w},c} U_{\mathbf{w},c}(x) \right|.$$

Example 4 *The Kolmogorov distance of variation is very effectively evaluated and bounded for the class of two dimensional densities having an expansion with respect to an orthonormal system of polynomials. A diagonal expansion is possible, for example, for Gaussian, sinusoidal and Pearson type II distributions, see McGraw and Wagner (1968), which also contains an extensive list of references on the subject.*

We take as an illustration the two dimensional Gaussian density. Hence $X = (X_i)_{i=1}^2$, *and* $\mathcal{X} = \mathbb{R} \times \mathbb{R}$. *The density* $f(x)$ *is determined by the respective variances* σ_1^2 *and* σ_2^2, *the respective means* m_1 *and* m_2, *and the coefficient of correlation* ρ. *Let* $\hat{f}(x) = f_1(x_1) f_2(x_2)$ *be the product of the two Gaussian marginal densities for* X_1 *and* X_2. *This corresponds again to an instance of the Naïve Bayes procedure. Then the classical Mehler expansion (Cramér, 1966) says for* $|\rho| < 1$ *that*

$$f(x) = f_1(x_1) \cdot f_2(x_2) \cdot \sum_{n=0}^{\infty} H_n\left(\frac{x_1 - m_1}{\sigma_1}\right) \cdot H_n\left(\frac{x_2 - m_2}{\sigma_2}\right) \cdot \frac{\rho^n}{n!},$$

where $H_n(x)$ *is the Hermite polynomial of order n, defined as* $H_n(x) = (-1)^n e^{x^2} \frac{d^n}{x^n} e^{-x^2}$ *for* $n = 0, 1, \ldots,$. *If we assume* $P(c) = \hat{P}(c)$, *then Theorem 5 entails, since* $H_0(x) = 1$,

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C)$$

$$\leqslant \sum_{c=1}^{k} P(c) \int_{\mathbb{R}\times\mathbb{R}} f_1(x_1|c) f_2(x_2|c) \left| \sum_{n=1}^{\infty} H_n\left(\frac{x_1 - m_1(c)}{\sigma_1}\right) \cdot H_n\left(\frac{x_2 - m_2(c)}{\sigma_2}\right) \cdot \frac{\rho^n(c)}{n!} \right| dx_1 dx_2,$$

where the means and coefficient of correlation are chosen to depend on c. The bound on the difference between the probabilities of correct classification is seen to be a power series in the absolute value of the coefficient of correlation. There are computational routines for the Hermite polynomials, and in addition integrals of the form $\int_{\mathbb{R}} \mid x_i - m_i \mid^k f_i(x_i) dx_i$ *involved here are explicitly computable. There is an extension of the Mehler expansion for n-variate densities (Slepian, 1972), which could be used in some of the examples below, but we will not expand on this due to the extensive notational machinery thereby required.*

There are certain well known inequalities between the Kolmogorov distance of variation and other distances or divergences between probability measures. These distances are often readily computable in an explicit form, a compendium is recapitulated in Kailath (1967). An up-to-date discussion of the inequalities to be presented below and several others is found in Topsoe (2000). Nguyen et al. (2005) have presented techniques of replacing the Bayesian probability of error by more general risk functions and analyzing them with corresponding divergences, which are surveyed in Topsoe (2000).

For two probability densities f and $\hat{f}$ we have the inequality due to Ch. Kraft, see Hoeffding and Wolfowitz (1958); Pitman (1979),

$$\frac{1}{2} \int_{\mathcal{X}} \left| f(x) - \hat{f}(x) \right| d\mu(x) \leqslant \sqrt{1 - \left[\int_{\mathcal{X}} \sqrt{f(x) \cdot \hat{f}(x)} d\mu(x) \right]^2}. \tag{17}$$

The quantity $\int_{\mathcal{X}} \sqrt{f(x) \cdot \hat{f}(x)} d\mu(x)$ is known as the affinity or as the Bhattacharyya coefficient or as the Hellinger integral.

We note next that (17) yields in (16)

$$\sum_{c=1}^{k} \int_{\mathcal{X}} \left| f(x|c)P(c) - \hat{f}(x|c)\hat{P}(c) \right| d\mu(x)$$

$$\leqslant 2\sqrt{1 - \left[\sum_{c=1}^{k} \sqrt{P(c)\hat{P}(c)} \int_{\mathcal{X}} \sqrt{f(x|c) \cdot \hat{f}(x|c)} d\mu(x) \right]^2}. \tag{18}$$

The Kullback-Leibler divergence (in natural logarithm) (Kullback, 1997; Cover and Thomas, 1991) defined as

$$D\left(f,\hat{f}\right)=\int_{\mathcal{X}} f(x|c)\log\left(\frac{f(x|c)}{\hat{f}(x|c)}\right)d\mu(x).$$

We have

$$-\frac{1}{2}D(f,\hat{f})=\int_{\mathcal{X}} f(x|c)\log\left(\frac{\hat{f}(x|c)}{f(x|c)}\right)^{\frac{1}{2}}d\mu(x).$$

By Jensen's inequality

$$\leqslant\log\int_{\mathcal{X}} f(x|c)\left(\frac{\hat{f}(x|c)}{f(x|c)}\right)^{\frac{1}{2}}d\mu(x)=\log\int_{\mathcal{X}}\sqrt{f(x|c)\,\hat{f}(x|c)}d\mu(x).$$

Hence

$$\int_{\mathcal{X}}\sqrt{f(x|c)\,\hat{f}(x|c)}d\mu(x)\geqslant e^{-\frac{1}{2}D(f,\hat{f})}.$$

Hoeffding and Wolfowitz (1958) were probably the first to observe this inequality. Furthermore,

$$\sqrt{1-\left[\int_{\mathcal{X}}\sqrt{f(x)\cdot\hat{f}(x)}d\mu(x)\right]^2}\leqslant\sqrt{1-e^{-D(f,\hat{f})}}. \tag{19}$$

In hypothesis testing and pattern classification it is desirable that $D(f_1,f_2)$ is large, or, the affinity is small. The opposite is desirable for plug-in classifiers. By symmetry of the affinity in (19) we get ($D(f,\hat{f})$ need not be equal to $D(\hat{f},f)$) that

$$\sqrt{1-\left[\int_{\mathcal{X}}\sqrt{f(x)\cdot\hat{f}(x)}d\mu(x)\right]^2}\leqslant\sqrt{1-e^{-D(\hat{f},f)}}.$$

Brown (1959) and Ku and Kullback (1969) developed a convergent iteration that finds $\hat{f}$ minimizing $D(\hat{f},f)$ in the class of all densities on discrete $\mathcal{X}$ that have some given set of lower order marginals. The iteration may in several situations be computationally infeasible without constraining f to some suitably simplified model family.

Example 5 *In Example 3 we get by (17) the bound*

$$P(\hat{c}_B(X)=C)-P(\hat{c}_{\hat{B}}(X)=C)\leqslant 2\sum_{c=1}^{k}P(c)\sqrt{1-\left[\sum_{x\in\mathcal{X}}f_1(x|c)\sqrt{\sum_{\mathbf{w}\neq\mathbf{0}}\beta_{\mathbf{w},c}U_{\mathbf{w},c}(x)}\right]^2}.$$

Since we shall need the generic formula in the sequel, we note that the Kullback-Leibler divergence involved in this context is for discrete $\mathcal{X}$

$$D\left(f(x|c)P(c),f_1(x|c)\hat{P}(c)\right)=\sum_{c=1}^{k}\sum_{x\in\mathcal{X}}f(x|c)P(c)\log\frac{f(x|c)P(c)}{f_1(x|c)\hat{P}(c)}$$

$$=\sum_{c=1}^{k}P(c)\sum_{x\in\mathcal{X}}f(x|c)\log\frac{f(x|c)}{f_1(x|c)}+\sum_{c=1}^{k}P(c)\log\frac{P(c)}{\hat{P}(c)} \tag{20}$$

$$= \sum_{c=1}^{k} P(c) D\left(f(x|c), f_1(x|c)\right) + D\left(P(c), \hat{P}(c)\right).$$

Example 6 *We continue with Example 3 but omit the assumption that $P(c) = \hat{P}(c)$. We consider the plug-in classifier with the first order term $f_1(x|c)$ in (4). In view of (11) we get*

$$\begin{aligned} \sum_{x\in\mathcal{X}} f(x|c) \log \frac{f(x|c)}{f_1(x|c)} &= \sum_{x\in\mathcal{X}} f(x|c) \sum_{\mathbf{w}\in\{0,1\}^d} \alpha_{\mathbf{w},c} U_{\mathbf{w},c}(x) \\ = \sum_{\mathbf{w}\in\{0,1\}^d} \alpha_{\mathbf{w},c} \sum_{x\in\mathcal{X}} f(x|c) U_{\mathbf{w},c}(x) &= \sum_{\mathbf{w}\in\{0,1\}^d} \alpha_{\mathbf{w},c} E_f\left[U_{\mathbf{w},c}(X)\right] \\ &= \sum_{\mathbf{w}\in\{0,1\}^d} \alpha_{\mathbf{w},c} \beta_{\mathbf{w},c}, \end{aligned} \tag{21}$$

where we evoked (8). Hence we have by (19) and (20) obtained the following bound for the performance of the Naïve Bayes classifier for binary feature vectors

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant 2\sqrt{1 - e^{-\left\{\sum_{c=1}^{k} P(c) \sum_{\mathbf{w}\in\{0,1\}^d} \alpha_{\mathbf{w},c}\beta_{\mathbf{w},c} + \sum_{c=1}^{k} P(c) \log \frac{P(c)}{\hat{P}(c)}\right\}}}.$$

For f in a kth order Bahadur class in (9) we can often, at least for relatively low d, readily evaluate the bounds above. An observation concerning the expression obtained in (21) is that $U_{\mathbf{0},c} = 1$ gives by (8) that $\beta_{\mathbf{0},c} = 1$, and by (12) that

$$\alpha_{\mathbf{0},c}\beta_{\mathbf{0},c} = -D\left(f_1(x|c), f(x|c)\right).$$

Example 7 *In Example 2 above the true and plug-in densities correspond to the distributions $N(H\mathbf{b}, \Sigma)$ and $N(M\mathbf{b}, \Sigma)$, respectively. Since $\mathcal{C} \subseteq \{-1, +1\}^N$ is a codebook of equally likely vectors $\{\mathbf{b}\}$, we modify the general notation for this example by denoting a label in $\mathcal{C}$ by $\mathbf{b}$. Hence $\hat{P}(\mathbf{b}) = P(\mathbf{b}) = \frac{1}{|C|}$, where $|C|$ is the cardinality of the codebook.*

In this example we use the bound (18). The expression for the required affinity is well known (Kailath, 1967) and equals

$$\int_{\mathbb{R}^d} \sqrt{f(x|\mathbf{b}) \cdot \hat{f}(x|\mathbf{b})} d\mu(x) = e^{-\frac{1}{4} D(N(H\mathbf{b},\Sigma), N(M\mathbf{b},\Sigma))}, \tag{22}$$

where $D\left(N(H\mathbf{b}, \Sigma), N(M\mathbf{b}, \Sigma)\right)$ is in fact the Kullback-Leibler divergence

$$D\left(N(H\mathbf{b}, \Sigma), N(M\mathbf{b}, \Sigma)\right) = \frac{1}{2}\left((H - M)\mathbf{b}\right)^T \Sigma^{-1}\left((H - M)\mathbf{b}\right), \tag{23}$$

see Kullback (1997)). Therefore we obtain for the plug-in classifier (suboptimal detector) defined in Example 2 by (18) that

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant 2\sqrt{1 - \frac{1}{|C|} \sum_{\mathbf{b}\in C} e^{-\frac{1}{8}((H-M)\mathbf{b})^T \Sigma^{-1} ((H-M)\mathbf{b})}}. \tag{24}$$

This expression can be used to compare different designs of suboptimal detectors represented by their respective matrices M.

We want a way to calculate $P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C)$ in terms of (only) $\hat{f}(x|c)\hat{P}(c)$. We will generalize the result in Theorem 5 (15) in the sense that it can be used when only certain parts in a factorization of $\hat{f}(x|c)$ are approximations. What we mean by 'components' will be made precise later.

6. Approximating Bayesian Networks in Classification

While (14) is an exact expression of $P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C)$ it might be infeasible to calculate it in practice. Thus we introduce more easily computable bounds in Corollary 8 and Theorem 11. To avoid making these approximations, or bounds upwards too loose, we also take into account the case where we have not approximated all of $P(c|x)$ (Theorem 7). If the factor of $P(c|x)$ that is not approximated is not functionally dependent of the factor that is approximated, we can develop sharper bounds on the degradation of the approximated classifier performance. Here we consider a class of general approximations, applicable to Bayesian networks (Cowell et al., 1999). For ease of reference we recapitulate first the definition of Bayesian networks.

Definition 6 *Given a directed acyclic graph $G = (V,E)$ with $\{X_i\}_{i=1}^d$ designating (the random variables at) the vertices, Π_i denotes the set of parents of X_i in the graph G. We use π_i to denote the parents states. The pair (G,P), is called a Bayesian network and satisfies*

$$f(x|c,G) = \prod_{i=1}^{d} f(x_i|\pi_i,c,G). \tag{25}$$

Williamson (2005) suggests the following method of approximation. $f(x|c)$ denotes a generic target probability mass function and G is a directed acyclic graph. In principle one can compute the probabilities $f(x_i|\pi_i,c)$ on G using f. Then $\hat{f}(x|c)$ is an approximating probability obtained by taking

$$\hat{f}(x_i|\pi_i,c,G) = f(x_i|\pi_i,c), \tag{26}$$

and multiplying $\hat{f}(x_i|\pi_i,c,G)$ according to (25). The best approximating G (in some family of directed acyclic graphs) is found by maximizing

$$\sum_{x\in\mathcal{X}} f(x|c) \sum_{i=1}^{d} \log \frac{f(x_i,\pi_i|c)}{f(x_i|c)\,f(\pi_i|c)},$$

which is shown to minimize $D\left(f,\hat{f}\right)$. This constitutes also a method of learning network structures, in case there is an effective algorithmic implementation.

We give next a few examples of Bayesian networks, which will also be used to illustrate some of the results in the sequel.

Example 8 *As in Example 1 we take $\mathcal{X} = \{0,1\}^d$ and assume that G is a rooted tree. We order the variables so that x_1 is the state at the root. Direction is defined from parent to child. In a rooted tree any node i, except for the root, has exactly one parent node $\pi(i) < i$. We write the parent state as $\pi_i = x_{\pi(i)}$. Then the factorization in (25) becomes*

$$f(x|c,G) = f(x_1|c) \prod_{i=2}^{d} f(x_i|x_{\pi(i)},c,G). \tag{27}$$

In other words this is a joint probability factorized along a rooted tree. This is in the sense of Lewis (1959), as discussed above, a product approximation of a density with d variables with at most two components per factor. We can also talk about a tree dependence. This dependence was introduced in Chow and Liu (1968).

We will use the form in Definition 6, (25) to express partial approximations. Let $S = (S_1, \ldots, S_4)$ be a partition of $\{X_i\}_{i=1}^d$, where $s = (s_1, \ldots, s_4)$ denotes the resulting partition of $\{x_i\}_{i=1}^d$.

We designate by $f(s_i|G)$ the class conditional density of all s.v.'s that are in S_i given its parents, that is $f(s_i|G)$ is short notation for

$$\prod_{\{i|X_i \in S_i\}} f(x_i|\pi_i, c, G).$$

When referring to the range of S_i we use $\mathcal{S}_i$. Next we describe an efficient choice of S. We make some relevant definitions.

X_i is an proper ancestor of X_j in G and X_j is a proper descendent of X_i in G if there exist a path from X_i to X_j in G for $i \neq j$. A path is a sequence $A_0, ..., A_n$ of distinct vertices such that $(A_{i-1}, A_i) \in E$ for all $i = 1, ..., n$. Given a Bayesian network (G, P) and $\hat{P}$, the partition S is defined for a class conditional density as follows:

- $X_i \in S_1$ if $f(x_i|\pi_i, c) \neq \hat{f}(x_i|\pi_i, c)$.
- $X_i \in S_2$ if for all x_i, π_i we have $f(x_i|\pi_i, c) = \hat{f}(x_i|\pi_i, c)$ and for all $j \neq i$ such that $X_j \in S_1$ we have $X_i \notin \pi_j$.
- $X_i \in S_3$ if for all x_i, π_i, we have $f(x_i|\pi_i, c) = \hat{f}(x_i|\pi_i, c)$, there exists $j \neq i$ such that $X_j \in S_1$ and $X_i \in \pi_j$. Furthermore no proper ancestor X_k of X_i in $G_{S \setminus S_2}$ is such that $X_k \in S_1$.
- $X_i \in S_4$ if $X_i \notin S_1$, $X_i \notin S_2$ and $X_i \notin S_3$.

Example 9 *We consider the rooted and directed tree in the Example 8. We approximate the joint density in (27) by the product of marginal densities. This corresponds to removing all the edges from the tree, the resulting set of nodes is a degenerate special case of a DAG.*

Then $S_1 = \{2, 3, \ldots, d\}$ and $S_3 = \{1\}$ = the root. The partitioning sets S_2 and S_4 are empty.

Example 10 *Context-Specific Independence in Bayesian networks (Boutilier et al., 1996). In this example $X_i \in \{0, 1\}$ and the graph for the Bayesian network is as in Figure 1. Then X_9 is a context in the sense that*

$$f(x_1|x_5, \ldots, x_9) = \begin{cases} f(x_1|x_5, x_6) & x_9 = 0 \\ f(x_1|x_7, x_8) & x_9 = 1 \end{cases}. \tag{28}$$

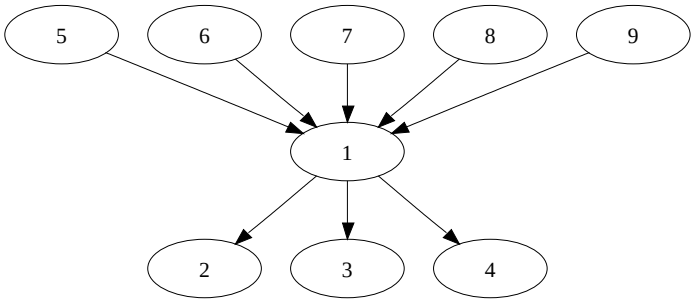
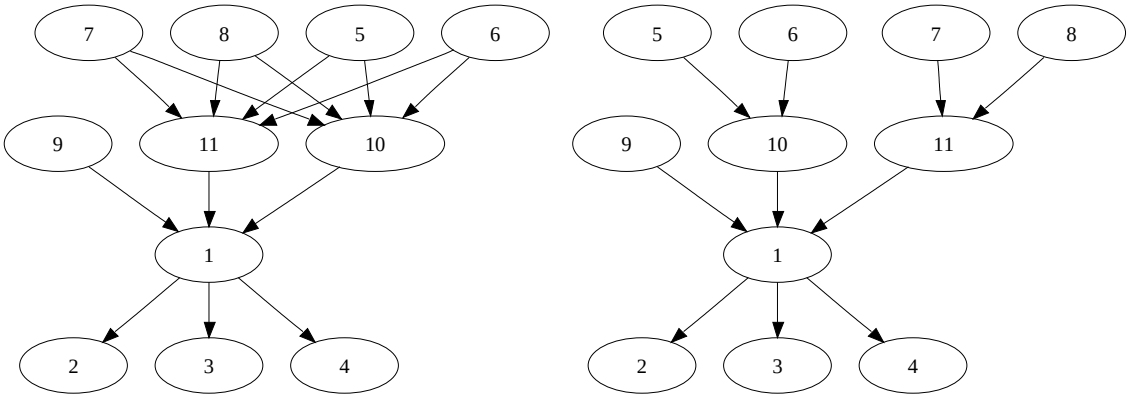


Figure 1: Original Bayesian network



(a) Transformed Bayesian network

(b) Transformed Bayesian network with the CSI assumption

Figure 2: Transformed Bayesian networks with and without the CSI assumption

To encode this in a Bayesian network we transform the original Bayesian network into the network in Figure 2(a). Figure 2(b) describes the graph, where the assumption in (28) holds, where $f(x_{10}|x_5,x_6) = f(x_1|x_5,x_6)$, $f(x_{11}|x_7,x_8) = f(x_1|x_7,x_8)$ *and*

$f(x_1\|x_9,x_{10},x_{10})$	x_9	x_{10}	x_{11}
0	0	0	0
0	0	0	1
1	0	1	0
1	0	1	1
0	1	0	0
1	1	0	1
0	1	1	0
1	1	1	1 .

If the context specific assumption is introduced as an approximation this would yield

$$\begin{cases} S_1 = \{X_{10}, X_{11}\} \\ S_2 = \{X_1, X_2, X_3, X_4, X_9\} \\ S_3 = \{X_5, X_6, X_7, X_8\} \\ S_4 = \{\varnothing\} \end{cases}.$$

Example 11 *In this example we depict a graph G given a partition s, with some abuse of notation. In Figure 3 if* $X_i \in S_1$ *we label the vertex* X_i *as* 1.

Theorem 7 $\int_{\mathcal{X}} \left| P(c) f(x|c) - \hat{P}(c) \hat{f}(x|c) \right| d\mu(x) =$

$$\int_{S_3} \int_{S_1 \times S_4} \left| P(c) f(s_1 \times s_4 | G) - \hat{P}(c) \hat{f}(s_1 \times s_4 | G) \right| d\mu(s_1 \times s_4) d f(s_3|G).$$

Proof We use S to write $\int_{\mathcal{X}} \left| P(c) f(x|c) - \hat{P}(c) \hat{f}(x|c) \right| d\mu(x)$ as

$$= \int_{\mathcal{X}} \left| P(c) f(s_1|G) - \hat{P}(c) \hat{f}(s_1|G) \right| \left[\prod_{j=2}^{4} f(s_j|G) \right] d\mu(x). \tag{29}$$

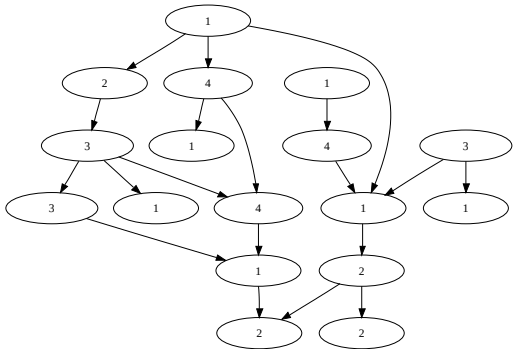


Figure 3: Bayesian network

Now we use the definition of S_2 and the Fubini theorem to write (29) as

$$\int_{\mathcal{S}_1\times\mathcal{S}_3\times\mathcal{S}_4} \left|P(c)f(s_1|G)-\hat{P}(c)\hat{f}(s_1|G)\right| \int_{\mathcal{S}_2}\left[\prod_{j=2}^{4} f(s_j|G)\right] d\mu(s_2)d\mu(s_1\times s_4)d\mu(s_3). \tag{30}$$

We can express the innermost integral as

$$\int_{\mathcal{S}_2} f(s_2\times s_3\times s_4|s_1,G)d\mu(s_2) = f(s_3\times s_4|s_1,G).$$

We continue with (30). Since for all $X_i \in S_3$ there exists no $X_j \in S_1 \bigcup S_4$ such that $X_j \in \pi_i$ we can write this as

$$\int_{\mathcal{S}_3} f(s_3|G) \int_{\mathcal{S}_1\times\mathcal{S}_4} \left|P(c)f(s_1\times s_4|G)-\hat{P}(c)\hat{f}(s_1\times s_4|G)\right| d\mu(s_1\times s_4)d\mu(s_3).$$

■

Since this result is an equality, it seems to indicate that isolated approximations are stable in the sense that the classification error they introduce does only depend on a local neighborhood in the original Bayesian network.

There exist several algorithms that can be used for finding an approximation of a BN such as the ones described in Chow and Liu (1968) and Chickering (2002). If an approximation has been constructed, Theorem 7 makes it possible to evaluate its effect depending on the approximations made, rather than on the original problem.

Next we extend the result in Theorem 5, (15) by specifying the difference in probability of correct classification in terms of the partial structure specific difference through combining Theorem 7 and Theorem 5.

Corollary 8

$$P(\hat{c}_B(X)=C)-P(\hat{c}_{\hat{B}}(X)=C) \leqslant$$

$$\sum_{c=1}^{k}\int_{\mathcal{S}_3}\int_{\mathcal{S}_1\times\mathcal{S}_4} \left|P(c)f(s_1\times s_4|G)-\hat{P}(c)\hat{f}(s_1\times s_4|G)\right| d\mu(s_1\times s_4)d\,f(s_3|G). \tag{31}$$

Of course, it might still be computationally difficult to calculate the bound in Corollary 8. When combining (31), (17) and (19) we obtain the following corollary.

Corollary 9 *If $\hat{P}(c) = P(c)$ for all $c \in \mathcal{C}$,*

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant$$

$$2\sum_{c=1}^{k} P(c) \int_{S_3} \sqrt{1 - e^{-D\left(f(s_1 \times s_4|c,G), \hat{f}(s_1 \times s_4|c,G)\right)}}\, d f(s_3|G).$$

The following examples demonstrate computable expressions for this bound.

Example 12 *We consider the approximation in Example 9, where $S_1 = \{2,3,\ldots,d\}$ and $S_3 = \{1\} =$ the root, and the other partitioning sets are empty. We have from (27)*

$$f(s_1|c,G) = \prod_{i=2}^{d} f(x_i|x_{\pi(i)},c,G),$$

which is a probability mass function on S_1 and

$$\hat{f}\,(s_1|c,G) = \prod_{i=2}^{d} f(x_i|c,G),$$

which is a probability mass function on S_1. Then in view of (19) and (20) we compute

$$D\left(f(s_1|c,G), \hat{f}\,(s_1|c,G)\right) = \sum_{s_1 \in S_1} f(s_1|c,G) \log \frac{f(s_1|c,G)}{\hat{f}\,(s_1|c,G)}$$

$$= \sum_{i=2}^{d} \sum_{x_i, x_{\pi(i)}} f\left(x_i, x_{\pi(i)}|c,G\right) \log \frac{f\left(x_i, x_{\pi(i)}|c,G\right)}{f\left(x_i|c,G\right) \cdot f\left(x_{\pi(i)}|c,G\right)}.$$

Here we recognize, see Cover and Thomas (1991), the mutual informations $I_{c,G}\left(x_i, x_{\pi(i)}\right)$ between x_i and $x_{\pi(i)}$ so that the expression in the right hand side of the preceding equation equals

$$= \sum_{i=2}^{d} I_{c,G}\left(x_i, x_{\pi(i)}\right).$$

It should be noted that this depends on $S_3 = \{1\}$ through those nodes i that have $\pi(i) = 1$. Chow and Liu (1968) developed an algorithm for finding the tree from data that maximizes sum of the mutual informations between a variable and its parents shown above.

Example 13 *The conditionally Gaussian regressions are useful probability models for Bayesian networks with both continuous and discrete variables, see Lauritzen (1990). In order to fit the framework above to these distributions, we suppose that (X_1, X_4) is a vector of r continuous random variables, and that the variables in X_3 are decomposed into the discrete ones in $X_3(\triangle)$ and into the t continuous variables $X_3(\gamma)$. In order not to overburden the notation we omit here the dependence on c in the expressions below.*

The conditionally Gaussian regressions are defined as follows. The conditional distribution of (X_1, X_4) *given* X_3 *is a multivariate normal distribution*

$$(X_1, X_4) \mid \left(X_3(\triangle) = \pi_\triangle, X_3(\gamma) = \pi_\gamma\right) \sim N_r\left(A(\pi_\triangle) + B(\pi_\triangle)\pi_\gamma, \Sigma(\pi_\triangle)\right),$$

where $\pi_\triangle$ *the state of the discrete parents,* π_γ *is the state of the continuous parents, and* $A(\pi_\triangle)$ *is a* $r \times 1$ *vector,* $B(\pi_\triangle)$*, is an* $r \times t$ *matrix,* $\Sigma(\pi_\triangle)$ *is a positive definite symmetric matrix. The Naïve Bayes classifiers approximate* $\Sigma(\pi_\triangle)$ *by a diagonal matrix.*

We illustrate, however, the simplest of the upper bounds with the Kullback-Leibler divergence by taking the plug-in distribution

$$\hat{N}_r\left(A + B(\pi_\triangle)\pi_\gamma, \Sigma(\pi_\triangle)\right).$$

Similarly to what has been done above, or more precisely, using Theorem 7, (18), and (22) above it follows that for $X = (X_i)_{i=1}^4$ *corresponding to the decomposition* $S = (S_1, \ldots, S_4)$

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant$$
$$2\int_{\mathcal{S}_3}\left[\sqrt{1 - \left(\sum_{c=1}^{k}\sqrt{P(c)\hat{P}(c)}e^{-\frac{1}{8}\left[\left(A(\pi_\triangle)-A\right)^T\Sigma^{-1}(\pi_\triangle)\left(A(\pi_\triangle)-A\right)\right]}\right)^2}\right] df(s_3|G),$$

where, as noted above, some of the dependencies on c are not explicitly accounted for. Here $f(s_3|G)$ *is not in general a Gaussian density, as* $\mathcal{S}_3$ *may, for example, involve discrete states.*

A way of further simplification of the bound in Corollary 8 is to bound the density upwards by the following quantity.

Definition 10 *Let*

$$\varepsilon(c) := \max_{s_1, s_3, s_4}\left|P(c)f(s_1 \times s_4|G) - \hat{P}(c)\hat{f}(s_1 \times s_4|G)\right|.$$

With this quantity we can simplify the computation of the bound in Theorem 8.

Theorem 11 $P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant \sum_{c=1}^{k}\varepsilon(c)\mu(s_1 \times s_4)$.

Proof From Corollary 8 (31) we have that $P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant$

$$\sum_{c=1}^{k}\int_{\mathcal{S}_3}\int_{\mathcal{S}_1\times\mathcal{S}_4}\left|P(c)f(s_1 \times s_4|G) - \hat{P}(c)\hat{f}(s_1 \times s_4|G)\right| d\mu(s_1 \times s_4) df(s_3|G).$$
$$\leqslant \sum_{c=1}^{k}\max_{s_3}\left[\int_{\mathcal{S}_1\times\mathcal{S}_4}\left|P(c)f(s_1 \times s_4|G) - \hat{P}(c)\hat{f}(s_1 \times s_4|G)\right| d\mu(s_1 \times s_4)\right].$$

We finish by using the definition of $\varepsilon(c)$, which yields

$$\leqslant \sum_{c=1}^{k}\varepsilon(c)\int_{\mathcal{S}_1\times\mathcal{S}_4} d\mu(s_1 \times s_4) = \sum_{c=1}^{k}\varepsilon(c)\mu(s_1 \times s_4).$$

■

In the next section, 7, we will be able to use the bound in Theorem 11 as a way of motivating the 'Naïve Bayes' approximation.

7. Bounding the Kolmogorov Distance of Variation with Respect to Naïve Bayes

In this section we assume that the feature space is finite and discrete, that is, $\mathcal{X} = \times_{i=1}^{d} \mathcal{X}_i$ and $\mathcal{X}_i = \{1, \ldots, r_i\}$. A popular plug-in classifier is the Naïve Bayes, already defined for three special cases in Examples 1, 4 and 13.

Definition 12 *A Naïve Bayes plug-in classifier is a classifier that assumes that the features of X are independent given c,*

$$\hat{f}(x|c) = \prod_{i=1}^{d} f(x_i|c).$$

In order not to overburden the notation we avoid symbols like $f_{X_i}(x_i|c)$ for marginal densities. In spite of this we are not restricted to the case where all marginal densities are identical.

There are several practical reasons for the popularity of the Naïve Bayes. For example, Toussaint (1972) has shown that if $\mathcal{X}_i$ is the same for every i, then $\hat{f}(x|c)$ is a polynomial.

There exist statistical tests for whether independence holds or not. In practice, however, we often exclude independence of features by domain knowledge.

When $c \in \{1,2\}$ (that is $k = 2$) and we have n samples $(x,c)^{(n)} = \{(x,c)_l\}_{l=1}^{n}$, we define $\hat{c}_{ERM}(x|x^{(n)})$ as the classifier that minimizes the empirical error on this sample. Without the Naïve Bayes assumption we can use bounds such as the following in (Devroye et al., 1996, Page 462) (for $k = 2$)

$$E\left[P\left(\widehat{c}_{ERM}\left(X|(X,C)^{(n)}\right) \neq C|(X,C)^{(n)}\right)\right] \leqslant P(\hat{c}_B(X) \neq c) + \varepsilon_1,$$

but with Naïve Bayes we have $0 \leqslant \varepsilon_2 \leqslant \varepsilon_1$ such that for $k = 2$ see Devroye et al. (1996, Chapter 27.3)

$$E\left[P\left(\widehat{c}_{ERM}\left(X|(X,C)^{(n)}\right) \neq C|(X,C)^{(n)}\right)\right] \leqslant P(\hat{c}_B(X) \neq c) + \varepsilon_2.$$

The Naïve Bayes assumption for specific data sets can actually perform better than a plug-in classifier incorporating some dependencies as shown in Titterington et al. (1981). In Friedman et al. (1997) Naïve Bayes has been reported as performing worse than taking dependence into account (but not on all data sets), and even then the difference was in many cases not large. In Huang et al. (2003) it is found as suboptimal in most data sets. A more in-depth Meta study on the subject is Hand and Yu (2001).

Our own experience does not speak against the conjecture that the advantage of taking dependence into account may depend on the context (Ekdahl, 2006, Section 7). Instead of running yet another simulation or arguing for or against on ad hoc grounds, we expand the existing theory around Naïve Bayes. The motivation for this can be intuitively outlined as follows.

Let us assume $P(C) = \hat{P}(C)$. Then the Kolmogorov variation distance is

$$\frac{1}{2} E_{\hat{f},\hat{P}} \left| \frac{f(X|C)}{\hat{f}(X|C)} - 1 \right|,$$

where now $\hat{f}(X|C)$ is as in Definition 12. As the level of dependence between the components in X increases, the dispersion of $\frac{f(X|C)}{\hat{f}(X|C)} - 1$ increases. In the case $d = 2$, or $X = (X_i)_{i=1}^{2}$ the Kolmogorov variation distance with respect to the product of marginal densities was first studied by Hoeffding (1942), who discovered that there is an upper bound for the distance, which is assumed when one

of X_1 or X_2 is a function of the other. This would seem to restrict the effectiveness of the Naïve Bayes classifier to those situations, where the degree of dependence between the components of X is moderate, as was to be expected. But to get a more diverse view we recall some of Vilmansen (1971) and Vilmansen (1973).

The Kolmogorov variation distance measuring the degree of association between X and C is in Vilmansen (1971, 1973) defined as

$$K(X,C) = \frac{1}{2}\sum_{c=1}^{k}\sum_{x\in\mathcal{X}} |f(x,c) - f(x)P(c)|.$$

The maximum of this distance is given by the next inequality, or

$$K(X,C) \leqslant 1 - \sum_{c=1}^{k} P^2(c). \tag{32}$$

This is shown in Vilmansen (1973), and is also found in Hoeffding (1942).

The maximum of $K(X,C)$ in (32) is obtained, as soon as there is a functional dependence between X and C in the sense that the supports of the class-conditional densities are disjoint subsets of $\mathcal{X}$. Hence, in the last mentioned case the probability of correct classification is one. This extreme case is approached when the dependence between X and C is "very close" to being functional, in the sense that each $f(x|c)$ is concentrated around its mode, say m_c, so that observation of m_c is an almost noise-free message, as it were, from the source c. In such a situation it should be possible for the Naïve Bayes classifier to perform well, too. The impact of strong dependence between the labels and feature vectors for Naïve Bayes has been argued in a different manner in Rish et al. (2001).

The following theorem shows in a more precise fashion how the modes of the densities $f(x|c)$ control the difference between the pertinent probabilities of the correct decision. We do not really need unimodality for our proofs, but this is a natural assumption for model based classification and simplifies the statements. In words the result in Theorem 13 below tells that, if the class conditional probability densities are predominantly well concentrated, the Kolmogorov variation distance with respect to Naïve Bayes is small.

Theorem 13 *Assume $f(x|c)$ is unimodal for every $c \in \mathcal{C}$. Let for any $c \in \mathcal{C}$,*

$$m_c := \arg\max_{x\in\mathcal{X}} f(x|c).$$

Then we have for the Naïve Bayes plug-in that

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant \sum_{c=1}^{k} P(c)\max\Big(f(m_c|c) - f(m_c|c)^d, 1 - f(m_c|c)\Big). \tag{33}$$

Proof We shall simplify notation without risk confusion by writing $f(x|c)$ as $f(x)$. We state first some elementary observations. By the chain rule

$$\begin{aligned} f(x) &= f(x_i)\, f(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n|x_i) \\ &\leqslant f(x_i), \text{for all } i, \end{aligned} \tag{34}$$

which implies that

$$f(x)^d = \prod_{i=1}^{d} f(x) \leqslant \prod_{i=1}^{d} f(x_i). \tag{35}$$

The claim to be established will follow if we can show that for all $x \in \mathcal{X}$

$$\left| f(x) - \prod_{i=1}^{d} f(x_i) \right| \leqslant \max\left(f(m) - f(m)^d, 1 - f(m) \right). \tag{36}$$

The proof of (36) is divided into three cases

1 Suppose that $x = m$.

1.1 If $f(m) \geqslant \prod_{i=1}^{d} f(m_i) \Rightarrow f(m) - \prod_{i=1}^{d} f(m_i) \leqslant f(m) - f(m)^d$ by (35).

1.2 If $f(m) < \prod_{i=1}^{d} f(m_i) \Rightarrow \prod_{i=1}^{d} f(m_i) - f(m) \leqslant 1 - f(m)$.

2 Consider next $x \neq m$. We have $\left| f(x) - \prod_{i=1}^{d} f(x_i) \right|$

$$= \max\left(f(x), \prod_{i=1}^{d} f(x_i) \right) - \min\left(f(x), \prod_{i=1}^{d} f(x_i) \right).$$

Since both $max(a_1, a_2)$ and $min(a_1, a_2)$ are positive for $0 \leq a_1, a_2 \leq 1$

$$\leqslant \max\left(f(x), \prod_{i=1}^{d} f(x_i) \right) \leqslant \max\left(f(x), f(x_j) \right),$$

where $f(x) \leqslant \sum_{z \neq m} f(z) = 1 - f(m)$. Here j is chosen so that $x_j \neq m_j$, which exists since $x \neq m$. By (34), $f(m_j) \geqslant f(m)$ we obtain

$$f(x_j) \leqslant \sum_{x_i \neq m_j} f(x_i) = 1 - f(m_j) \leqslant 1 - f(m).$$

The inequality (36) and Theorem 5 imply (33), and thus the assertion in Theorem 13 is established as claimed. ■

The inequality (36) is an improvement of a result in Rish et al. (2001). We have constructed a tighter upper bound for $\left| f(x) - \prod_{i=1}^{d} f(x_i) \right|$ than the one in Rish et al. (2001), which is recapitulated in the next theorem.

Theorem 14 *For all* $x \in \mathcal{X}$, $\left| f(x) - \prod_{i=1}^{d} f(x_i) \right| \leqslant d(1 - f(m))$.

The sharpness of Theorem 14 can be seen though a plot of $\max_{x \in \mathcal{X}} \left| f(x) - \prod_{i=1}^{d} f(x_i) \right|$ as a function of $\max_{x \in \mathcal{X}} f(x)$ in two and tree dimensions in Figure 4. The plots are for a binary hypercube ($\mathcal{X} = \{0,1\}^d$). The simulation tests several distributions and takes the worst one for each value of simulated $f(m)$ (the details of the simulation can be found in the simulation appendix).

It is possible to increase $f(m)$ dividing for example multimodal class conditional densities into many unimodal densities (Vilata and Rish, 2003), but a theoretical investigation of identification and interpretation of such a partition scheme is lacking.

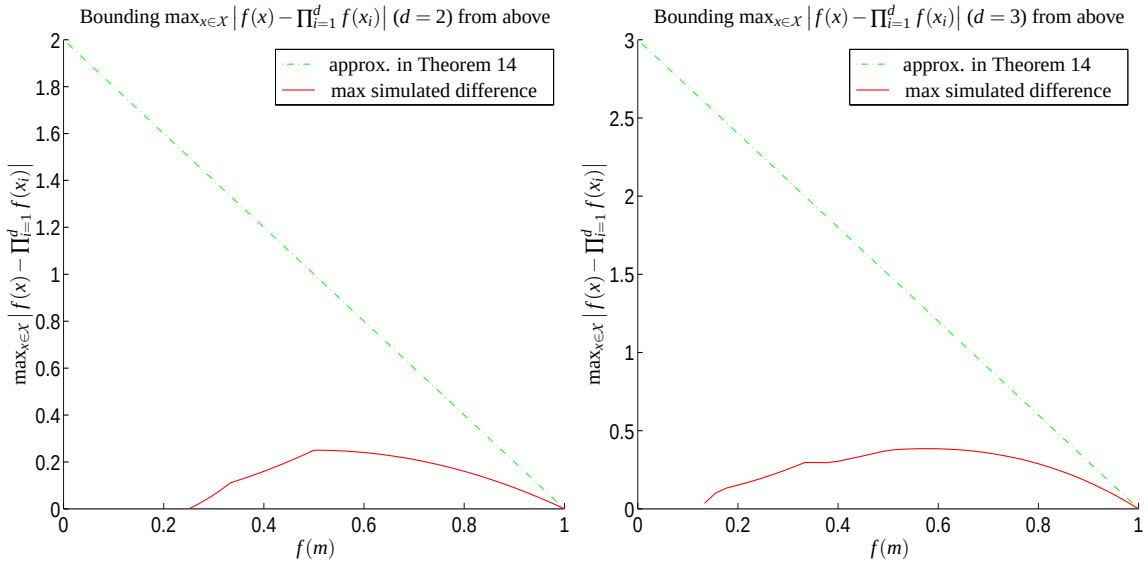


Figure 4: Illustration of the bound in Theorem 14.

Next we verify that the inequality (36) is in fact an improvement of Theorem 14, that is,

$$\max\Big(f(m) - f(m)^d, 1 - f(m)\Big) \leqslant d(1 - f(m)).$$

It is enough to show that $1 - f(m) \leqslant d(1 - f(m))$ **and** $f(m) - f(m)^d \leqslant d(1 - f(m))$. Here $1 - f(m) \leqslant d(1 - f(m))$ follow since $1 - f(m) \geqslant 0$ and $d \geqslant 2$. The remaining inequality can be shown using Bernoulli's inequality, so that

$$f(m) - f(m)^d = f(m) - (1 - (1 - f(m)))^d \leqslant f(m) - (1 - d\,(1 - f(m))) \leqslant d\,(1 - f(m)). \quad (37)$$

As with Theorem 14 we plot $\max_{x\in\mathcal{X}} \left| f(x) - \prod_{i=1}^{d} f(x_i) \right|$ as function of $f(m)$ in two and three dimensions and compare the maximal difference with the bounds in Theorem 14 and (36) (Figures 5 and 6).

From the three to five dimensional cases in Figures 5 and 6 we see that the inequality (36) is often sharp enough, if the probability density is concentrated, that is $f(m)$ is close to 1.

We give an additional upper bound ((39) below) readily derived from Theorem 13. We introduce the entropy (in natural logarithm)

$$H(X|C = c) = -\sum_{x\in\mathcal{X}} f(x|c) \ln f(x|c).$$

Then it is well known, see Arimoto (1971), that

$$1 - f(m_c|c) \leqslant \frac{H(X|C = c)}{\ln 2}. \quad (38)$$

Let us split $\mathcal{C}$ into two sets

$$\mathcal{C}_1 = \Big\{c \big| f(m_c|c) - f(m_c|c)^d \geqslant 1 - f(m_c|c)\Big\}$$

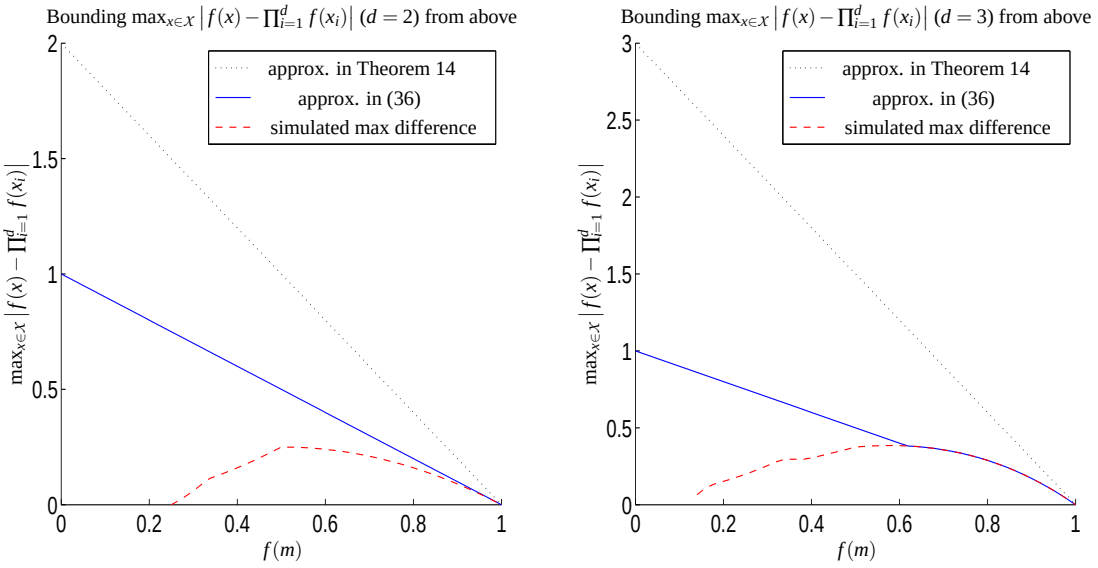


Figure 5: Illustration of the bounds in Theorem 14 and inequality (36) in two and three dimensions.

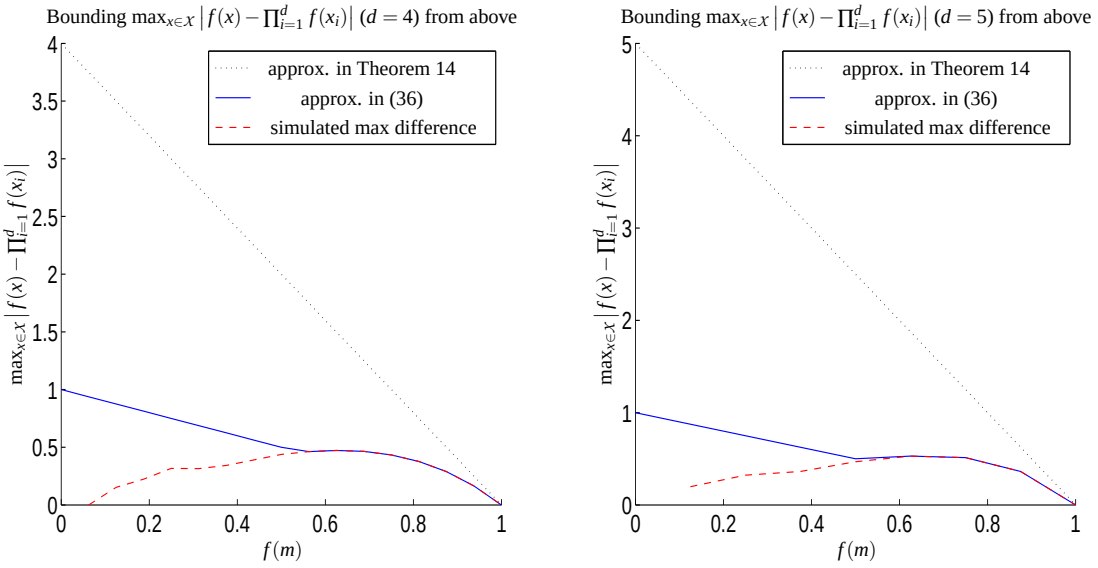


Figure 6: Illustration of the bounds in Theorem 14 and (36) in four and five dimensions.

and

$$\mathcal{C}_2 = \left\{ c | f(m_c|c) - f(m_c|c)^d < 1 - f(m_c|c) \right\}.$$

Thereby we get from (38)

$$\sum_{c=1}^{k} P(c) \max\left(f(m_c|c) - f(m_c|c)^d, 1 - f(m_c|c) \right)$$

$$\leqslant \sum_{\mathcal{C}_1} P(c) \left(f(m_c|c) - f(m_c|c)^d \right) + \sum_{\mathcal{C}_2} P(c) \frac{H(X|C=c)}{\ln 2}.$$

It was shown above, see (37), that $f(m_c|c) - f(m_c|c)^d \leqslant d\,(1 - f(m_c|c))$. We introduce the conditional entropy

$$H(X|C) = \sum_{c=1}^{k} P(c) H(X|C=c).$$

Then we obtain

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant (d-1) \left(1 - \sum_{\mathcal{C}_1} P(c) f(m_c|c) \right) + \frac{H(X|C)}{\ln 2}. \tag{39}$$

This bound needs not to be sharp in general, but it gives some additional insight. The first term in the right hand side is of similar form as the upper bound in (32) and can therefore be thought as measuring the degree of functional dependence between X and C. The entropy $H(X|C)$ is known as equivocation and measures the uncertainty about X if C is observed (Cover and Thomas, 1991).

Corollary 8 can be combined with the inequality (36). We will do that in the same way as in Section 6, that is we will allow for a partial Naïve Bayes assumption in the following sense

$$f(x|c) = f(s_2 \times s_3 \times s_4 | c, s_1) \prod_{\{i|X_i \in S_1\}} f(x_i|c)$$

leading to the abridged notation

$$m_1 \times m_4 := \arg \max_{s_1, s_3, s_4 \in \mathcal{S}_1 \times \mathcal{S}_3 \times \mathcal{S}_4} \prod_{\{i|X_i \in S_1 \cup S_4\}} f(x_i|\pi_i, c, G).$$

Corollary 15 *Let $P(c) = \hat{P}(c)$, and take the partial Naïve Bayes on s_1 and s_4. Then*

$$P(\hat{c}_B(X) = \varsigma) - P(\hat{c}(X) = \varsigma)$$

$$\leqslant \sum_{c=1}^{k} P(c) \max\left(f(m_1 \times m_4|G) - f(m_1 \times m_4|G)^d, 1 - f(m_1 \times m_4|G) \right) \prod_{\{i|X_i \in S_1 \cup S_4\}} r_i.$$

Example 14 *Let us take $k = 10$, and $d = 1000$. Most of the features are independent; however expert knowledge reveals that each class has exactly three features $(X_i, \ldots, X_{i+2})$ that depend on each other in accordance with the DAG in Figure 7. These features are very concentrated in the sense that there exists a vector a such that $P((X_i, \ldots, X_{i+2}) = a|c) > 0.995$. The expert explains that this it due to the fact that they correspond to a physical feature critical to each class. Which features have this dependence is, however, unknown and detection is complicated since*

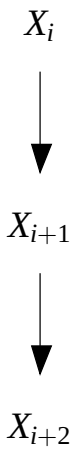


Figure 7: Graphical representation of dependence

$P(\text{there exists at least one } j \neq i \text{ such that} (X_j, \ldots, X_{j+2}|c) = a)$ *is large. Here it is possible to model with independence, since Corollary 15 then yields*

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) \leqslant \sum_{c=1}^{10} \varepsilon(c)\mu[\mathcal{S}_1 \cup \mathcal{S}_4] = 0.048.$$

8. Plug-In Classifiers that Make Optimal Decisions

In this section the plug-in classifier is not necessarily Naïve Bayes and $\mathcal{X}$ need not be discrete, although the conditions in the results below are more difficult to satisfy in continuous settings. As expounded above, it is easy to approximate $f(x|c)f(c)$, when the classes are well separated, in the sense there is almost a functional dependence between C and X. Sometimes it is possible to express the dependence by simply observing that

$$f(x|c)P(c) - f(x|\widetilde{c})P(\widetilde{c})$$

is large for all $x \in \mathcal{X}$ and all $c, \tilde{c} \in \mathcal{C}$ such that $c \neq \tilde{c}$. Here we present sufficient conditions for this pointwise separation between classes so that the probability of correct classification does not decrease by plugging in $\hat{f}(x|c)\hat{P}(c)$. The question is, how close must $\hat{f}(x|c)\hat{P}(c)$ be to $f(x|c)P(c)$, so that there should be no decrease in the probability of correct classification.

Definition 16 *Let $\varepsilon_2(c)$ be any bound such that for all $x \in \mathcal{X}$*

$$\left| f(x|c)P(c) - \hat{f}(x|c)\hat{P}(c) \right| \leqslant \varepsilon_2(c). \tag{40}$$

For example, let $\mathcal{X}$ be discrete and finite. If $\varepsilon_2(x,c) = f(x|c)P(c) - \hat{f}(x|c)\hat{P}(c)$, then we obviously take $\varepsilon_2(c) = \max_{x \in \mathcal{X}} \mid \varepsilon_2(x,c) \mid$. Let us suppose that the approximation is a lower bound, that is, $f(x|c)P(c) > \hat{f}(x|c)\hat{P}(c)$ for all $x \in \mathcal{X}$, which is found by variation and normalization (Jordan

et al., 1999; Wainwright and Jordan, 2003). These techniques give even expressions for $\varepsilon_2(x,c)$, for example for the exponential family of densities. We continue, however, with a lemma that involves this kind of differences in general.

Lemma 17 *Assume that* $\varepsilon_2(c) > 0$ *and* $\varepsilon_2(\widetilde{c}) > 0$ *exist as defined in (40). If* $P(c|x) > P(\widetilde{c}|x)$ *and*

$$|f(x|c)P(c) - f(x|\widetilde{c})P(\widetilde{c})| \geqslant \varepsilon_2(c) + \varepsilon_2(\widetilde{c})$$

then $\hat{P}(c|x) \geqslant \hat{P}(\widetilde{c}|x)$.

Proof We prove this by contradiction. First we assume that $\hat{P}(c|x) < \hat{P}(\widetilde{c}|x)$. With the plug-in classifier and (2) we get

$$\hat{f}(x|c)\hat{P}(c) < \hat{f}(x|\widetilde{c})\hat{P}(\widetilde{c}).$$

Now we continue by applying (40), that is, increasing margin in this inequality, which gives

$$\Rightarrow f(x|c)P(c) - \varepsilon_2(c) < f(x|\widetilde{c})P(\widetilde{c}) + \varepsilon_2(\widetilde{c})$$

$$\Leftrightarrow f(x|c)P(c) - f(x|\widetilde{c})P(\widetilde{c}) < \varepsilon_2(c) + \varepsilon_2(\widetilde{c}).$$

By the assumption $P(c|x) > P(\widetilde{c}|x)$ and by (1) we get that the quantity in the left hand side of the last inequality is positive, and hence the desired contradiction follows.

■

Lemma 17 used to state sufficient conditions such that $f(x|c)$ can be approximated without affecting the probability of correct classification.

Theorem 18 *If for all* $c, \widetilde{c} \in \mathcal{C}$

$$|f(x|c)P(c) - f(x|\widetilde{c})P(\widetilde{c})| \geqslant \varepsilon_2(c) + \varepsilon_2(\widetilde{c}),$$

then $P(\hat{c}_B(X) = C) = P(\hat{c}_{\hat{B}}(X) = C)$.

Proof From (13) we have that

$$P(\hat{c}_B(X) = C) - P(\hat{c}_{\hat{B}}(X) = C) = \int_{\{x|\hat{c}_{\hat{B}}(x) \neq c\}} (P(c)f(x|c) - P(\hat{c})f(x|\hat{c}))\, d\mu(x).$$

Now the result follows since Lemma 17 implies (through (1)) that $P(c|x) = P(\hat{c}|x)$. ■

We can also combine Theorem 18 with inequality (36). This gives us next corollary.

Corollary 19 *When* $\varepsilon_2(c) = \max\left(f(m|c) - f(m|c)^d, 1 - f(m|c)\right)$ *and* $|P(c|x)P(c) - P(\widetilde{c}|x)P(\widetilde{c})| \geqslant \varepsilon_2(c)P(c) + \varepsilon_2(\widetilde{c})P(\widetilde{c})$, *then*

$$P(\hat{c}_{\hat{B}}(X) = C) = P(\hat{c}_B(x) = C).$$

In the context of Naïve Bayes our Corollary 19 can be seen as a generalization of the results in Domingos and Pazzani (1997), the finding of which is that Naïve Bayes is optimal for learning conjunctions and disjunctions of literals, as well as an extension of the more general result in Rish et al. (2001), which says that Naïve Bayes is optimal if Bayes classifier assigns only one feature to class 1 in a two-class problem. For example Corollary 19 is more general in the sense that a classifier that assigns more than one feature to a class can be optimal if the margin is wide enough.

9. Summary

We have presented exact and easily computable bounds for the degradation of probability of correct classification when Bayes classifiers are used with respect to partial plug-in conditional densities in a Bayesian network model (Theorem 7, Corollary 8 and Theorem 11).

An example of a Bayesian network plug-in classifier is the Naïve Bayes classifier (Definition 12). In the case where a Naïve Bayesian classifier is used, we have sharpened a bound of evaluating its effect (Theorem 13).

We have presented a bound on the class conditional approximation as well as the class probabilities such that the probability of making a correct decision is not degraded when basing the decision on $\hat{c}_{\hat{B}}(x)$ instead of $\hat{c}_B(x)$ using the bound in Theorem 13, thus generalizing the theory for explaining when Naïve Bayes is optimal.

Acknowledgments

A preliminary version of the results was reported at The 23rd Annual Workshop of the Swedish Artificial Intelligence Society (Ekdahl and Koski, 2006). The research was partially supported by Swedish Research Council, grant 4042/401 and EU 6th Programme grant QLK3-CT-2002-02097 (BACDRIVERS).

Appendix A. Simulation

Algorithm 1 Simulate($granularity, d$)

$atom = \frac{1}{granularity}$
$state$ is a placement of atoms on $\mathcal{X}$ such that for a state x, $f(x) = \frac{nr\ atoms\ there}{granularity}$
while not all unique placements of atoms have been searched **do**
  $p = \max_{x \in \mathcal{X}} f(x)$
  $diff = \max_{x \in \mathcal{X}} \left| f(x) - \prod_{i=1}^{d} f(x_i) \right|$
  if $diff > maxdiffs[p]$ **then**
    $maxdiffs[p] = diff$
  end if
  $state =$ next unique placement of atoms
end while
return $maxdiffs$

References

S.M. Ali and S.D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B*, 28(1):131–142, 1966.

S. Anoulova, P. Fischer, S. Polt, and H.U. Simon. Probably almost Bayes decisions. *Information and Computation*, 129(1):63–71, 1996.

S. Arimoto. Information-theoretical considerations on estimation problems. *Information and Control*, 19:181–194, 1971.

R.R. Bahadur. On classification based on responses to *n* dichotomous items. In H. Solomon, editor, *Studies in Item Analysis and Prediction*, pages 169–177, Stanford, California, 1961a. Stanford University Press.

R.R. Bahadur. A representation of the joint distribution of responses to *n* dichotomous items. In H. Solomon, editor, *Studies in Item Analysis and Prediction*, pages 158–168, Stanford, California, 1961b. Stanford University Press.

L.C. Barbosa. Maximum likelihood sequence estimators: A geometric view. *IEEE Transactions on Information Theory*, 35(2), 1989.

B.K. Bhattacharyya and G.T. Toussaint. An upper bound on the probability of misclassification in terms of Matusita's measure of affinity. *Annals of the Institute of Statistical Mathematics*, 34: 161–165, 1982.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Uncertainty in Artificial Intelligence (UAI-96)*, pages 115–123, 1996.

D.T. Brown. A note on approximations to discrete probability distributions. *Information and Control*, 2:386–392, 1959.

H.D. Brunk and D.A. Pierce. Estimation of discrete multivariate densities for computer-aided differential diagnosis. *Biometrika*, 61(3):493–499, 1974.

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

C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependency trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

G. Cooper. The computational complexity of probabilistic inference using bayesian Belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

T.M. Cover and J.A Thomas. *Elements of Information Theory*, chapter 2.2, page 169. Wiley, 1991.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

H. Cramér. *Mathematical Methods of Statistics, 11 th printing*. Princeton University Press, 1966.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2):103–130, 1997.

M. Ekdahl. *Approximations of Bayes Classifiers for Statistical Learning of Clusters*. Licentiate thesis, Linköpings Universitet, 2006.

M. Ekdahl and T. Koski. On the performance of model based approximations in classification. In *Proceedings of The 23rd Annual Workshop of the Swedish Artificial Intelligence Society*, pages 73–82. http://sais2006.cs.umu.se/, 2006.

J.H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29 (2):1–36, 1997.

N. Glick. Sample-based classification procedures derived from density estimators. *Journal of the American Statistical Association*, 67(337):116–122, 1972.

N. Glick. Sample based multinomial classification. *Biometrics*, 29(2):241–256, 1973.

L.A. Goodman and W.H. Kruskal. Measures of association for cross classifications. *Journal of the American Statistical Association*, 49(268):732–763, 1954.

M. Gyllenberg and T. Koski. Probabilistic models for bacterial taxonomy. *International Statistical Review*, 69:249–276, 2001.

D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2001.

D.J. Hand and K. Yu. Idiot's Bayes–not so stupid after all? *International Statistical Review*, 69(3): 385–398, 2001.

W. Hoeffding. Stochastische unabhängigkeit und funktionaler zusammenhang. *Skandinavisk Aktuarietidskrift*, 25:200–227, 1942.

W. Hoeffding and J. Wolfowitz. Distinguishability of sets of distributions. *The Annals of Mathematical Statistics*, 29(3):700–718, 1958.

K. Huang, I. King, and M.R. Lyu. Finite mixture model of bounded semi-naive Bayesian network classifier. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-2003)*, volume 2714 of *Lecture Notes in Computer Science*. Springer, 2003.

M.I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications Technology*, 15(1):52–60, 1967.

G.K. Kaleh. Channel equalization for block transmission systems. *IEEE Journal on Selected Areas in Communications*, 13(1):150–121, 1995.

K.B. Korb and A.E. Nicholson. *Bayesian Artificial Intelligence*. Chapman and Hall, 2004.

H.H. Ku and S. Kullback. Approximating discrete probability distributions. *IEEE Transactions on Information Theory*, 15:444–447, 1969.

S. Kullback. *Information Theory and Statistics*, chapter 9, page 190. Dover Publications Inc., 1997.

S.L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108, 1990.

P.M. Lewis. Approximating probability distributions to reduce storage requirements. *Information and Control*, 2:214–225, 1959.

D.K. McGraw and J.F. Wagner. Elliptically symmetric distributions. *IEEE Transactions on Information Theory*, 14(1):110 – 120, 1968.

D.H.II. Moore. Evaluation of five discrimination procedures for binary variables. *Journal of the American Statistical Association*, 68(342):399–404, 1973.

X. Nguyen, M.J. Wainwright, and M.I. Jordan. On divergences, surrogate loss functions, and decentralized detection. Technical Report 695, University of California, Berkeley, 2005.

J. Ott and R.A. Kronmal. Some classification procedures for multivariate binary data using orthogonal functions. *Journal of the American Statistical Association*, 71(354):391–399, 1976.

G.A. Pistone, E.A. Riccomagno, and H.P.A. Wynn. Gröbner bases and factorisation in discrete probability and Bayes. *Statistics and Computing*, 11(1):37–46, 2001.

E.J.G. Pitman. *Some Basic Theory for Statistical Inference*, chapter 2. Chapman and Hall, 1979.

B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

I. Rish, J. Hellerstein, and J. Thathachar. An analysis of data characteristics that affect Naive Bayes performance. Technical Report RC21993, IBM, 2001.

J.Van Ryzin. Bayes risk consistency of classification procedures using density estimation. *Sankhya Series A*, 28:261–270, 1966.

P. Scheinck. Symptom diagnosis Bayes's and Bahadur's distribution. *International Journal of Biomedical Computing*, 3:17–28, 1972.

M.J. Schervish. *Theory of Statistics*. Springer, second edition, 1995.

D. Slepian. On the symmetrized Kronecker power of a matrix and extensions of Mehler's formula for Hermite polynomials. *SIAM Journal on Mathematical Analysis*, 3:606–616, 1972.

H. Strasser. *Mathematical Theory of Statistics*, chapter 2.2. Walter de Gruyter, 1985.

J.L. Teugels. Some representations of the multivariate Bernoulli and binomial distributions. *Journal of Multivariate Analysis*, 33(1):256–268, 1990.

D.M. Titterington, G.D. Murray, L.S. Murray, D.J. Spiegelhalter, A.M. Skene, J.D.F. Habbema, and G.J. Gelpke. Comparison of discrimination techniques applied to a complex data set of head injured patients. *Journal of the Royal Statistical Society.*, 144(2):145–175, 1981.

F. Topsoe. Some inequalities for information divergence and related measures of discrimination. *IEEE Transactions on Information Theory*, 46(4):1602–1609, 2000.

G.T. Toussaint. Polynomial representation of classifiers with independent discrete-valued features. *IEEE Transactions on Computers*, 21:205–208, 1972.

R. van Engelen. Approximating bayesian Belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intellignce*, 19(8):916–920, 1997.

R. Vilata and I. Rish. A decomposition of classes via clustering to explain and improve Naive Bayes. In *Machine Learning: ECML 2003: 14th European Conference on Machine Learning*, pages 444 – 455, 2003.

T.R. Vilmansen. Feature evaluation with measures of probabilistic dependence. *IEEE Transactions on Computers*, 22(4):381–387, 1973.

T.R. Vilmansen. On dependence and discrimination in pattern recognition. *IEEE Transactions on Computers*, 21:1029–1031, 1971.

M.J Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Dept. of Statistics, 2003.

J. Williamson. *Bayesian Nets and Causality: Philosophical and Computational Foundations*. Oxford University Press, 2005.

Journal of Machine Learning Research 7 (2006) 2481-2514 Submitted 1/06; Revised 5/06; Published 11/06

Estimation of Gradients and Coordinate Covariation in Classification

Sayan Mukherjee SAYAN@STAT.DUKE.EDU
Qiang Wu QIANG@STAT.DUKE.EDU
Institute for Genome Sciences & Policy
Institute of Statistics and Decision Sciences
Department of Computer Science
Duke University
Durham, NC 27708, USA

Editor: Isabelle Guyon

Abstract

We introduce an algorithm that simultaneously estimates a classification function as well as its gradient in the supervised learning framework. The motivation for the algorithm is to find salient variables and estimate how they covary. An efficient implementation with respect to both memory and time is given. The utility of the algorithm is illustrated on simulated data as well as a gene expression data set. An error analysis is given for the convergence of the estimate of the classification function and its gradient to the true classification function and true gradient.

Keywords: Tikhnonov regularization, variable selection, reproducing kernel Hilbert space, generalization bounds, classification

1. Introduction

The advent of data sets with many variables or coordinates in the biological and physical sciences has driven the use of a variety of machine learning approaches based on Tikhonov regularization (global shrinkage estimators in the statistics literature) such as support vector machines (SVMs) (Vapnik, 1998) and regularized least square classification (Poggio and Girosi, 1990). These algorithms have been very successful in classification (binary regression) problems.

In a number of applications, such as the analysis of gene expression data, classical questions from statistical modeling of which variables are of relevance and how these variables interact arise. In the context of genomic data an objective of the analysis is to build an interpretable model of the biological process giving rise to the data. An example of this is that genes co-regulated by a biological pathway may be modeled as features that covary. Estimation of feature covariation is not considered in standard regression or classification methods that allow for variable selection: recursive feature elimination (RFE) (Guyon et al., 2002), least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996), and basis pursuits denoising (Chen et al., 1999). Gradient information was used in Hermes and Buhmann (2000) and Evgeniou et al. (2000a) to select features via a sensitivity analysis on the gradient of the SVM solution. This approach does not directly estimate the gradient and its shortcomings will be described in Remark 3. Statistical models based on shrinkage or regularization were applied to the problem of learning coordinate covariation and relevance for regression problems in Mukherjee and Zhou (2006). We extend this approach to the

binary regression or classification setting by simultaneously estimating the classification function as well as its gradient.

1.1 Review on Convex Risk Minimization Approach for Classification

In this subsection we review the convex risk minimization approach.

Let X be a compact metric space and $Y = \{1,-1\}$. Let $\rho(x,y)$ be a probability distribution on $Z := X \times Y$ and $\mathbf{z} = \{(x_i,y_i)\}_{i=1}^m \in Z^m$ a random sample independently drawn according to $\rho(x,y)$.

Convex risk minimization methods, which include support vector machines (SVMs) and boosting as typical examples, have been successful in a variety of classification problems. This approach involves a convex loss function ϕ and learns a real-valued classification function from a given sample $\mathbf{z} = \{(x_i,y_i)\}_{i=1}^m$ by minimizing the convex empirical risk functional in a hypothesis space $\mathcal{H}$ (often with a regularization or penalty term):

$$f_{\mathbf{z}} = \arg\min_{f\in\mathcal{H}} \Big\{ \frac{1}{m}\sum_{i=1}^m \phi(y_i f(x_i)) \Big\}. \tag{1}$$

The loss function may take the form of the hinge loss $\phi(t) = (1-t)_+$ in SVMs and logistic loss $\phi(t) = \log(1+e^{-t})$ in boosting. Define the expected error of a function f as

$$\mathcal{R}(f) = \int \phi(yf(x))\, d\rho(x,y),$$

and the real-valued classification function as the function in $L^2_{\rho_X}$, where ρ_X is the marginal distribution on x, that minimizes

$$f_\phi = \arg\min_{f\in L^2_{\rho_X}} \mathcal{R}(f).$$

Under certain conditions (Vapnik, 1998; Bartlett et al., 2005) $\mathrm{sgn}[f_\phi]$ is a Bayes optimal classifier. Extensive investigation in learning theory (Cortes and Vapnik, 1995; Vapnik, 1998; Evgeniou et al., 2000b; Schoelkopf and Smola, 2001; Zhang, 2004; Bartlett et al., 2005; Wu and Zhou, 2005) has shown that $\mathcal{R}(f_{\mathbf{z}}) \to \mathcal{R}(f_\phi)$, which implies that the error of $\mathrm{sgn}(f_{\mathbf{z}})$ converges to the error of a Bayes optimal classifier with respect to the misclassification error:

$$\mathcal{C}(\mathrm{sgn}(f)) = \mathrm{Prob}\{\mathrm{sgn}(f(x)) \neq y\}.$$

This forms the theoretical foundation of the convex risk minimization method.

1.2 Learning the Classification Function and Gradient

In this paper we are interested in simultaneously learning f_ϕ and its gradient from the sample values, $\mathbf{z} = \{(x_i,y_i)\}_{i=1}^m$. Denote $x = (x^1,x^2,\ldots,x^n)^T \in \mathbb{R}^n$. The gradient of f_ϕ is the vector of functions (if the partial derivatives exist)

$$\nabla f_\phi = \left(\frac{\partial f_\phi}{\partial x^1}, \ldots, \frac{\partial f_\phi}{\partial x^n} \right)^T.$$

Note that gradient learning is meaningful for classification problems in this sense because f_ϕ is real-valued and may be smooth. For example, in the case of the logistic function (Hastie et al., 2001)

$$\phi(yf(x)) = \log\big(1+e^{-yf(x)}\big).$$

the classification function has a clear statistical interpretation (modeling the conditional probability $\rho(y|X)$ as a Bernoulli random variable)

$$\mathbf{P}\text{rob}(y = \pm 1|x) = \frac{1}{1+e^{-y f_\phi(x)}}.$$

In this case the classification function is

$$f_\phi(x) = \ln\left[\frac{\mathbf{P}\text{rob}(y=1|x)}{\mathbf{P}\text{rob}(y=-1|x)}\right]$$

and the gradient of f_ϕ exists under very mild conditions on the underlying distribution ρ. This is one of the reasons we use a logistic model rather than learning the gradient of a $\{-1,1\}$ classification function. In addition, the logistic model incorporates the uncertainty of the conditional probability at each x which the binary classification function does not.

The relevance of learning the gradient with respect to the problems of variable selection and estimating coordinate covariation is that the gradient provides the following information:
(a) variable selection: the norm of a partial derivative $\|\frac{\partial f_\phi}{\partial x^j}\|_{L^2_{\rho_X}}$ indicates the relevance of this variable, since a small norm implies a small change in the discriminative function f_ϕ with respect to the j-th coordinate,
(b) coordinate covariation: the inner product between partial derivatives $\left\langle \frac{\partial f_\phi}{\partial x^j}, \frac{\partial f_\phi}{\partial x^\ell} \right\rangle$ indicates the covariance of the j-th and ℓ-th coordinates with respect to variation in f_ϕ.

At first glance, the problem of estimating the gradient is equivalent to that of computing classical numerical derivatives in inverse problems. This is the case if we know the sample pair $\{(x_i, f_\phi(x_i)\}_{i=1}^m$. But we face the difficulty that what we have in hand is the set of samples $\mathbf{z}$ where $y_i \in \{\pm 1\}$ is not an approximation of the value $f_\phi(x_i)$ but only its sign. So the classical methods for numerical derivatives fail for learning gradients in the classification setting. Instead, we will motivate a new approach.

The derivation of our gradient learning algorithm can be motivated by the Taylor expansion of f_ϕ, assuming it exists:

$$f_\phi(x) \approx f_\phi(u) + \nabla f_\phi(x) \cdot (x-u), \qquad \text{for } x \approx u.$$

Our objective will be to estimate f_ϕ by a function g and its gradient ∇f_ϕ by a vector valued function $\vec{f} = (f_1, f_2, \ldots, f_n)^T : X \to \mathbb{R}^n$. If the estimates are accurate then the following should hold

$$f_\phi(x) \approx g(u) + \vec{f}(x) \cdot (x-u), \qquad \text{for } x \approx u.$$

The optimization given in (1) suggests a method for estimating g and $\vec{f}$: we minimize a quantity that is like the convex empirical risk but with $f(x_i)$ replaced by $g(u) + \vec{f}(x_i) \cdot (x_i - u)$ with some $u \approx x_i$. As for the choice of u, a natural idea is to set $u = x_j$ and take a weighted average with the weights being chosen to enforce the locality constraints $x_j \approx x_i$ implicit in the Taylor expansion. Various weights may play the same role whenever they satisfies $w_{i,j} \to 0$ as $|x_i - x_j| \to 0$. Throughout this paper we will use a Gaussian with variance s as our weight function:[1]

$$w_{i,j} = w_{i,j}^{(s)} = \frac{1}{s^{n+2}} e^{-\frac{|x_i - x_j|^2}{2s^2}} = w(x_i - x_j), \qquad i,j = 1,\ldots,m.$$

1. In standard problems such as density estimation the Gaussian is normalized by a term of the form $\frac{1}{s^n}$ since the following integral should be invariant with respect to dimension

$$\frac{1}{s^n}\int_{\mathbb{R}^n} e^{-|x|^2/s^2} dx = \text{constant}.$$

Other weight functions can be used as long as the bandwidth of the weight function decreases with the number of samples. Using the Gaussian weight function leads to the following empirical error functional.

Definition 1 *Given a sample $\mathbf{z} \in Z^m$, a function $g : X \to \mathbb{R}$, and a vector-valued function $\vec{f} : X \to \mathbb{R}^n$, we define the empirical error as follows:*

$$\mathcal{E}_{\mathbf{z}}(g, \vec{f}) = \frac{1}{m^2} \sum_{i,j=1}^{m} w_{i,j}^{(s)} \phi\big(y_i(g(x_j) + \vec{f}(x_i) \cdot (x_i - x_j))\big).$$

We may expect that minimizing this error functional using functions in a hypothesis space $\mathcal{H}^{n+1}$ leads to $g_{\mathbf{z}}$ and $\vec{f}_{\mathbf{z}}$ such that

$$g_{\mathbf{z}}(u) + \vec{f}_{\mathbf{z}}(x) \cdot (x-u) \approx f_{\mathbf{z}}(x) \approx f_\phi(x) \approx f_\phi(u) + \nabla f_\phi(x) \cdot (x-u), \qquad \text{for } x \approx u.$$

This in general leads to $g_{\mathbf{z}} \approx f_\phi$ and $\vec{f}_{\mathbf{z}} \approx \nabla f_\phi$.

To formulate the algorithm, we need to specify the hypothesis space. In this paper we will restrict $\mathcal{H}$ to be a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_K$ with an associated Mercer kernel $K : X \times X \to \mathbb{R}$ that is continuous, symmetric and positive semidefinite. The RKHS is defined (Aronszajn, 1950) to be the completion of the linear span of the set of functions $\{K_x := K(\cdot, x) : x \in X\}$ with the inner product $\langle \cdot, \cdot \rangle_K$ satisfying $\langle K_u, K_v \rangle_K = K(u,v)$. The reproducing property of $\mathcal{H}_K$ is

$$\langle K_x, f \rangle_K = f(x), \qquad \forall x \in X, f \in \mathcal{H}_K. \tag{2}$$

This implies that every function $f \in \mathcal{H}_K$ is continuous and bounded. Hence $\mathcal{H}_K \subset C(X) \subset L^2_{\rho_X}(X)$.

Regularizing or shrinking the empirical error $\mathcal{E}_{\mathbf{z}}(g, \vec{f})$ with respect to the RKHS norm defines the following optimization problem.

Definition 2 *Given a sample $\mathbf{z} \in Z^m$ we can estimate the classification function, $g_{\mathbf{z}}$, and its gradient, $\vec{f}_{\mathbf{z}}$, as follows:*

$$(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}}) = \underset{(g,\vec{f}) \in \mathcal{H}_K^{n+1}}{\arg\min} \left\{ \mathcal{E}_{\mathbf{z}}(g, \vec{f}) + \frac{\lambda}{2}(\|g\|_K^2 + \|\vec{f}\|_K^2) \right\}, \tag{3}$$

where $s, \lambda > 0$ are the regularization parameters and, for $\vec{f} = (f^1, \ldots, f^n) \in \mathcal{H}_K^n$, $\|\vec{f}\|_K^2 = \sum_{i=1}^n \|f^i\|_K^2$.

The immediate advantages of this technique are preventing overfitting and easy computability due to a representer theorem (see Section 2). Another advantage of our method is the derived functions are already approximations of the partial derivatives and they have RKHS inner products which are computed in the estimation process. The inner products reflect the nature of the measure, which is often on a low dimensional manifold embedded in a high dimensional space.

In our paper for technical reasons that will become apparent in the proofs the following quantity should be invariant with respect to dimension

$$\frac{1}{s^{n+2}} \int_{\mathbb{R}^n} e^{-|x|^2/s^2} |x|^2 dx = \text{constant}.$$

Remark 3 *One may consider a natural approach of finding an approximation of f_ϕ (for example by* (1)*) and then computing partial derivatives. But recall our aim is feature (or variable) selection. The problem with this approach is that the partial derivatives may no longer be in the RKHS of the classification function. This leaves us with the problem of not having a norm or computable metric to work with.*

The hypothesis space $\mathcal{H}_K^n$ in the optimization problem (3) may be replaced by some other space of vector-valued functions (Micchelli and Pontil, 2005) in order to learn the gradients.

The distance between points in the Taylor expansion as well as in the weighting function are in the input space and not the feature space of the kernel. This is a natural formulation and an argument for this formulation is that with this distance the algorithms can be extended to a manifold setting without any changes (Mukherjee et al., 2006).

1.3 Overview

In Section 2, we show that the minimizer of the optimization problem (3) satisfies a representer theorem and then provide a procedure to compute the parameters. In Section 3, we prove the convergence of our estimate of the gradient, $\vec{f}_{\mathbf{z}}$, to the true gradient of the classification function, ∇f_ϕ. The utility of the algorithm is illustrated in Section 4 on simulated data as well as gene expression data. We close with a brief discussion in Section 5.

2. Representer Theorem and Parameter Computation

The optimization problem defined by Equation (3) is a convex optimization problem because $\phi(\cdot)$, $\|g\|_K^2$, and $\|\vec{f}\|_K^2$ are all convex functionals. Denote $\mathbb{R}^{p\times q}$ as the space of $p \times q$ matrices. The algorithm that implements the optimization procedure is given in Section 2.1.

The following theorem is an analog of the standard representer theorem (Wahba, 1990; Schoelkopf and Smola, 2001) that states the minimizer of the optimization problem defined by Equation (3) has a finite dimensional representation.

Proposition 4 *Given a sample $\mathbf{z} \in Z^m$ the solution of (3) exists and takes the form*

$$g_{\mathbf{z}}(x) = \sum_{i=1}^m \alpha_{i,\mathbf{z}} K(x,x_i) \quad \textit{and} \quad \vec{f}_{\mathbf{z}}(x) = \sum_{i=1}^m c_{i,\mathbf{z}} K(x,x_i) \tag{4}$$

with $c_{\mathbf{z}} = (c_{1,\mathbf{z}},\ldots,c_{m,\mathbf{z}}) \in \mathbb{R}^{n\times m}$ and $\alpha_{\mathbf{z}} = (\alpha_{1,\mathbf{z}},...,\alpha_{m,\mathbf{z}})^T \in \mathbb{R}^m$.

Proof The existence follows from the convexity of ϕ and functionals $\|g\|_K^2$ and $\|\vec{f}\|_K^2$. Suppose $(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}})$ is a minimizer. We can write functions $g_{\mathbf{z}} \in \mathcal{H}_K$ and $\vec{f}_{\mathbf{z}} \in \mathcal{H}_K^n$ as

$$g_{\mathbf{z}} = g_\| + g_\perp \ \text{ and } \ \vec{f}_{\mathbf{z}} = \vec{f}_\| + \vec{f}_\perp$$

where $g_\|$ and each element of $\vec{f}_\|$ is in the span of $\{K_{x_1},,...,K_{x_m}\}$, and $g_\perp$ and $\vec{f}_\perp$ are functions in the orthogonal complement. By the reproducing property $g_{\mathbf{z}}(x_i) = g_\|(x_i)$ and $\vec{f}(x_i) = \vec{f}_\|(x_i)$ for all x_i. So the functions $g_\perp$ and $\vec{f}_\perp$ do not have an effect on $\mathcal{E}_{\mathbf{z}}(g,\vec{f})$. But $\|g_{\mathbf{z}}\|_K^2 = \|g_\| + g_\perp\|_K^2 > \|g_\|\|_K^2$ and $\|\vec{f}_{\mathbf{z}}\|_K^2 = \|\vec{f}_\| + \vec{f}_\perp\|_K^2 > \|\vec{f}_\|\|_K^2$ unless $g_\perp, \vec{f}_\perp = 0$. This implies that $g_{\mathbf{z}} = g_\|$ and $\vec{f}_{\mathbf{z}} = \vec{f}_\|$. This results in the representations in Equation (4). ■

The optimization in Equation (3) can be written in terms of the coefficients $c_{\mathbf{z}}$ and $\alpha_{\mathbf{z}}$. We define a matrix $C = (c_1, c_2, \ldots, c_m) \in \mathbb{R}^{n\times m}$ (when optimized these will be the coefficients $c_{\mathbf{z}}$ in the gradient expansion) and the vector $\alpha \in \mathbb{R}^m$ (when optimized the vector will be $\alpha_{\mathbf{z}}$). We denote the kernel matrix K where $K_{ij} = K(x_i, x_j)$ for $i, j = 1, ..., m$ and the i-th row of the matrix as k_i. The optimization function can be written in matrix form as

$$\Phi(C,\alpha) = \frac{1}{m^2}\sum_{i,j=1}^{m} w_{i,j}\phi\big(y_i(k_j\alpha + k_iC^T(x_i - x_j))\big) + \frac{\lambda}{2}\big(\alpha^T K\alpha + \mathrm{Tr}(CKC^T)\big), \tag{5}$$

where $\mathrm{Tr}(M)$ is the trace of a matrix M.

Proposition 5 *If ϕ is differentiable, then the coefficients $c_{\mathbf{z}}$ and $\alpha_{\mathbf{z}}$ can be computed from the equation $\nabla\Phi(\alpha, C) = 0$.*

We can optimize (5) by using Newton's method to solve $\nabla\Phi(\alpha, C) = 0$. The matrix C however is an $n \times m$ matrix and optimizing in $\mathbb{R}^{mn}$ is problematic for applications where $n \gg m$. We will show that the coefficients can be computed by the optimization of an $m \times d$ matrix, where typically $d \ll m$. We will then apply Newton's method in this reduced space.

Define a vector-valued function

$$h = ((h^0)^T, (h_1)^T, \ldots, (h_m)^T)^T : \mathbb{R}^{(n+1)m} \to \mathbb{R}^{(n+1)m}$$

with

$$h^0 = (h_1^0, \ldots, h_m^0)^T, \quad h_j^0(\alpha, C) = \frac{1}{m^2}\sum_{i=1}^{m} w_{i,j}\phi'\big(y_i(k_j\alpha + k_iC^T(x_j - x_i))\big)y_i + \lambda\alpha_j$$

and, for $i = 1, \ldots, m$,

$$h_i(\alpha, C) = \frac{1}{m^2}\sum_{j=1}^{m} w_{i,j}\phi'\big(y_i(k_j\alpha + k_iC^T(x_j - x_i))\big)y_i(x_i - x_j) + \lambda c_i.$$

By direct computation, we have

$$\nabla\Phi(\alpha, C) = \begin{pmatrix} K & 0 \\ 0 & K \otimes I_n \end{pmatrix} h(\alpha, C) \tag{6}$$

where I_n is the $n \times n$ identity matrix. Solving for the coefficients will give us the following proposition.

Proposition 6 *If the solution to the equation $h(\alpha, C) = 0$ exists, then the coefficients $c_{\mathbf{z}}$ in the representation of $\vec{f}_{\mathbf{z}}$ satisfy the constraint for every $i = 1, \ldots, m$ $c_{i,\mathbf{z}} \in V_{\mathbf{x}} = \mathrm{span}\left\{x_i - x_j : i, j = 1, \ldots, m\right\}$.*

Proof By the assumption, there exists $(\alpha_{\mathbf{z}}, c_{\mathbf{z}})$ solving the equation $h(\alpha, C) = 0$. So $\nabla\Phi(\alpha_{\mathbf{z}}, c_{\mathbf{z}}) = 0$ and $(\alpha_{\mathbf{z}}, c_{\mathbf{z}})$ gives the representation of $g_{\mathbf{z}}$ and $\vec{f}_{\mathbf{z}}$. By the definition of h, we have $h_i(\alpha_{\mathbf{z}}, c_{\mathbf{z}}) = 0$ which implies $c_{i,\mathbf{z}} \in V_{\mathbf{x}}$. This proves the proposition. ■

Remark 7 *We know the solution $(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}})$ exists and even is unique. This implies the existence of the solution to $\nabla\Phi(\alpha, C) = 0$. But the existence of the solution to $h(\alpha, C) = 0$ is not clear. In fact, this may not be always the case when K is not invertible.*

Proposition 6 states that the coefficients $c_{i,\mathbf{z}}$ are in the span of the pairwise differences of the input data, which is a low dimensional subspace of $\mathbb{R}^n$. This allows us to reduce the dimension of the optimization problem of solving for the coefficients $c_{\mathbf{z}}$. We apply the well known approach of singular value decomposition to the matrix involving the data $\mathbf{x}$ given by

$$M_{\mathbf{x}} = (x_1 - x_m, x_2 - x_m, \ldots, x_{m-1} - x_m, x_m - x_m) \in \mathbb{R}^{n \times m}.$$

Assume the rank of $M_{\mathbf{x}}$ is d. The theory of singular value decomposition tells us that there exists an $n \times n$ orthogonal matrix $V = (V_1, V_2, \ldots, V_n)$ and an $m \times m$ orthogonal matrix $U = (U_1, U_2, \ldots, U_m)$ such that

$$M_{\mathbf{x}} = V \Sigma U^T = (V_1\ V_2\ \cdots\ V_n) \begin{pmatrix} \operatorname{diag}\{\sigma_1, \sigma_2, \cdots, \sigma_d\} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U_1^T \\ U_2^T \\ \vdots \\ U_m^T \end{pmatrix}.$$

Here $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > \sigma_{d+1} = \ldots = \sigma_{\min\{m,n\}} = 0$ are the singular values of $M_{\mathbf{x}}$. The matrix Σ is $n \times m$ and has entries of zero except for $\Sigma_{i,i} = \sigma_i$ for $i = 1, \ldots, d$. The vectors $\{V_i\}_{i=1}^d$ form an orthonormal basis for $V_{\mathbf{x}}$ and denote $V = (V_1, \ldots, V_d)$.

Set $\beta_i \in \mathbb{R}^d$ to satisfy $x_i - x_m = V\beta_i$ for $i = 1, \ldots, m$. For $\gamma^0 \in \mathbb{R}^m$ and $\gamma = (\gamma_1, \ldots, \gamma_m) \in \mathbb{R}^{d \times m}$, define the vector-valued function

$$u = ((u^0)^T, (u_1)^T, \ldots, (u_m)^T)^T : \mathbb{R}^{m(d+1)} \to \mathbb{R}^{m(d+1)}$$

by

$$u^0 = (u_1^0, \ldots, u_m^0)^T, \quad u_j^0(\gamma^0, \gamma) = \frac{1}{m^2} \sum_{i=1}^m w_{i,j} \phi'\big(y_i(k_j\gamma^0 + k_i\gamma^T(\beta_i - \beta_j))\big) y_i + \lambda\gamma_j^0,$$

and, for $i = 1, \ldots, m$,

$$u_i(\gamma^0, \gamma) = \frac{1}{m^2} \sum_{j=1}^m w_{i,j} \phi'\big(y_i(k_j\gamma^0 + k_i\gamma^T(\beta_i - \beta_j))\big) y_i(\beta_i - \beta_j) + \lambda\gamma_i.$$

Proposition 8 *If $\gamma_{\mathbf{z}}^0 \in \mathbb{R}^m$ and $\gamma_{\mathbf{z}} = (\gamma_{1,\mathbf{z}}, \ldots, \gamma_{m,\mathbf{z}}) \in \mathbb{R}^{d \times m}$ are solutions of the equation $u(\gamma^0, \gamma) = 0$, then $c_{\mathbf{z}}$ and $\alpha_{\mathbf{z}}$ defined by*

$$\alpha_{\mathbf{z}} = \gamma_{\mathbf{z}}^0, \qquad c_{i,\mathbf{z}} = V\gamma_{i,\mathbf{z}} \ \text{ for } \ i = 1, \ldots, m,$$

solve $\nabla\Phi(\alpha, C) = 0$ and hence yield a representation of $g_{\mathbf{z}}$ and $\vec{f}_{\mathbf{z}}$ respectively.

Proof By the facts that $c_i = V\gamma_i$ for $i = 1, \ldots, m$ defines a one-to-one mapping from $V_{\mathbf{x}}$ onto $\mathbb{R}^d$ and $V^TV = I_d$ the d-dimensional identity matrix, direct computation shows that $u(\gamma_{\mathbf{z}}^0, \gamma_{\mathbf{z}}) = 0$ implies $h(\alpha_{\mathbf{z}}, c_{\mathbf{z}}) = 0$. Then the conclusion follows from Proposition 5 and Equation (6). ■

We now use Proposition 8 to derive the update rule in Newton's method to optimize the coefficients γ^0 and γ. Let $\eta = ((\gamma^0)^T, (\gamma_1)^T, ..., (\gamma_m)^T)^T \in \mathbb{R}^{m(d+1)}$ and consider the map $u(\eta)$ on $\mathbb{R}^{m(d+1)}$ defined as $u = ((u^0)^T, (u_1)^T, ..., (u_m)^T)^T$. When ϕ is twice differentiable, we can use Newton's method to solve the equation $u(\eta) = 0$ by the following iterative update rule

$$\eta_{t+1} = \eta_t - (\nabla u(\eta_t))^{-1} u(\eta_t).$$

2.1 The Optimization Algorithm

The results of the previous section are summarized here to formulate the algorithm that implements the optimization procedure. Before we state the algorithm we restate the matrices and vectors involved in the optimization:

1. the input data $(x_i)_{i=1}^m$
2. the kernel matrix $\mathbf{K}$ given the kernel function
$$\mathbf{K}_{i,j} = K(x_i, x_j) \text{ for } i, j = 1, .., m$$
3. the elements of the weight matrix given the parameter s
$$w_{i,j} = \exp(-\|x_i - x_j\|^2/2s^2) \text{ for } i, j = 1, .., m$$
4. the label vector computed from the output variables $\mathbf{y} = (y_1, \ldots, y_m)^T$
5. $M_{\mathbf{x}} = [x_1 - x_m, x_2 - x_m, \ldots, x_{m-1} - x_m, x_m - x_m] \in \mathbb{R}^{n \times m}$
6. $V = (v_1, v_2, \ldots, v_d)$ the d left eigenvectors of $M_{\mathbf{x}}^T M_{\mathbf{x}}$
7. $\beta_i = V^T(x_i - x_m)$ for $i = 1$ to m
8. at iteration t we have the vector $\eta_t = ((\gamma^0)^T, (\gamma_1)^T, ..., (\gamma_m)^T)^T \in \mathbb{R}^{m(d+1)}$ with $\gamma^0 := \eta_t(1:m)$, $\gamma_i := \eta_t(m + (i-1)d + 1 : m + id)$, and $\gamma := (\gamma_1, \ldots, \gamma_m)$
9. at each iteration the matrix $\mathbf{a} \in \mathbb{R}^{m \times m}$ is defined by its components
$$\mathbf{a}_{i,j} = w_{i,j} \phi'\big(y_i(k_j \gamma^0 + k_i \gamma^T(\beta_i - \beta_j))\big)$$
where k_i is the i-th column of the kernel matrix
10. at each iteration the matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ is defined by its components
$$\mathbf{A}_{i,j} = w_{i,j} \phi''\big(y_i(k_j \gamma^0 + k_i \gamma^T(\beta_i - \beta_j))\big)$$
11. given the matrix $\mathbf{a}$ we define the vectors $b_0 = \mathbf{a}^T \mathbf{y}$ and
$$b_i = y_i \sum_{j=1}^m \mathbf{a}_{i,j}(\beta_i - \beta_j) \text{ where } i = 1, \ldots, m$$

12. given the matrix $\mathbf{A}$ we define the $m \times m$ matrix

$$K_0 = \text{diag}(\mathbf{A}\mathbf{1}_m)\mathbf{K} \text{ where } \mathbf{1}_m = (1,1,\ldots,1)^T$$

13. from the matrices

$$K_1(j,\ell) = \sum_{i=1}^{m} \mathbf{A}_{i,j} K(x_i, x_\ell)(\beta_i - \beta_j)^T \text{ where } j,\ell = 1,..,m$$

construct the matrix

$$\tilde{K_1} = \begin{pmatrix} K_1(1,1) & \ldots & K_1(1,m) \\ \vdots & \ddots & \vdots \\ K_1(m,1) & \ldots & K_1(m,m) \end{pmatrix}$$

14. from the matrices

$$K_2(i,\ell) = \sum_{j=1}^{m} \mathbf{A}_{i,j} K(x_j, x_\ell)(\beta_i - \beta_j) \text{ where } i,\ell = 1,..,m$$

construct the matrix

$$\tilde{K_2} = \begin{pmatrix} K_2(1,1) & \ldots & K_2(1,m) \\ \vdots & \ddots & \vdots \\ K_2(m,1) & \ldots & K_2(m,m) \end{pmatrix}$$

15. from the matrices

$$B_i = \sum_{j=1}^{m} \mathbf{A}_{i,j}(\beta_i - \beta_j)(\beta_i - \beta_j)^T \text{ where } i = 1,..,m$$

construct the matrix

$$\tilde{K_3} = \begin{pmatrix} B_1K(x_1,x_1) & B_1K(x_1,x_2) & \ldots & B_1K(x_1,x_m) \\ B_2K(x_2,x_1) & B_2K(x_2,x_2) & \ldots & B_2K(x_2,x_m) \\ \vdots & \vdots & \ddots & \vdots \\ B_mK(x_m,x_1) & B_mK(x_m,x_2) & \ldots & B_mK(x_m,x_m) \end{pmatrix}.$$

16. the coefficents of the classification function estimate, $\alpha_{\mathbf{z}} \in \mathbb{R}^m$ and $g_{\mathbf{z}} = \sum_{i=1}^{m} \alpha_{i,\mathbf{z}} K(x, x_i)$

17. the coefficents of the gradient estimate $(c_{i,\mathbf{z}})^T \in \mathbb{R}^p$ for $i = 1,\ldots,m$ and $\vec{f}_{\mathbf{z}} = \sum_{i=1}^{m} c_{i,\mathbf{z}} K(x, x_i)$.

Given the above quantities we now state the algorithm for solving the optimization problem for learning gradients.

Algorithm 1: Algorithm for computing $g_{\mathbf{z}}$ and $\vec{f}_{\mathbf{z}}$

input : inputs $\mathbf{x} = (x_i, \ldots, x_m)$, labels $\mathbf{y} = (y_1, \ldots, y_m)^T$, kernel K, weights $(w_{i,j})$, regularization parameter s, $\lambda > 0$ and threshold $\varepsilon > 0$

return: coefficients $\alpha_{\mathbf{z}}$ and $(c_{i,\mathbf{z}})_{i=1}^m$

$M_{\mathbf{x}} = [x_1 - x_m, x_2 - x_m, \ldots, x_{m-1} - x_m, x_m - x_m]$;
$[V, \Sigma, U] = \text{svd}(M_{\mathbf{x}})$;

$\eta_0 = \mathbf{0}$; stop $\leftarrow$ false; $t \leftarrow 0$;
repeat
  $u(\eta_t) = \frac{1}{m^2}(b_0^T, b_1^T, \ldots, b_m^T)^T + \lambda \eta_t$;
  $\nabla u(\eta_t) = \lambda I_{m(d+1)} + \frac{1}{m^2}\begin{pmatrix} K_0 & \tilde{K_1} \\ \tilde{K_2} & \tilde{K_3} \end{pmatrix}$;
  $\Delta \eta_t = (\nabla u(\eta_t))^{-1} u(\eta_t)$;
  $\eta_{t+1} = \eta_t - \Delta \eta_t$;
  $t \leftarrow t + 1$
  If $\|\Delta \eta_t\| \le \varepsilon$ stop $\leftarrow$ true
until *stop=true* ;

$\alpha_{\mathbf{z}} = \eta_t(1 : m)$;
$\gamma_{i,\mathbf{z}} = \eta_t(m + (i-1)d + 1 : m + id)$ for $i = 1, \ldots, m$;
$c_{i,\mathbf{z}} = V \gamma^{i,\mathbf{z}}$ for $i = 1, \ldots, m$;

3. Error Analysis

In this section, we investigate the statistical performance of the algorithm. We will show that under certain conditions, $g_{\mathbf{z}} \to f_\phi$ and $\vec{f}_{\mathbf{z}} \to \nabla f_\phi$ as $\lambda, s \to 0$. Let us first illustrate this by a specific case where $\phi(\cdot)$ is the logistic loss and $(f_\phi, \nabla f_\phi) \in \mathcal{H}_K^{n+1}$ (this case corresponds to the realizable setting in the PAC learning paradigm). Denote as ∂X the boundary of X and $d(x, \partial X)$ the distance of $x \in X$ from ∂X.

Theorem 9 *Let ϕ be the logistic loss. Assume that for some constants $c_\rho > 0$ and $0 < \theta \le 1$ the marginal distribution ρ_X satisfies*

$$\rho_X\left(\{x \in X : d(x, \partial X) < s\}\right) \le c_\rho s, \tag{7}$$

and the density $p(x)$ of ρ_X exists and satisfies

$$\sup_{x \in X} p(x) \le c_\rho \quad \textit{and} \quad |p(x) - p(u)| \le c_\rho |x - u|^\theta, \ \forall u, x \in X. \tag{8}$$

Suppose that $K \in C^2$ *and* $(f_\phi, \nabla f_\phi) \in \mathcal{H}_K^{n+1}$. *Choose* $\lambda = \lambda(m) = m^{-\frac{2\theta}{3(n+2+2\theta)}}$ *and* $s = s(m) = m^{-\frac{1}{3(n+2+2\theta)}}$. *Then there exists a constant* $C > 0$ *such that for any* $0 < \eta < 1$ *with confidence* $1-\eta$

$$\begin{aligned} \|g_{\mathbf{z}} - f_\phi\|_{L^2_{\rho_X}} &\leq C \log\frac{4}{\eta}\left(\frac{1}{m}\right)^{\frac{\theta}{6(n+2+2\theta)}}, \\ \|\vec{f}_{\mathbf{z}} - \nabla f_\phi\|_{L^2_{\rho_X}} &\leq C \log\frac{4}{\eta}\left(\frac{1}{m}\right)^{\frac{\theta}{6(n+2+2\theta)}}. \end{aligned}$$

Condition (8) means the density of the marginal distribution is Hölder θ. Condition (7) is about the behavior of ρ_X near the boundary of X. When the boundary is piecewise smooth, (8) implies (7).

Theorem 9 is a consequence of the more general Theorem 10 which we prove in Section A.3.

We first define two quantities that will be used extensively.

$$\kappa = \sup_{x\in X}\sqrt{K(x,x)}; \qquad D = \max_{x,u\in X}|x-u|.$$

Note that the reproducing property (2) of the RKHS $\mathcal{H}_K$ implies $\|f\|_\infty \leq \kappa\|f\|_K$ for $f \in \mathcal{H}_K$. This will be used constantly in the following.

For a convex loss function ϕ and $r > 0$, define

$$\begin{aligned} L_r &= \max\left\{|\phi'(\kappa(1+D)r)|, |\phi'(-\kappa(1+D)r)|\right\}, \\ M_r &= \max\left\{\phi(\kappa(1+D)r), \phi(-\kappa(1+D)r)\right\}. \end{aligned}$$

By convexity of ϕ both L_r and M_r increase with r.

Theorem 10 *Let the convex loss function* ϕ *be twice differentiable and satisfy*

$$q_1(T) = \inf_{|t|\leq T}\phi''(t) > 0, \qquad q_2(T) = \sup_{|t|\leq T}\phi''(t) < \infty.$$

Assume ρ *satisfies* (7) *and* (8), $K \in C^2$, $(f_\phi, \nabla f_\phi) \in \mathcal{H}_K^{n+1}$. *Then there exists a constant* $\tilde{C}$ *such that for* $0 < \delta < 1/2$, $0 < s, \lambda \leq 1$ *with probability at least* $1-2\delta$

$$\max\left\{\|g_{\mathbf{z}} - f_\phi\|^2_{L^2_{\rho_X}}, \|\vec{f}_{\mathbf{z}} - \nabla f_\phi\|^2_{L^2_{\rho_X}}\right\} \leq \tilde{C}\left\{r^2 s^\theta + B_r\left(\frac{L_r r + M_r \log\frac{2}{\delta}}{\sqrt{m}s^{n+2}} + s^2 + \lambda\right)s^{-\theta}\right\},$$

where

$$r = \tilde{c}\left\{1 + \frac{s^2}{\lambda} + \left(\frac{L_{\lambda,s}}{\sqrt{\lambda}s^{n+2}} + M_{\lambda,s}\log\frac{2}{\delta}\right)\frac{1}{\sqrt{m}\lambda s^{n+2}}\right\}^{1/2} \tag{9}$$

with some $\tilde{c} \geq 1$, $L_{\lambda,s} = L_{\sqrt{2\phi(0)/\lambda s^{n+2}}}$, *and* $M_{\lambda,s} = M_{\sqrt{2\phi(0)/\lambda s^{n+2}}}$ *and* $B_r = \min\left\{\frac{1}{q_1(c_0 r)}, r\right\}$ *with some* $c_0 > 0$.

Remark 11 *Theorem 10 applies only to the loss functions satisfying* $\phi''(t) > 0$ *because of the requirements on* q_1, *which excludes the SVM hinge loss. As for the quantity* B_r, *we note that it does not increase very quickly with* r. *One can take* $B_r = r$ *for logistic loss and exponential loss where* $q_1(T)$ *decays exponentially fast with* T. *While for the square loss,* $B_r = 1$ *for* $q_1(T) \equiv 1$.

Since the entire proof is rather complicated it has been postponed to the appendix. We now prove Theorem 9 using Theorem 10.

Proof of Theorem 9. Note that for logistic loss, $\phi'(t) = \frac{-e^{-t}}{1+e^{-t}} \in (-1,1)$. So $L_r \leq 1$ and $L_{\lambda,s} \leq 1$. Since $\phi(t) \leq \phi(0) + |t| < 1 + |t|$, we have $M_r \leq (1+\kappa(1+D))r$ when $r \geq 1$ and so $M_{\lambda,s} \leq 2(1+\kappa(1+D))(\lambda s^{n+2})^{-1}$. Also, $\phi''(t) = \frac{2e^{-t}}{(1+e^{-t})^2}$ implies $\frac{1}{q_1(r)} \geq c_0 r$, $q_2(c_0 r) \leq 1/2$. Substitute $L_{\lambda,s}$ and $M_{\lambda,s}$ into (9). The choice of λ, s ensures that $r \leq r_0$ with $r_0 > 1$ an absolute constant. Since B_r, L_r, M_r are increasing with respect to r, so is the upper bound in Theorem 10. Substituting r_0 into this upper bound and by the choice of λ, s, we obtain with confidence at least $1-2\delta$

$$\max\left\{ \|g_{\mathbf{z}} - f_\phi\|^2_{L^2_{\rho_X}}, \|\vec{f}_{\mathbf{z}} - \nabla f_\phi\|^2_{L^2_{\rho_X}} \right\} \leq C \log\frac{2}{\delta} \left(\frac{1}{m}\right)^{\frac{\theta}{3(n+2+2\theta)}},$$

where

$$C = \tilde{C}\left((r_0)^2 + B_{r_0}(L_{r_0} r_0 + M_{r_0} + 2)\right).$$

Setting $\delta = \frac{\eta}{2}$ finishes the proof. ■

Remark 12 *In order to calculate the learning rate, we have imposed a rigid assumption on f_ϕ: both f_ϕ and each element of ∇f_ϕ are in $\mathcal{H}_K$. But the convergence may hold under milder conditions, say, they lie in the closure of $\mathcal{H}_K$ in $L^2_{\rho_X}$. This is in general true if $\mathcal{H}_K$ is dense in $L^2_{\rho_X}$, for example the case of a Gaussian kernel.*

4. Simulated Data and Gene Expression Data

In this section we apply the gradient learning algorithm (3) to the problem of estimating a classification function and simultaneously selecting relevant variables and measuring their covariance. The idea is to rank the importance of variables according to the norm of their partial derivatives $\|\frac{\partial f_\phi}{\partial x^\ell}\|$, since a small norm implies small changes of the classification function with respect to this variable. By our error analysis, we expect $\vec{f}_{\mathbf{z}} \approx \nabla f_\phi$. So we shall use the norms of the components of $\vec{f}_{\mathbf{z}}$ to rank the variables.

Definition 13 *The relative magnitude of the norm for the variables is defined as*

$$s^\phi_\ell = \frac{\|(\vec{f}_{\mathbf{z}})_\ell\|_K}{\left(\sum_{j=1}^n \|(\vec{f}_{\mathbf{z}})_j\|^2_K\right)^{1/2}}, \qquad \ell = 1, \ldots, n.$$

In the same way, we can study coordinate covariances by an empirical matrix.

Definition 14 *The* **empirical gradient matrix** *(EGM), $F_{\mathbf{z}}$, is the $n \times m$ matrix whose columns are $\vec{f}_{\mathbf{z}}(x_j)$ with $j = 1, \ldots, m$. The* **empirical covariance matrix** *(ECM), $\Xi_{\mathbf{z}}$, is the $n \times n$ matrix of inner products of the directional derivative of two coordinates*

$$Cov(\vec{f}_{\mathbf{z}}) := \left[\langle (\vec{f}_{\mathbf{z}})_p, (\vec{f}_{\mathbf{z}})_q \rangle_K\right]^n_{p,q=1} = c_{\mathbf{z}} K c^T_{\mathbf{z}} = \sum_{i,j=1}^m c_{i,\mathbf{z}} c^T_{j,\mathbf{z}} K(x_i, x_j).$$

The ECM gives us the covariance between the coordinates while the EGM gives us information as how the variables differ over different sections of the space.

We apply our idea to three data sets. The first two data sets are artificial ones which we use to illustrate the procedure. The third is a cancer classification problem that has been well studied and serves as further confirmation of the utility of the method. For all three the parameter s of the Gaussian was set as the median of all pairwise distances between points in the data. More experiments including data sets for more challenging classification problems can be found in Mukherjee et al. (2006) where we also developed a novel feature selection procedure via learning gradients.

4.1 Linearly Separable Simulation

Linearly separable data is drawn from two classes in an $n = 80$ dimensional space. Samples from class -1 were drawn from

$$\begin{aligned} x^j &\sim \mathbf{No}(1.5,1), \text{ for } j = 1,\ldots,10, \\ x^j &\sim \mathbf{No}(-3,1), \text{ for } j = 11,\ldots,20, \\ x^j &\sim \mathbf{No}(0,\sigma_{\text{noise}}), \text{ for } j = 21,\ldots,80, \end{aligned}$$

where $\mathbf{No}(\mu,\sigma)$ is the normal distribution with mean μ and standard deviation σ. Samples from class $+1$ were drawn from

$$\begin{aligned} x^j &\sim \mathbf{No}(1.5,1), \text{ for } j = 41,\ldots,50, \\ x^j &\sim \mathbf{No}(-3,1), \text{ for } j = 51,\ldots,60, \\ x^j &\sim \mathbf{No}(0,\sigma_{\text{noise}})), \text{ for } j = 1,\ldots,40,61,\ldots,80. \end{aligned}$$

We ran our algorithm on draws of the above data using a linear kernel and report both the results of the gradient estimate as well as the classification function passed through a logistic function.

Drawing twenty samples from the two respective classes results in a design matrix $\mathbf{x}$ that is 80×40 where the first twenty samples belong to class -1 and the remaining to class $+1$. Figure 1 contains results for data where we set $\sigma_{\text{noise}} = .1$. A draw of this matrix is displayed in Figure 1a. In Figure 1d we display the conditional likelihoods obtained by the classification function on the training data. A leave-one-out analysis yields similar results. For Figure 2 the data was generated with $\sigma_{\text{noise}} = 1$. Note that in the this case standard methods such as PCA would not find the correct features since the variance in all dimensions is equal. The plots corresponding to Figures 2a-d are analogous to those in Figure 1.

In Figures 1b and 2b we plot the norm of each component of the estimate of the gradient, $\{\|(\vec{f}_{\mathbf{z}})_\ell\|_K\}_{\ell=1}^{80}$. The norm of each component gives an indication of the importance of a variable and variables with small norms can be eliminated. Note that the coordinates with large norms are the ones we expect, $\ell = 1,\ldots,20,41,\ldots,60$. Figures 1c and 2c display the empirical covariance matrix. The blocking structure of this matrix indicates the covariance of coordinates.

4.2 Nonlinearly Separable Simulation

Data is drawn from two classes in an $n = 200$ dimensional space that are nonlinearly separable in the first two dimensions. Samples from class $+1$ were drawn from

$$\begin{aligned} (x^1,x^2) &= (r\sin(\theta), r\cos(\theta)), \text{ where } r \sim U[0,1] \text{ and } \theta \sim U[0,2\pi], \\ x^j &\sim \mathbf{No}(0.0,.2), \text{ for } j = 3,\ldots,200, \end{aligned}$$

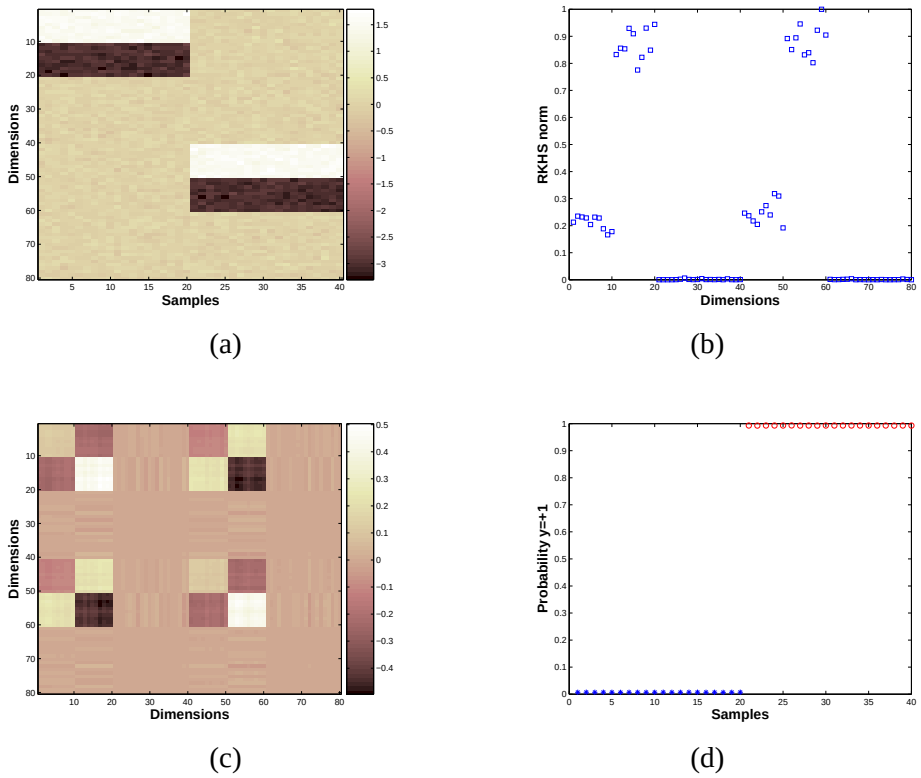


Figure 1: a) The data matrix $\mathbf{x}$ where each sample corresponds to a column and the first twenty samples correspond to class -1 and the second twenty to class $+1$, b) the RKHS norm for each dimension, c) the empirical covariance matrix, d) the predicted class probabilities on the training data.

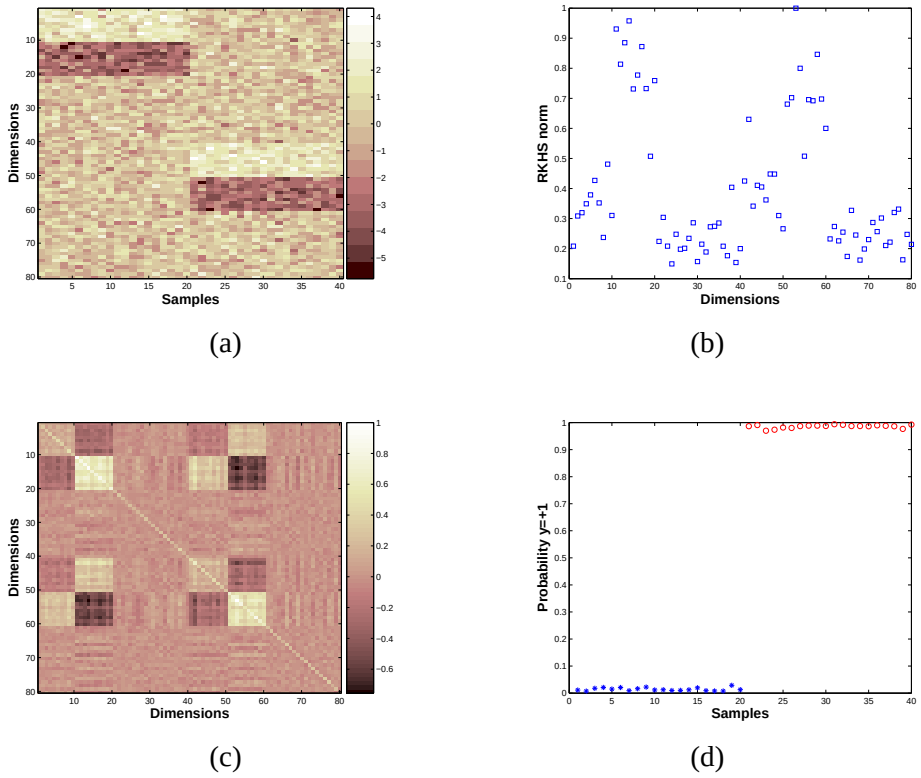


Figure 2: a) The data matrix $\mathbf{x}$ where each sample corresponds to a column and the first twenty samples correspond to class -1 and the second twenty to class $+1$, b) the RKHS norm for each dimension, c) the empirical covariance matrix, d) the predicted class probabilities on the training data.

where $U[a,b]$ is the uniform distribution with support on the interval $[a,b]$. Samples from class -1 were drawn from

$$\begin{aligned}(x^1,x^2) &= (r\sin(\theta), r\cos(\theta)), \text{ where } r \sim U[2,3] \text{ and } \theta \sim U[0,2\pi],\\ x^j &\sim \mathbf{No}(0.0,.2), \text{ for } j = 3,\ldots,200.\end{aligned}$$

Note that the data can be separated by a circle in the first two dimensions.

Drawing thirty samples from the two respective classes results in a design matrix $\mathbf{x}$ that is 200×60 where the first thirty samples belong to class -1 and the remaining to class $+1$. A draw of the first two dimensions of the data is displayed in Figure 3a. Since a linear function cannot accurately classify the data we used a Gaussian kernel

$$K(u,v) = e^{-|u-v|^2/2\sigma^2},$$

where σ was set to the median pairwise distances between points. In the following we report both the results of the gradient estimate as well as the classification function passed through a logistic

function. In Figure 3c,d we plot the norm of each component of the estimate of the gradient. The norm of first two coordinates are much larger than the norm of any of the other coordinates,

$$\frac{\min_{i=1,2} \| \left(\vec{f}_{\mathbf{z}}\right)_i \|_K}{\max_{i=3,\ldots,200} \| \left(\vec{f}_{\mathbf{z}}\right)_i \|_K} > 90.$$

In Figure 3b we plot the ECM. The blocking structure of the ECM indicates the covariance of the first two coordinates. In Figure 3e we display the conditional likelihoods obtained by the classification function on the training data without any feature selection. The classification accuracy improves when we rerun our algorithm using only the dimensions with nonzero norms (3f). The classification results are comparable to what would be obtained by using regularized logistic regression.

4.3 Gene Expression Data

In computational biology, specifically in the subfield of gene expression analysis variable selection and estimation of covariation is of fundamental importance. Microarray technologies enable experimenters to measure the expression level of thousands of genes, the entire genome, at once. The expression level of a gene is proportional to the number of copies of mRNA transcribed by that gene. This readout of gene expression is considered a proxy of the state of the cell. The goals of gene expression analysis include using the expression level of the genes to predict classes, for example tissue morphology or treatment outcome, or real-valued quantities such as drug toxicity or sensitivity. Fundamental to understanding the biology giving rise to the outcome or toxicity is determining which genes are most relevant for the prediction.

4.4 Leukemia Classification

We apply our procedure to a well studied expression data set. The data set is a result of a study using expression data to discriminate acute myeloid leukemia (AML) from acute lymphoblastic leukemia (ALL) (Golub et al., 1999; Slonim et al., 2000) and estimating the genes most relevant to this discrimination. The data set contains 48 samples of AML and 25 samples of ALL. Expression levels of $n = 7,129$ genes and expressed sequence tags (ESTs) were measured via an oligonucleotide microarray for each sample. This data set was split into a training set of 38 samples and a test set of 35 samples.

Various variable selection algorithms have been applied to this data set by using the training set specified in Golub et al. (1999) to select variables and build a classification model and then compute the classification error on the test set. In the same spirit as recursive feature elimination (RFE) we iteratively run our procedure on the training data and remove all variables except for the $\mathcal{S}$ with the largest norm, s_ℓ^ϕ. In Table 1 we report test errors for various values of $\mathcal{S}$ that result from the following procedure:

1. given training data $\mathbf{z}_{7129}$ and test data $\mathbf{tz}_{7129}$ compute the number of errors on the test data $\text{ter}_{7129}(\mathbf{tz}_{7129}) = |\text{sign}[g_{\mathbf{z}_{7129}}(\mathbf{tz}_{7129})] \neq ty|$ and the vector of norms $\{s_\ell^\phi\}_{\ell=1}^{7129}$

2. for $\mathcal{S} = 3000, 1000, 500, 400, 300, 200, 100, 50$ repeat steps 3,4

3. project the test and training data into the dimensions corresponding to the top $\mathcal{S}$ values of $\{s_\ell^\phi\} : \mathbf{z}_\mathcal{S}$ and $\mathbf{tz}_\mathcal{S}$

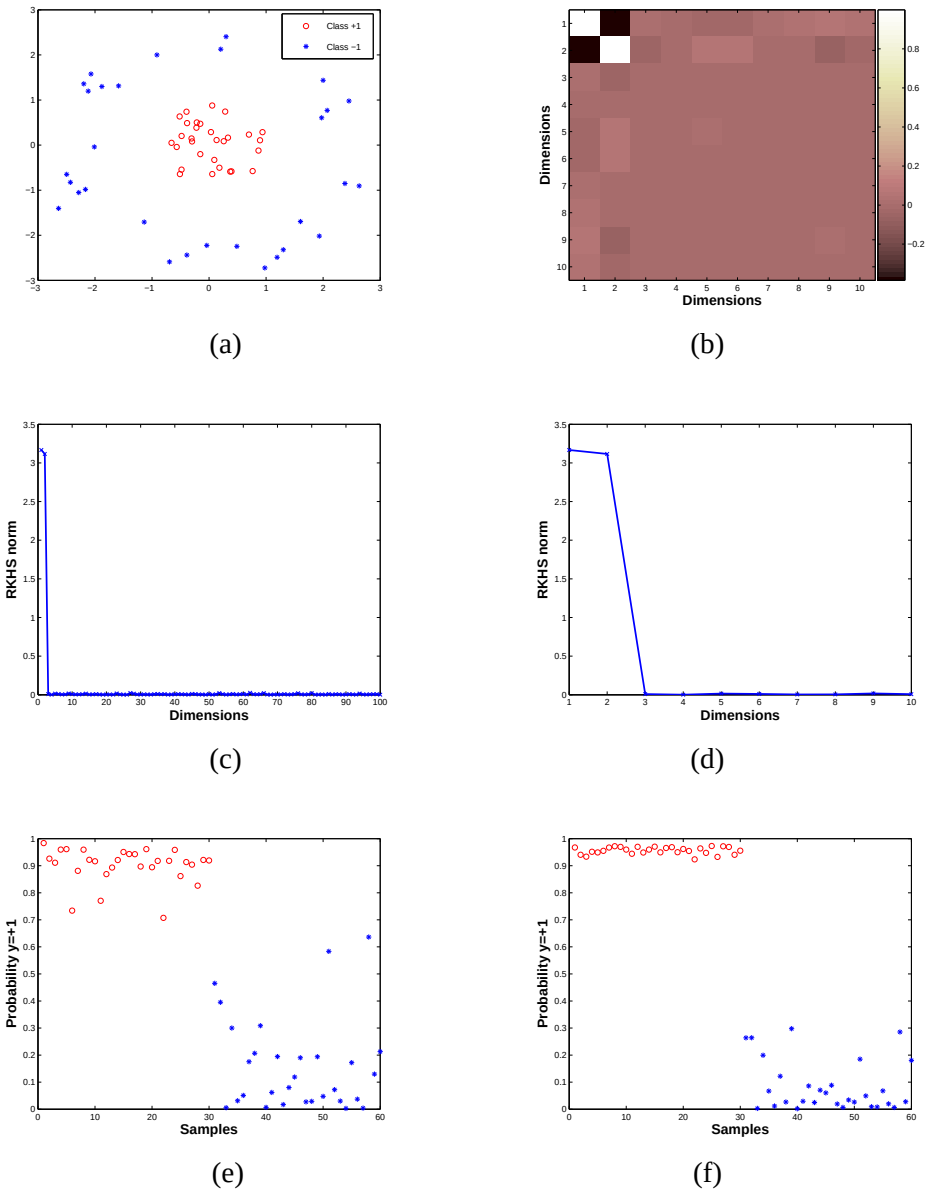


Figure 3: a) The first two dimensions of the data matrix class $+1$ are the circles and class -1 are the stars, b) the empirical covariance matrix for the first 10 dimensions, c) the RKHS norm for the first 100 dimensions , d) the RKHS norm for the first 10 dimensions, e) the predicted class probabilities on the training data with no feature selection again circles are class $+1$ and stars are class -1, f) the predicted class probabilities on the training data with feature selection.

4. given the training data $\mathbf{z}_{\mathcal{S}}$ and test data $\mathbf{tz}_{\mathcal{S}}$ compute the number of errors on the test data $\mathrm{ter}_{\mathcal{S}}(\mathbf{tz}_{\mathcal{S}}) = |\mathrm{sign}[g_{\mathbf{z}_{\mathcal{S}}}(\mathbf{tz}_{\mathcal{S}})] \neq ty|$ and the vector of norms $\{s_\ell^\phi\}$.

The classification accuracy is very similar to other feature selection algorithms such as recursive feature elimination (RFE) (Guyon et al., 2002; Lee et al., 2004) and radius-margin bound (RMB) (Chapelle et al., 2002) both of which were developed specifically for SVMs. In this context we are doing as well as state of the art methods. However, it is important to note that many methods will do very well on this data set and the previously mentioned methods cannot address the issue of covariation.

genes (S)	50	100	200	300	400	500	1,000	3,000	7,129
test errors	2	1	1	1	1	1	1	1	2

Table 1: Number of errors in classification for various values of $\mathcal{S}$ using the genes corresponding to dimensions with the largest norms. The predictions were made using the sign of the classification function output by our method evaluated at each test sample.

In Figure 4a-d we plot the relative magnitude sequence s_ℓ^ϕ for the genes. On this data set the decay in the ranked scores $s_{(\ell)}^\phi$ is steeper than that for most statistics that have been previously used on this data. To illustrate this we compared the gradient score to the Fisher score (Slonim et al., 2000) for each gene

$$t_\ell = \frac{|\hat{\mu}_\ell^{\mathrm{AML}} - \hat{\mu}_\ell^{\mathrm{ALL}}|}{\hat{\sigma}_\ell^{\mathrm{AML}} + \hat{\sigma}_\ell^{\mathrm{ALL}}},$$

where $\hat{\mu}_\ell^{\mathrm{AML}}$ is the mean expression level for the AML samples in the ℓ-th gene, $\hat{\mu}_\ell^{\mathrm{ALL}}$ is the mean expression level for the ALL samples in the ℓ-th gene, $\hat{\sigma}_\ell^{\mathrm{AML}}$ is the standard deviation of the expression level for the AML samples in the ℓ-th gene, and $\hat{\sigma}_\ell^{\mathrm{ALL}}$ is the standard deviation of the expression level for the ALL samples in the ℓ-th gene. We then normalize these scores

$$s_\ell^F = \frac{t_\ell}{\left(\sum_{p=1}^n t_p^2\right)^{1/2}}.$$

Figure 4a-d displays the relative decay of $s_{(\ell)}^\phi$ and $s_{(\ell)}^F$ over various numbers of dimensions. In all plots it is apparent that the decay rate of $s_{(\ell)}^\phi$ is much steeper. Plotting the decay of the elements for the normalized hyperplane $\frac{w^0}{\|w^0\|}$ that is the solution of a linear SVM or the solution of regularized linear logistic regression results in a plot much more like that of the Fisher score than the gradient statistic. Whether and how this steepness (sparsity) has an implication on the generalization error is an open question.

We can also examine the EGM and the ECM. The EGM in this case is a $7,129 \times 38$ matrix and the ECM is $7,129 \times 7,129$ matrix. In Figure 5 we plot the ECM for the 50 dimensions that resulted from the iterative procedure outlined above. This matrix indicates how the dimensions covary and can be used to construct clusters of genes.

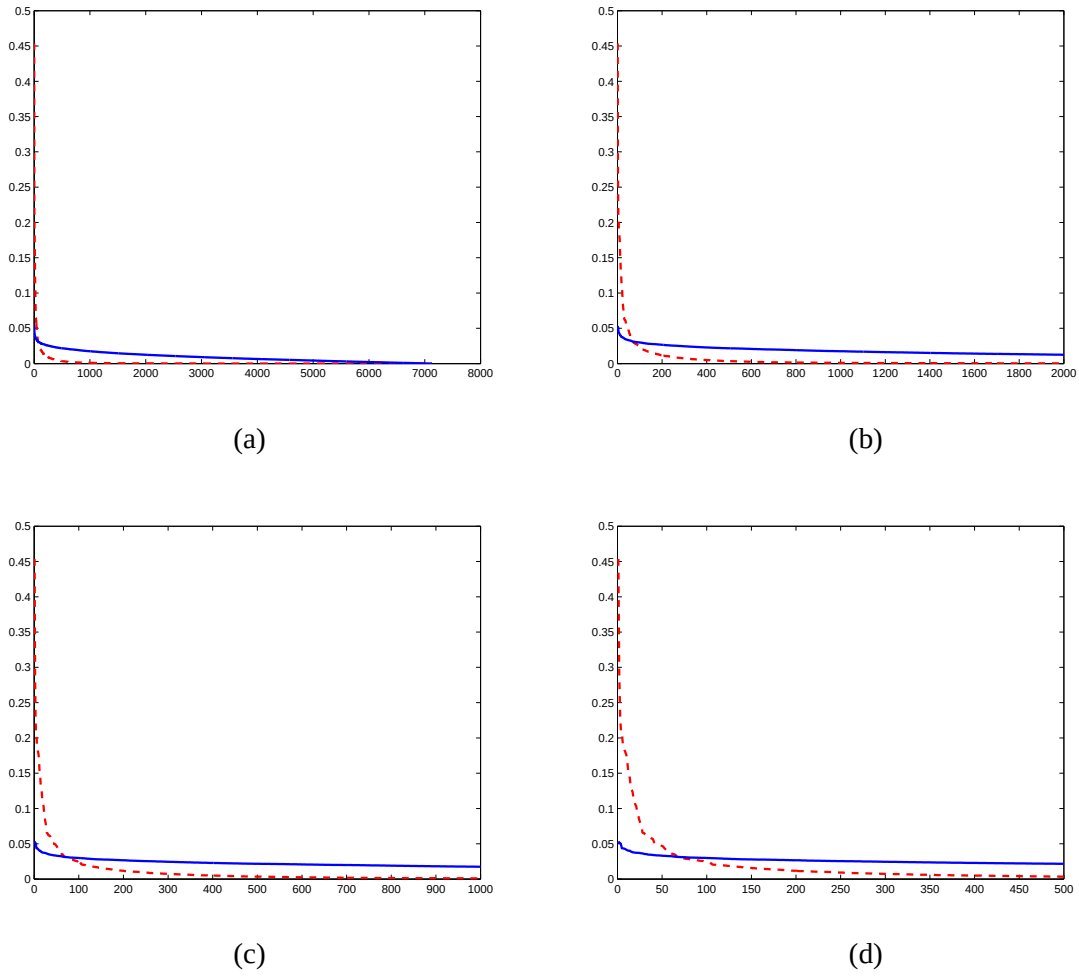


Figure 4: The decay of $s^{\phi}_{(\ell)}$ (dashed line) and $s^{F}_{(\ell)}$ (solid line) over: a) all the genes/dimensions, b) the top 3000 genes/dimensions, c) the top 1000 genes/dimensions, d) the top 500 genes/dimensions.

5. Discussion

We introduce an algorithm that learns a classification function and its gradient from sample data in the logistic regression framework. The relevance of this method for variable selection is motivated. An error analysis is given for the convergence of the estimated classification function and gradient to the true ones respectively. This method also places the problem of variable selection into the powerful framework of Tikhonov regularization. There are many extensions and refinements and open questions regarding this method which we discuss below:

1. Accuracy of classification function: It seems intuitive that the classification function obtained by our method should be strictly worse than that obtained by standard regularized logistic

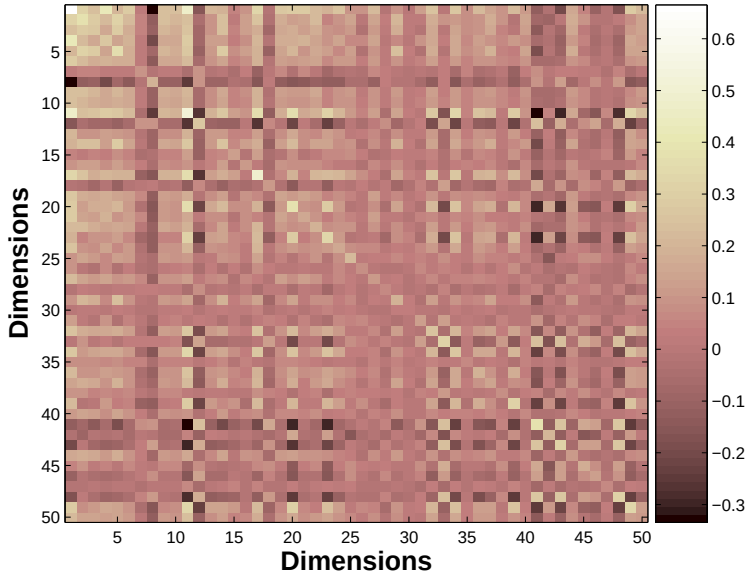


Figure 5: The ECM for the top 50 dimensions.

regression. This is simply a corollary of very useful dictum proposed by Vladimir Vapnik (Vapnik, 1998), "When solving a a given problem, try to avoid solving a more general problem as an intermediate step." Although we strongly expect our classification function to be less accurate than that provided by regularized logistic regression we need to do more empirical work to confirm this.

2. Logistic regression models: An alternative optimization problem was proposed in Mukherjee and Zhou (2006) for estimating the gradient $\vec{f}_{\mathbf{z}}$ in the binary regression problem

$$\vec{f}_{\mathbf{z},\lambda} = \arg\min_{\vec{f}\in\mathcal{H}_K^n} \left\{ \frac{1}{m^2} \sum_{i,j=1}^{m} w_{i,j}^{(s)} \phi\Big(y_i \big(y_j + \vec{f}(x_i)\cdot(x_i - x_j) \big) \Big) + \lambda \|\vec{f}\|_K^2 \right\}.$$

This optimization problem does not follow from the Taylor expansion since in general y_j need not be close to $f_\phi(x_j)$, only the signs of the two functions need agree. This formulation does have an interesting interpretation for variable selection in that variables that are relevant in the classification problem will have large gradient norms and those not relevant will have norms near zero. In practice, for large values of λ the gradient estimates of the above formulation will be similar to those given by the optimization in (3).

3. Fully Bayesian model: The Tikhonov regularization framework coupled with the use of an RKHS allows us to implement a fully Bayesian version of the procedure in the context of Bayesian radial basis (RB) models (Liang et al., 2006). The Bayesian RB framework can be extended to develop a proper probability model for the gradient learning problem. The optimization procedure (3) would be replaced by Markov Chain Monte-carlo methods and the full posterior rather than the maximum a posteriori estimate would be computed. A very

useful result of this is that in addition to the point estimates for the gradient we would also be able to compute credible (confidence) intervals.

4. Intrinsic dimension: In Theorem 9 the rate of convergence of the gradient has the form of $O(m^{-1/n})$ which can be extremely slow if n is large. However, in most data sets and when variable selection is meaningful the data are concentrated on a much lower dimensional manifold embedded in the high dimensional space. In this setting an analysis that replaces the ambient dimension n with the intrinsic dimension of the manifold $n_{\mathcal{M}}$ would be of great interest.

5. Semi-supervised setting: Intrinsic properties of the manifold X can be further studied by unlabeled data. This is one of the motivations of semi-supervised learning. In many applications, it is much easier to obtain unlabeled data with a larger sample size $u \gg m$. For our purpose, unlabeled data $\mathbf{x} = (x_i)_{i=m+1}^{m+u}$ can be used to reduce the dimension or correlation. Since we learn the gradient by $\vec{f}$, it is natural to use the unlabeled data to control the approximate norm of $\vec{f}$ in some Sobolev spaces and introduce a semi-supervised learning algorithm as minimizing over $(g, \vec{f}) \in \mathcal{H}_K^{n+1}$

$$\left\{ \mathcal{E}_{\mathbf{z}}(g, \vec{f}) + \frac{\mu}{(m+u)^2} \sum_{i,j=1}^{m+u} W_{i,j} |\vec{f}(x_i) - \vec{f}(x_j)|^2_{\ell^2(\mathbb{R}^n)} + \lambda \|\vec{f}\|_K^2 \right\},$$

where $\{W_{i,j}\}$ are edge weights in the data adjacency graph, μ is another regularization parameter and often satisfies $\lambda = o(\mu)$.

6. Conclusion

The practical motivation for this work came from a problem in computational biology: pathway extraction. The basic problem is given model systems with known genetic or molecular perturbations infer gene expression "signatures of pathways" (sets of genes that characterize the perturbation in the model system). The term pathway has both a biological and statistical connotation. A statistical definition of a pathway is a set of genes that given a perturbation coordinately differentially co-express with respect to the perturbation. This statistical definition allows us to formulate the biological problem in the mathematical and computational framework of variable (gene) selection. A variety of methods have been proposed for variable selection (Tibshirani, 1996; Chen et al., 1999; Golub et al., 1999; Tusher et al., 2001; Chapelle et al., 2002; Guyon et al., 2002). However, all of these methods have the shortcoming that they cannot determine which variables covary in addition to being salient. This is the primary motivation for the method we propose.

Our proposal is that by studying the gradient of the classification function we can determine which variables are salient with respect to the classification problem and how these variables covary. The conceptual key is that an estimate of the gradient allows us to measure coordinate covariation since the inner product between partial derivatives $\left\langle \frac{\partial f_\phi}{\partial x^j}, \frac{\partial f_\phi}{\partial x^\ell} \right\rangle$ indicates the covariance of the j-th and ℓ-th coordinates with respect to variation in the classification function f_ϕ. This information is of central importance when an understanding is required of the effect of perturbing a salient explanatory variables (input features) on the other explanatory variables in addition to the response variable (the output). The method proposed in this paper gives an estimate of this covariation quantity. We implemented the method and tested it on a variety of simulated and real data sets,

further testing is provided in Mukherjee et al. (2006). These simulations suggest that the method does work for variable selection and some degree of covariation can be estimated. The efficacy of the method was clearly demonstrated on the simulated data and applying the method to gene expression data as well as images of digits (Mukherjee et al., 2006) gave an indication of its utility in understanding models of real data.

The method as currently implemented is designed for the setting of few samples and many dimensions. In this context it is more computationally intensive than methods that consider dimensions separately (Golub et al., 1999; Tusher et al., 2001) and of a similar complexity as methods based upon penalized loss (Tibshirani, 1996; Chen et al., 1999; Chapelle et al., 2002; Guyon et al., 2002). The method as is will scale very poorly as the number of examples increases. This can be addressed by using a basis set different than the difference between data points, for example the bases proposed in Lin and Zhang (2006).

To realize the objective of providing methodology and software to be used by biologists and clinicians for pathway extraction a system that works "right out of the box" is required. This means that the setting of the parameters of our algorithm (see Section 2.1) as well as decisions as to which variables are salient and which covary need to be automated. In addition finding blocks in the covariance matrix is a problem that needs to be addressed. We provide matlab code for the method—`http://www.stat.duke.edu/~sayan/covar1.html`.

Acknowledgments

We would like to thank D.X. Zhou for very useful discussions. We would like to thank the reviewers for many useful suggestions and comments. We would like to acknowledge support for this project from the Institute for Genome Sciences & Policy at Duke as well as the NIH grant for the Center for Public Genomics.

Appendix A. Proof of Theorem 10

The idea behind the proof is to first bound the $L^2_{\rho_X}$ differences by the excess error in Section A.1 and then bound the excess error in Section A.2. The proof is finished in Section A.3.

A.1 Bounding $L^2_{\rho_X}$ Differences by the Excess Error

Recall the empirical error (Definition 1) for $(g, \vec{f}) : X \to \mathbb{R}^{n+1}$

$$\mathcal{E}_{\mathbf{z}}(g, \vec{f}) = \frac{1}{m^2} \sum_{i,j=1}^{m} w_{i,j}^{(s)} \phi\big(y_i(g(x_j) + \vec{f}(x_i) \cdot (x_i - x_j))\big).$$

One can similarly define the expected error

$$\mathcal{E}(g, \vec{f}) = \int_Z \int_X w(x-u)\phi(y(g(u) + \vec{f}(x) \cdot (x-u)))\mathrm{d}\rho_X(u)\mathrm{d}\rho(x,y).$$

Unlike the standard setting of classification and regression $\mathcal{E}(g, \vec{f})$ and $\mathcal{E}_{\mathbf{z}}(g, \vec{f})$ are not respectively the expected and empirical mean of a random variable. This is due to the extra $\mathrm{d}\rho_X$ in the expected

error term. However, since

$$\mathbb{E}_{\mathbf{z}}[\mathcal{E}_{\mathbf{z}}(g,\vec{f})] = \frac{1}{ms^{n+2}}\mathcal{R}(g) + \frac{m-1}{m}\mathcal{E}(g,\vec{f}),$$

the empirical and expected errors should be close to each other if the empirical error concentrates with m increasing.

Define

$$\mathcal{R}_s = \int_X \int_Z w(x-u)\phi(yf_\phi(x))\mathrm{d}\rho(x,y)\mathrm{d}\rho_X(u).$$

We will use the *excess error,* $\mathcal{E}(g,\vec{f}) - \mathcal{R}_s$, to bound the $L^2_{\rho_X}$ differences.

For $r > 0$, denote

$$\mathcal{F}_r = \left\{(g,\vec{f}) \in \mathcal{H}_K^{n+1} : \|g\|_K^2 + \|\vec{f}\|_K^2 \le r^2\right\}.$$

Theorem 15 *Assume ρ_X satisfies the conditions* (7) *and* (8) *and $(f_\phi, \nabla f_\phi) \in \mathcal{H}_K^{n+1}$. For $(g,\vec{f}) \in \mathcal{F}_r$ with some $r > 1$, there exist constants $C_0, C_1 > 0$ such that*

$$\|g - f_\phi\|^2_{L^2_{\rho_X}} \le C_0\left(s^\theta r^2 + s^{2-\theta}B_r(\mathcal{E}(g,\vec{f}) - \mathcal{R}_s)\right)$$

and

$$\|f - \nabla f_\phi\|^2_{L^2_{\rho_X}} \le C_1\left(s^\theta r^2 + s^{-\theta}B_r(\mathcal{E}(g,\vec{f}) - \mathcal{R}_s)\right),$$

where $B_r = \min\left\{\frac{1}{q_1(c_0 r)}, r\right\}$ with some $c_0 > 0$.

To prove Theorem 15 we will need the following several lemmas which require the definition of the following quantities.

Definition 16 *Define for $(g,\vec{f}) : X \to \mathbb{R}^{n+1}$ the square error functional*

$$Q(g,\vec{f}) = \int_X \int_X w(x-u)\left(g(x) - f_\phi(x) + (\vec{f}(x) - \nabla f_\phi(x))\cdot(x-u)\right)^2 \mathrm{d}\rho_X(u)\mathrm{d}\rho_X(x),$$

the border set

$$X_s = \left\{x \in X : d(x,\partial X) > s \text{ and } p(x) \ge (1+c_\rho)s^\theta\right\},$$

and the moments for $0 \le p < \infty$,

$$N_p = \int_{\{t\in\mathbb{R}^n : |t|\le 1\}} e^{-\frac{|t|^2}{2}}|t|^p \mathrm{d}t, \quad \text{and} \quad \tilde{N}_p = \int_{\mathbb{R}^n} e^{-\frac{|t|^2}{2}}|t|^p \mathrm{d}t.$$

Note that X_s is nonempty when s is small enough.

Lemma 17 *Under assumptions of Theorem 15*

$$\frac{N_0}{s^{2-\theta}}\int_{X_s}(g(x) - f_\phi(x))^2 \mathrm{d}\rho_X(x) + \frac{N_2 s^\theta}{n}\int_{X_s}|\vec{f}(x) - \nabla f_\phi(x)|^2 \mathrm{d}\rho_X(x) \le Q(g,\vec{f}).$$

Proof For $x \in X_s$, $\{u \in X : |u-x| \le s\} \subset X$ since $d(x,\partial X) > s$. For $u \in X$ such that $|u-x| \le s$

$$p(u) = p(x) - (p(x) - p(u)) \ge (1+c_\rho)s^\theta - c_\rho |u-x|^\theta \ge s^\theta.$$

Therefore,

$$\begin{aligned}
Q(g,\vec{f}) &\ge \int_{X_s}\int_{|u-x|\le s} w(x-u)\Big(g(x) - f_\phi(x) + (\vec{f}(x) - \nabla f_\phi(x))\cdot(x-u)\Big)^2 p(u)\mathrm{d}u\mathrm{d}\rho_X(x) \\
&\ge s^\theta \int_{X_s}\int_{|u-x|\le s} w(x-u)\Big(g(x) - f_\phi(x) + (\vec{f}(x) - \nabla f_\phi(x))\cdot(x-u)\Big)^2 \mathrm{d}u\mathrm{d}\rho_X(x) \\
&= s^\theta \int_{X_s}\int_{|u-x|\le s} w(x-u)(g(x)-f_\phi(x))^2 \mathrm{d}u\mathrm{d}\rho_X(x) \\
&\quad + 2s^\theta \int_{X_s}\int_{|u-x|\le s} w(x-u)(g(x)-f_\phi(x))((\vec{f}(x) - \nabla f_\phi(x))\cdot(x-u))\mathrm{d}u\mathrm{d}\rho_X(x) \\
&\quad + s^\theta \int_{X_s}\int_{|u-x|\le s} w(x-u)((\vec{f}(x) - \nabla f_\phi(x))\cdot(x-u))^2 \mathrm{d}u\mathrm{d}\rho_X(x) \\
&:= J_1 + J_2 + J_3.
\end{aligned}$$

It can be verified that

$$J_1 = \frac{1}{s^{2-\theta}}\int_{X_s}(g(x)-f_\phi(x))^2 \int_{|t|\le 1} e^{-\frac{|t|^2}{2}}\mathrm{d}t\mathrm{d}\rho_X(x) = \frac{N_0}{s^{2-\theta}}\int_{X_s}|g(x)-f_\phi(x)|^2\mathrm{d}\rho_X(x) \ .$$

In the following, denote by the superscripts of $x,u,t \in \mathbb{R}^n$ the corresponding coordinate indices. For every $i \in \{1,\ldots,n\}$

$$\int_{|u-x|\le s} w(x-u)(x^i-u^i)\mathrm{d}u = \frac{1}{s}\int_{|t|\le 1} e^{-\frac{|t|^2}{2}} t^i \mathrm{d}t = 0.$$

It follows that $J_2 = 0$.

Note that $((\vec{f}(x) - \nabla f_\phi(x))\cdot(x-u))^2$ equals

$$\sum_{i=1}^n\sum_{j=1}^n \left(f^i(x) - \frac{\partial f_\phi}{\partial x^i}(x)\right)\left(f^j(x) - \frac{\partial f_\phi}{\partial x^j}(x)\right)(x^i-u^i)(x^j-u^j).$$

But when $j \ne i$,

$$\int_{|u-x|\le s} w(x-u)(x^i-u^i)(x^j-u^j)\mathrm{d}u = \int_{|t|\le 1} e^{-\frac{|t|^2}{2}} t^i t^j \mathrm{d}t = 0.$$

Therefore

$$J_3 = s^\theta \sum_{i=1}^n \int_{X_s}(f^i(x) - \frac{\partial f_\phi}{\partial x^i}(x))^2 \int_{|t|\le 1} e^{-\frac{|t|^2}{2}}(t^i)^2 \mathrm{d}t\mathrm{d}\rho_X(x) = \frac{N_2 s^\theta}{n}\int_{X_s}|\vec{f}(x) - \nabla f_\phi(x)|^2\mathrm{d}\rho_X(x).$$

Plugging J_1 and J_3 into the inequality completes the proof. ■

In Lemma 20 below we will bound $Q(g,\vec{f})$ by the excess error $\mathcal{E}(g,\vec{f}) - \mathcal{R}_s$. For this purpose, we prove two facts which we state in Lemmas 18 and 19 and define the local error function of $t \in \mathbb{R}$ at $x \in X$ as

$$\mathrm{err}_x(t) = \mathbb{E}_{y\sim Y}[\phi(yt)] = \phi(t)P(1|x) + \phi(-t)P(-1|x),$$

which is a twice differentiable, univariate convex function for every $x \in X$.

Lemma 18 *For almost every $x \in X$, the following hold*

(i) *$f_\phi(x)$ is a minimizer of the function* $\mathrm{err}_x(t)$, i.e., $f_\phi(x) = \arg\min_{t\in\mathbb{R}} \mathrm{err}_x(t)$.

(ii) *If $T > \max\{|t|, \|f_\phi\|_\infty\}$, then*

$$\tfrac{1}{2}q_1(T)(t - f_\phi(x))^2 \le \mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) \le \tfrac{1}{2}q_2(T)(t - f_\phi(x))^2.$$

(iii) *If $T \ge \{|t|, 3\|f_\phi\|_\infty\}$ there exists a constant $c_1 > 0$ such that*

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) \ge c_1 \max\left\{q_1(T), \frac{1}{T}\right\}(t - f_\phi(x))^2.$$

Proof The first conclusion is a direct consequence of the fact

$$\mathcal{R}(f) = \int_X \mathrm{err}_x(f(x))d\rho_x(x).$$

Note that $(\mathrm{err}_x)'(f_\phi(x)) = 0$ since $f_\phi(x)$ is a minimizer of $\mathrm{err}_x(t)$. By a Taylor series expansion, there exists t_0 between t and $f_\phi(x)$ such that

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) = \frac{1}{2}(\mathrm{err}_x)''(t_0)(t - f_\phi(x))^2.$$

Since $(\mathrm{err}_x)''(t_0) = \phi''(t_0)P(1|x) + \phi''(-t_0)P(-1|x)$ and $|t_0| \le T$ the following holds

$$q_1(T) \le \phi''(t_0), \phi''(-t_0) \le q_2(T).$$

It follows $q_1(T) \le (\mathrm{err}_x)''(t_0) \le q_2(T)$ which proves (ii).

To show (iii), write

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) = \int_{f_\phi(x)}^{t}\int_{f_\phi(x)}^{r} (\mathrm{err}_x)''(a)\mathrm{d}a\mathrm{d}r.$$

Since $(\mathrm{err}_x)''(a)$ is positive, if $t \ge 3\|f_\phi\|_\infty := 3M_\phi$, then

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) \ge \int_{2M_\phi}^{t}\int_{|f_\phi(x)|}^{2M_\phi} (\mathrm{err}_x)''(a)\mathrm{d}a\mathrm{d}r \ge q_1(2M_\phi)M_\phi(|t| - 2M_\phi)$$

and, if $t \le -3M_\phi$, then

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) \ge \int_{-2M_\phi}^{t}\int_{-|f_\phi(x)|}^{-2M_\phi} (\mathrm{err}_x)''(a)\mathrm{d}a\mathrm{d}r \ge q_1(2M_\phi)M_\phi(|t| - 2M_\phi).$$

So, if $|t| > 3M_\phi$,

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) \ge q_1(2M_\phi)M_\phi(|t| - 2M_\phi) \ge \frac{3q_1(2M_\phi)M_\phi}{16T}(t - f_\phi(x))^2,$$

where we have used the facts $|t|-2M_\phi \geq \frac{1}{4}|t-f_\phi(x)|$ and $|t-f_\phi(x)| \leq T+M_\phi \leq \frac{4}{3}T$. On the other hand, by (ii), if $|t| \leq 3M_\phi$

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) \geq \frac{1}{2}q_1(3M_\phi)(t-f_\phi(x))^2 \geq \frac{3q_1(3M_\phi)M_\phi}{2T}(t-f_\phi(x))^2.$$

Hence for all $|t| \leq T$,

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) \geq \frac{3q_1(3M_\phi)M_\phi}{16T}(t-f_\phi(x))^2.$$

Together with (ii), we obtain

$$\mathrm{err}_x(t) - \mathrm{err}_x(f_\phi(x)) \geq c_1 \max\left\{q_1(T), \frac{1}{T}\right\}(t-f_\phi(x))^2$$

with $c_1 = \min\left\{\frac{1}{2}, \frac{3q_1(3M_\phi)M_\phi}{16}\right\}$. ■

Lemma 19 *If $K \in C^2$, then there exists a constant $c_K > 0$ depending only on K such that*

$$|f(x)-f(u)| \leq c_K \|f\|_K |x-u|, \quad \forall f \in \mathcal{H}_K,\ x,u \in X.$$

Proof It follows from the reproducing property that

$$|f(x)-f(u)| = |\langle f, K(x,\cdot)-K(u,\cdot)\rangle| \leq \|f\|_K \sqrt{K(x,x)-2K(x,u)+K(u,u)}.$$

Denote $\nabla_1 K(x,u)$ as the gradient of $K(x,u)$ with respect to the first variable x. Since $K \in C^2$, we have

$$\begin{aligned} & K(x,x)-2K(x,u)+K(u,u) \\ = & \int_0^1 (\nabla_1(K(u+t(x-u),x) - \nabla_1 K(u+t(x-u),y)) \cdot (x-u)\mathrm{d}t \\ \leq & \int_0^1 |\nabla_1(K(u+t(x-u),x) - \nabla_1 K(u+t(x-u),y)|\,|x-u|\mathrm{d}t \\ \leq & (c_K)^2|x-u|^2 \end{aligned}$$

with

$$(c_K)^2 = \max\left\{\left\|\frac{\partial^2 K}{\partial x^i \partial u^j}\right\|_\infty, i,j = 1,\ldots,n\right\}.$$

Hence the conclusion is true. ■

Lemma 20 *Under the assumptions of Theorem 15, there exists a constant $c_2 > 0$ such that*

$$Q(g,\vec{f}) \leq c_2\left(r^2 s^2 + B_r(\mathcal{E}(g,\vec{f}) - \mathcal{R}_s)\right),$$

were B_r is defined as in Theorem 15 with $c_0 = \kappa \max\left\{3\|f_\phi\|_K, (1+D)\right\}$.

Proof For $(g,\vec{f})\in\mathcal{F}_r$ and $u,x\in X$, we have

$$|g(u)+\vec{f}(x)(x-u)|\le\kappa\|g\|_K+\kappa D\|\vec{f}\|_K\le c_0 r.$$

Since $c_0 r\ge 3\kappa\|f_\phi\|_K\ge 3\|f_\phi\|_\infty$, by Lemma 18 (iii),

$$\begin{aligned}\mathcal{E}(g,\vec{f})-R_s &= \int_X\int_X w(x-u)\Big(\mathrm{err}_x\big(g(u)+\vec{f}(x)\cdot(x-u)\big)-\mathrm{err}_x\big(f_\phi(x)\big)\Big)\mathrm{d}\rho_x(x)\mathrm{d}\rho_x(u)\\ &\ge \frac{c_1}{c_0}\frac{1}{B_r}\int_X\int_X w(x-u)\Big(g(u)+\vec{f}(x)\cdot(x-u)-f_\phi(x)\Big)^2\mathrm{d}\rho_x(x)\mathrm{d}\rho_x(u),\end{aligned}$$

Denote

$$\begin{aligned}t_1 &= g(u)-f_\phi(u)+(\vec{f}(u)-\nabla f_\phi(u))\cdot(x-u),\\ t_2 &= \big(f_\phi(u)-f_\phi(x)+\nabla f_\phi(u)\cdot(x-u)\big)+(\vec{f}(x)-\vec{f}(u))\cdot(x-u).\end{aligned}$$

We have

$$\mathcal{Q}(g,\vec{f})=\int_X\int_X w(x-u)(t_1)^2\mathrm{d}\rho_x(x)\mathrm{d}\rho_x(u).$$

Note that

$$\Big(g(u)+\vec{f}(x)\cdot(x-u)-f_\phi(x)\Big)^2=(t_1+t_2)^2\ge(t_1)^2+2t_1t_2\ge(t_1)^2-2|t_1||t_2|.$$

There holds

$$\frac{c_0}{c_1}B_r(\mathcal{E}(g,\vec{f})-R_s)\ge\mathcal{Q}(g,\vec{f})-2\int_X\int_X w(x-u)|t_1||t_2|\mathrm{d}\rho_x(x)\mathrm{d}\rho_x(u).$$

By the fact $\nabla f_\phi\in\mathcal{H}_K^n$ and Lemma 19, we have

$$|t_2|\le c_K(\|\nabla f_\phi\|_K+\|\vec{f}\|_K)|x-u|^2\le c_K(\|\nabla f_\phi\|_K+r)|x-u|^2.$$

Together with the assumption $p(x)\le c_\rho$ we obtain

$$\begin{aligned}&\int_X\int_X w(x-u)|t_1||t_2|\mathrm{d}\rho_x(x)\mathrm{d}\rho_x(u)\le\sqrt{\mathcal{Q}(g,\vec{f})}\left(\int_X\int_X w(x-u)|t_2|^2\mathrm{d}\rho_x(x)\mathrm{d}\rho_x(u)\right)^{1/2}\\ &\le c_K\left(\|\nabla f_\phi\|_K+r\right)\sqrt{\mathcal{Q}(g,\vec{f})}\left(c_\rho\int_X\int_{\mathbb{R}^n}w(x-u)|x-u|^4\mathrm{d}x\mathrm{d}\rho_x(u)\right)^{1/2}\\ &\le c_K\left(\|\nabla f_\phi\|_K+r\right)\sqrt{c_\rho\tilde{N}_4 s}\sqrt{\mathcal{Q}(g,\vec{f})}.\end{aligned}$$

Combining the above arguments we obtain

$$\mathcal{Q}(g,\vec{f})-2c_K\left(\|\nabla f_\phi\|_K+r\right)\sqrt{c_\rho\tilde{N}_4 s}\sqrt{\mathcal{Q}(g,\vec{f})}\le\frac{1}{c_1}\min\left\{\frac{1}{q_1(c_0r)},c_0r\right\}(\mathcal{E}(g,\vec{f})-\mathcal{R}_s).$$

Solving this inequality gives

$$\sqrt{\mathcal{Q}(g,\vec{f})}\le 2c_K\left(\|\nabla f_\phi\|_K+r\right)\sqrt{c_\rho\tilde{N}_4 s}+\sqrt{\frac{1}{c_1}\min\left\{\frac{1}{q_1(c_0r)},c_0r\right\}(\mathcal{E}(g,\vec{f})-\mathcal{R}_s)}.$$

This implies the conclusion with $c_2 = 2\max\left\{2(c_K)^2\left(\|\nabla f_\phi\|_K+1\right)^2 c_\rho \tilde{N}_4, \frac{c_0}{c_1}\right\}$. ■

Proof of Theorem 15. Write

$$\|g-f_\phi\|^2_{L^2_{\rho_X}} = \int_{X\setminus X_s}(g(x)-f_\phi(x))^2 d\rho_x(x) + \int_{X_s}(g(x)-f_\phi(x))^2 d\rho_x(x). \tag{10}$$

We have

$$\rho_x(X\setminus X_s) \le c_\rho s + (1+c_\rho)c_\rho|X|s^\theta \le (c_\rho+(1+c_\rho)c_\rho|X|)s^\theta,$$

where $|X|$ is the Lebesgue measure of X. So the first term on the right of (10) is bounded by

$$\kappa^2(r+\|f_\phi\|_K)^2(c_\rho+(1+c_\rho)c_\rho|X|)s^\theta.$$

By Lemmas 17 and 20, the second term on the right of (10) is bounded by

$$\frac{s^{2-\theta}}{N_0}c_2\left(r^2s^2+B_r\left(\mathcal{E}(f,\vec{f})-\mathcal{R}_s\right)\right)$$

Combing these two estimates finishes the proof of the first claim with

$$C_0 = \kappa^2(1+\|f_\phi\|_K)^2(c_\rho+(1+c_\rho)c_\rho|X|)+\frac{c_2}{N_0}.$$

Similarly, we can show the second claim with

$$C_1 = \kappa^2(1+\|\nabla f_\phi\|_K)^2(c_\rho+(1+c_\rho)c_\rho|X|)+\frac{nc_2}{N_2}.$$ ■

In order to apply Theorem 15 to $(g_{\mathbf{z}},\vec{f}_{\mathbf{z}})$, we need a bound on $\|g_{\mathbf{z}}\|^2_K+\|\vec{f}_{\mathbf{z}}\|^2_K$. We first state a rough bound.

Lemma 21 *For every $s>0$ and $\lambda>0$, $\|g_{\mathbf{z}}\|^2_K+\|\vec{f}_{\mathbf{z}}\|^2_K \le \frac{2\phi(0)}{\lambda s^{n+2}}$.*

Proof The conclusion follows from the fact

$$\frac{\lambda}{2}\left(\|g_{\mathbf{z}}\|^2_K+\|\vec{f}_{\mathbf{z}}\|^2_K\right) \le \mathcal{E}_{\mathbf{z}}(g_{\mathbf{z}},\vec{f}_{\mathbf{z}})+\frac{\lambda}{2}\left(\|g_{\mathbf{z}}\|^2_K+\|\vec{f}\|^2_K\right) \le \mathcal{E}_{\mathbf{z}}(0,\vec{0})+0 = \frac{\phi(0)}{s^{n+2}}.$$

■

Remark 22 *Using this quantity the bound in Theorem 15 is at least of order $O(\frac{1}{\lambda s^{n+2-\theta}})$ which tends to ∞ as $s\to 0$ and $\lambda\to 0$. So a sharper bound is needed. We will obtain such a bound in Section A.3.*

A.2 Bounding the Excess Error

In this section, we bound the quantity $\mathcal{E}(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}}) - \mathcal{R}_s$. Let

$$(g_\lambda, \vec{f}_\lambda) = \operatorname*{arg\,min}_{(g,\vec{f})\in\mathcal{H}_K^{n+1}} \left\{ \mathcal{E}(g,\vec{f}) + \frac{\lambda}{2}(\|g\|_K^2 + \|\vec{f}\|_K^2) \right\}.$$

Theorem 23 *If $(f_\phi, \nabla f_\phi) \in \mathcal{H}_K^{n+1}$, $(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}})$ and $(g_\lambda, \vec{f}_\lambda)$ are in $\mathcal{F}_r$ for some $r \geq 1$, then with confidence $1-\delta$*

$$\mathcal{E}(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}}) - \mathcal{R}_s \leq C_2 \left(\frac{L_r r + M_r \log\frac{2}{\delta}}{\sqrt{m} s^{n+2}} + s^2 + \lambda \right),$$

where $C_2 > 0$ is a constant depending on ϕ and ρ but not on r, s and λ.

By a standard decomposition procedure, we have the following result.

Proposition 24 *The following hold*

$$\mathcal{E}(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}}) - \mathcal{R}_s + \frac{\lambda}{2}\left(\|g_{\mathbf{z}}\|_K^2 + \|\vec{f}_{\mathbf{z}}\|_K^2 \right) \leq \mathscr{S}(\mathbf{z}) + \mathscr{A}(\lambda)$$

where

$$\mathscr{S}(\mathbf{z}) = \left(\mathcal{E}(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}}) - \mathcal{E}_{\mathbf{z}}(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}}) \right) + \left(\mathcal{E}_{\mathbf{z}}(g_\lambda, \vec{f}_\lambda) - \mathcal{E}(g_\lambda, \vec{f}_\lambda) \right)$$

and

$$\mathscr{A}(\lambda) = \inf_{(g,\vec{f})\in\mathcal{H}_K^{n+1}} \left\{ \mathcal{E}(g,\vec{f}) - \mathcal{R}_s + \frac{\lambda}{2}\left(\|g\|_K^2 + \|\vec{f}\|_K^2 \right) \right\}.$$

The quantity $\mathscr{S}(\mathbf{z})$ is called the sample error and can be bound by controlling

$$S(\mathbf{z}, r) := \sup_{(g,\vec{f})\in\mathcal{F}_r} |\mathcal{E}_{\mathbf{z}}(g,\vec{f}) - \mathcal{E}(g,\vec{f})|.$$

In fact, if both $(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}})$ and $(g_\lambda, \vec{f}_\lambda)$ are in $\mathcal{F}_r$ for some $r > 0$, then

$$\mathscr{S}(\mathbf{z}) \leq 2S(\mathbf{z}, r). \tag{11}$$

Again $\mathcal{E}_{\mathbf{z}}(g,\vec{f})$ and $\mathcal{E}(g,\vec{f})$ are not the empirical and expected means of a random variable. We will use McDiarmid's inequality (McDiarmid, 1989) to bound $S(\mathbf{z}, r)$.

Lemma 25 *For every $r > 0$*

$$\mathbf{P}\mathrm{rob}\left\{ |S(\mathbf{z}, r) - \mathbb{E}S(\mathbf{z}, r)| > \varepsilon \right\} \leq 2\exp\left(-\frac{m\varepsilon^2 s^{2(n+2)}}{2M_r^2} \right).$$

Proof Denote by $\mathbf{z}_i'$ the sample which coincides with $\mathbf{z}$ except for the i-th pair (x_i, y_i) replaced by (x_i', y_i'). It is easy to verify that

$$\begin{aligned} S(\mathbf{z}, r) - S(\mathbf{z}_i', r) &= \sup_{(g,\vec{f})\in\mathcal{F}_r} \left(\mathcal{E}_{\mathbf{z}}(g,\vec{f}) - \mathcal{E}(g,\vec{f}) \right) - \sup_{(g,\vec{f})\in\mathcal{F}_r} \left(\mathcal{E}_{\mathbf{z}_i'}(g,\vec{f}) - \mathcal{E}(g,\vec{f}) \right) \\ &\leq \sup_{(g,\vec{f})\in\mathcal{F}_r} \left(\mathcal{E}_{\mathbf{z}}(g,\vec{f}) - \mathcal{E}_{\mathbf{z}_i'}(g,\vec{f}) \right) \leq \frac{2m-1}{m^2} \frac{M_r}{s^{n+2}}. \end{aligned}$$

Interchanging the roles of $\mathbf{z}$ and $\mathbf{z}_i'$ gives $|S(\mathbf{z},r)-S(\mathbf{z}_i',r)| \le \frac{2M_r}{ms^{n+2}}$. By McDiarmid's inequality we obtain the desired estimate. ■

Lemma 26 *For every $r>0$*

$$\mathbb{E}S(\mathbf{z},r) \le \frac{8L_r(\kappa(1+2D)r+\phi(0))}{s^{n+2}\sqrt{m}} + \frac{2M_r}{ms^{n+2}}.$$

In order to prove this lemma, we need Rademacher complexities. We refer to Koltchinskii and Panchenko (2000) and van der Vaart and Wellner (1996) for definitions and properties.

Proof Denote $\xi(x,y,u) = w(x-u)\phi(y(g(u)+\vec{f}(x)\cdot(x-u)))$ for simplicity. Then $\mathcal{E}(g,\vec{f}) = \mathbb{E}_u\mathbb{E}_{(x,y)}\xi(x,y,u)$ and $\mathcal{E}_{\mathbf{z}}(g,\vec{f}) = \sum_{i,j=1}^m \xi(x_i,y_i,x_j)$. One can easily check that

$$\begin{aligned}
S(\mathbf{z},r) &\le \sup_{(g,\vec{f})\in\mathcal{F}_r}\left|\mathcal{E}(g,\vec{f}) - \frac{1}{m}\sum_{j=1}^m \mathbb{E}_{(x,y)}\xi(x,y,x_j)\right| + \sup_{(g,\vec{f})\in\mathcal{F}_r}\left|\frac{1}{m}\sum_{j=1}^m \mathbb{E}_{(x,y)}\xi(x,y,x_j) - \mathcal{E}_{\mathbf{z}}(g,\vec{f})\right| \\
&\le \mathbb{E}_{(x,y)}\sup_{(g,\vec{f})\in\mathcal{F}_r}\left|\mathbb{E}_u\xi(x,y,u) - \frac{1}{m}\sum_{i=1}^m \xi(x,y,x_j)\right| \\
&\quad + \frac{1}{m}\sum_{j=1}^m \sup_{(g,\vec{f})\in\mathcal{F}_r}\sup_{u\in X}\left|\mathbb{E}_{(x,y)}\xi(x,y,u) - \frac{1}{m-1}\sum_{\substack{i=1\\ i\ne j}}^m \xi(x_i,y_i,u)\right| \\
&\quad + \frac{1}{m}\sum_{j=1}^m\left(\frac{1}{m}\xi(x_j,y_j,x_j) + \frac{1}{m(m-1)}\sum_{\substack{i=1\\ i\ne j}}^m \xi(x_i,y_i,x_j)\right) \\
&:= S_1+S_2+S_3.
\end{aligned}$$

Let $\varepsilon_i, i=1,\dots,m$ be independent Rademacher variables. Denote

$$G_{(x,y)} = \left\{h(u) = y(g(u)+\vec{f}(x)\cdot(x-u)) : (g,\vec{f})\in\mathcal{F}_r\right\}$$

for every $(x,y)\in Z$. For S_1, by using the properties of Rademacher complexities, we have

$$\begin{aligned}
\mathbb{E}S_1(\mathbf{z}) &= \mathbb{E}_{(x,y)}\mathbb{E}\sup_{h\in G_{x,y}}\left|\mathbb{E}_u[w(x-u)\phi(h(u))] - \frac{1}{m}\sum_{j=1}^m w(x-x_j)\phi(h(x_j))\right| \\
&\le 2\sup_{(x,y)\in Z}\mathbb{E}\sup_{h\in G_{(x,y)}}\left|\frac{1}{m}\sum_{j=1}^m \varepsilon_j w(x-x_j)\phi(h(x_j))\right| \\
&\le \frac{4}{s^{n+2}}\sup_{(x,y)\in Z}\mathbb{E}\sup_{h\in G_{(x,y)}}\left|\frac{1}{m}\sum_{j=1}^m \varepsilon_j\phi(h(x_j))\right| \\
&\le \frac{4L_r}{s^{n+2}}\left(\sup_{(x,y)\in Z}\mathbb{E}\sup_{h\in G_{(x,y)}}\left|\frac{1}{m}\sum_{j=1}^m \varepsilon_j h(x_j)\right| + \frac{\phi(0)}{\sqrt{m}}\right) \\
&\le \frac{4L_r}{s^{n+2}}\left(\mathbb{E}\sup_{\|g\|_K^2\le r^2}\left|\sum_{j=1}^m \varepsilon_j g(x_j)\right| + 2\kappa r\sup_{x\in X}\mathbb{E}\left|\sum_{j=1}^m \varepsilon_j\|x-x_j\|\right| + \frac{\phi(0)}{\sqrt{m}}\right) \\
&\le \frac{4L_r\big(\kappa(1+2D)r+\phi(0)\big)}{s^{n+2}\sqrt{m}}.
\end{aligned}$$

Similarly, we can verify

$$\mathbb{E}S_2(\mathbf{z}) \leq \frac{4L_r(\kappa(1+2D)r+\phi(0))}{s^{n+2}\sqrt{m-1}}.$$

Obviously $S_3 \leq \frac{2M_r}{ms^{n+2}}$. Combining the estimates for S_1, S_2, and S_3 completes the proof. ■

Proposition 27 *Assume $r > 1$. There exists a constant $c_2 > 0$ such that with confidence at least $1-\delta$*

$$\mathscr{S}(\mathbf{z}) \leq c_3 \frac{L_r r + M_r \log\frac{2}{\delta}}{\sqrt{m}s^{n+2}}.$$

Proof The result is a direct application of inequality (11) and Lemmas 25 and 26. ■

We now bound the approximation error $\mathscr{A}(\lambda)$.

Proposition 28 *If $(f_\phi, \nabla f_\phi) \in \mathcal{H}_K^{n+1}$, then $\mathscr{A}(\lambda) \leq c_4(s^2+\lambda)$ for some $c_4 > 0$.*

Proof By the definition of $\mathscr{A}(\lambda)$ and the fact that $(f_\phi, \nabla f_\phi) \in \mathcal{H}_K^{n+1}$

$$\mathscr{A}(\lambda) \leq \mathcal{E}(f_\phi, \nabla f_\phi) - \mathcal{R}_s + \frac{\lambda}{2}(\|f_\phi\|_K^2 + \|\nabla f_\phi\|_K^2).$$

By Lemma 18 (ii), we have

$$\begin{aligned}
\mathcal{E}(f_\phi, \nabla f_\phi) - \mathcal{R}_s &= \int_X \int_X w(x-u)\Big(\mathrm{err}_x(f_\phi(u) + \nabla f_\phi(x)\cdot(x-u)) - \mathrm{err}_x(f_\phi(x)\Big)\mathrm{d}\rho_x(u)\mathrm{d}\rho_x(x) \\
&\leq q_2(\tilde{M}_\phi)\int_X \int_X w(x-u)\Big(f_\phi(u) - f_\phi(x) + \nabla f_\phi(x)\cdot(x-u)\Big)^2 \mathrm{d}\rho_x(u)\mathrm{d}\rho_x(x) \\
&\leq q_2(\tilde{M}_\phi)(c_K)^2\|\nabla f_\phi\|_K^2 c_\rho \int_X \int_X w(x-u)|x-u|^4 \mathrm{d}u\mathrm{d}\rho_x(x) \leq q_2(\tilde{M}_\phi)(c_K)^2\|\nabla f_\phi\|_K^2 c_\rho \tilde{N}_4 s^2,
\end{aligned}$$

where $\tilde{M}_\phi = \kappa\|f_\phi\|_K + \kappa D\|\nabla f_\phi\|_K$. Taking

$$c_4 = \max\{q_2(\tilde{M}_\phi)(c_K)^2\|\nabla f_\phi\|_K^2 c_\rho \tilde{N}_4, \tfrac{1}{2}(\|f_\phi\|_K^2 + \|\nabla f_\phi\|_K^2)\},$$

the result follows. ■

Theorem 23 follows directly from Propositions 24, 27 and 28.

A.3 Proof of Theorem 10

We will use Theorems 15 and 23 to prove Theorem 10.

Notice that both theorems need a bound r so that $(g_\mathbf{z}, \vec{f}_\mathbf{z})$ and $(g_\lambda, \vec{f}_\lambda)$ are in $\mathcal{F}_r$. In Lemma 21 we have shown

$$\|g_\mathbf{z}\|_K^2 + \|\vec{f}_\mathbf{z}\|_K^2 \leq \frac{2\phi(0)}{\lambda s^{n+2}}.$$

Similarly we can show $\|g_\lambda\|_K^2 + \|\vec{f}_\lambda\|_K^2$ is also bounded by $\frac{2\phi(0)}{\lambda s^{n+2}}$. So $\sqrt{\frac{2\phi(0)}{\lambda s^{n+2}}}$ is a universal bound for r such that $(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}})$ and $(g_\lambda, \vec{f}_\lambda)$ are in $\mathcal{F}_r$. However, this bound is not sharp enough to be useful for Theorem 15 (see Remark 22).

A sharper bound will be given below. This bound also improves the sample error estimate and the estimate in Theorem 23.

Lemma 29 *Under the assumptions of Theorem 10*

$$\|g_\lambda\|_K^2 + \|\vec{f}_\lambda\|_K^2 \le 2c_4\left(\frac{s^2}{\lambda} + 1\right).$$

Proof Since $\mathcal{E}(g, \vec{f}) - \mathcal{R}_s$ is non-negative for all pairs $(g, \vec{f})$, we have

$$\tfrac{\lambda}{2}(\|g_\lambda\|_K^2 + \|\vec{f}_\lambda\|_K^2) \le \mathcal{E}(g_\lambda, \vec{f}_\lambda) - \mathcal{R}_s + \tfrac{\lambda}{2}(\|g_\lambda\|_K^2 + \|\vec{f}_\lambda\|_K^2) = \mathscr{A}(\lambda).$$

This in conjunction with Proposition 28 implies the conclusion. ■

Lemma 30 *Under the assumptions of Theorem 10 with confidence at least* $1-\delta$

$$\|g_{\mathbf{z}}\|_K^2 + \|\vec{f}_{\mathbf{z}}\|_K^2 \le c_5\left\{1 + \frac{s^2}{\lambda} + \left(\frac{L_{\lambda,s}}{\sqrt{\lambda s^{n+2}}} + M_{\lambda,s}\log\frac{2}{\delta}\right)\frac{1}{\sqrt{m\lambda s^{n+2}}}\right\}$$

for some $c_5 > 0$.

Proof By the fact $\mathcal{E}(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}}) - \mathcal{R}_s > 0$ and Proposition 24 we have

$$\tfrac{\lambda}{2}\left(\|g_{\mathbf{z}}\|_K^2 + \|\vec{f}_{\mathbf{z}}\|_K^2\right) \le \mathscr{S}(\mathbf{z}) + \mathscr{A}(\lambda).$$

Since both $(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}})$ and $(g_\lambda, \vec{f}_\lambda)$ are in $\mathcal{F}_{\sqrt{2\phi(0)/\lambda s^{n+2}}}$, we apply Proposition 27 to get with probability at least $1-\delta$

$$\mathscr{S}(\mathbf{z}) \le c_3\left(L_{\lambda,s}\sqrt{\frac{2\phi(0)}{\lambda s^{n+2}}} + M_{\lambda,s}\log\frac{2}{\delta}\right)\frac{1}{\sqrt{m}s^{n+2}}.$$

Together with Proposition 28, we obtain the desired estimate with $c_5 = 2\max\{c_3, c_4\}$. ■

We now prove Theorem 10.

Proof of Theorem 10. By Theorems 15 and 23 we have with probability at least $1-\delta$ both $\|g_{\mathbf{z}} - f_\phi\|_{L^2_{\rho_X}}$ and $\||\vec{f}_{\mathbf{z}} - \nabla f_\phi|\|_{L^2_{\rho_X}}^2$ are bounded by

$$\max\{C_0, C_1\}\left\{r^2 s^\theta + C_2 B_r\left(\frac{L_r r + M_r\log\frac{2}{\delta}}{\sqrt{m}s^{n+2}} + s^2 + \lambda\right)s^{-\theta}\right\}, \tag{12}$$

if both $(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}})$ and $(g_\lambda, \vec{f}_\lambda)$ are in $\mathcal{F}_r$ for some $r > 1$. By Lemmas 29 and 30 we can state that both $\{(g_{\mathbf{z}}, \vec{f}_{\mathbf{z}}) \in \mathcal{F}_r\}$ and $\{(g_\lambda, \vec{f}_\lambda) \in \mathcal{F}_r\}$ with probability at least $1-\delta$ if

$$r^2 = \max(c_4, c_5, 1)\left\{1 + \frac{s^2}{\lambda} + \left(\frac{L_{\lambda,s}}{\sqrt{\lambda s^{n+2}}} + M_{\lambda,s}\log\frac{2}{\delta}\right)\frac{1}{\sqrt{m\lambda s^{n+2}}}\right\}.$$

Substituting the above r into (12) gives us the desired bound with confidence at least $1-2\delta$. ■

References

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68(6):337–404, 1950.

B. L. Bartlett, M. L. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2005.

O. Chapelle, V.N. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 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, 1999.

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

T. Evgeniou, M. Pontil, C. Papageorgiou, and T. Poggio. Image representations for object detection using kernel classifiers. In *Proceedings of Asian Conference on Computer Vision*, 2000a.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000b.

T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286: 531–537, 1999.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

L. Hermes and J.M. Buhmann. Feature selection for support vector machines. In *International Conference on Pattern Recognition*, 2000.

V.I. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. In J. Wellner E. Giné, D. Mason, editor, *High Dimensional Probability II*, pages 443–459, 2000.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and applications to the classification of microarray data and satellite radiance data. *J. Amer. Stat. Soc.*, 99:67–81, 2004.

F. Liang, S. Mukherjee, and M. West. Understanding the use of unlabeled data in predictive modeling. *Statistical Science*, 2006. accepted.

Y. Lin and H.H. Zhang. Component selection and smoothing in smoothing spline analysis of variance models. *Annals of Statistics*, 2006. in press.

C. McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, volume 141, pages 148–188. LMS Lecture Notes Series, 1989.

C.A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005.

S. Mukherjee and D.X. Zhou. Learning coordinate covariances via gradients. *Journal of Machine Learning Research*, 7:519–549, 2006.

S. Mukherjee, Q. Wu, and D. X. Zhou. Learning gradients and feature selection on manifolds. *Annals of Statistics*, 2006. submitted.

T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

B. Schoelkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

D.K. Slonim, P. Tamayo, J.P. Mesirov, T.R. Golub, and E.S. Lander. Class prediction and discovery using gene expression data. In *Proc. of the 4th Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 263–272, 2000.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J Royal Stat Soc B*, 58(1):267–288, 1996.

V.G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proc Natl Acad Sci U S A*, 98(9):5116–21, 2001.

A. van der Vaart and J. Wellner. *Weak convergence and empirical processes*. Springer-Verlag, New York, 1996.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

G. Wahba. *Splines Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

Q. Wu and D.X. Zhou. Support vector machine classifiers: Linear programming versus quadratic programming. *Neural Computation*, 17:1160–1187, 2005.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Ann. Statis.*, 32:56–85, 2004.

Journal of Machine Learning Research 7 (2006) 2515-2540 Submitted 2/05; Revised 3/06; Published 11/06

Expectation Correction for Smoothed Inference in Switching Linear Dynamical Systems

David Barber DAVID.BARBER@IDIAP.CH
IDIAP Research Institute
Rue du Simplon 4
CH-1920 Martigny
Switzerland

Editor: David Maxwell Chickering

Abstract

We introduce a method for approximate smoothed inference in a class of switching linear dynamical systems, based on a novel form of Gaussian Sum smoother. This class includes the switching Kalman 'Filter' and the more general case of switch transitions dependent on the continuous latent state. The method improves on the standard Kim smoothing approach by dispensing with one of the key approximations, thus making fuller use of the available future information. Whilst the central assumption required is projection to a mixture of Gaussians, we show that an additional conditional independence assumption results in a simpler but accurate alternative. Our method consists of a single Forward and Backward Pass and is reminiscent of the standard smoothing 'correction' recursions in the simpler linear dynamical system. The method is numerically stable and compares favourably against alternative approximations, both in cases where a single mixture component provides a good posterior approximation, and where a multimodal approximation is required.

Keywords: Gaussian sum smoother, switching Kalman filter, switching linear dynamical system, expectation propagation, expectation correction

1. Switching Linear Dynamical System

The Linear Dynamical System (LDS) (Bar-Shalom and Li, 1998; West and Harrison, 1999) is a key temporal model in which a latent linear process generates the observed time-series. For more complex time-series which are not well described globally by a single LDS, we may break the time-series into segments, each modeled by a potentially different LDS. This is the basis for the Switching LDS (SLDS) where, for each time-step t, a switch variable $s_t \in 1,\ldots,S$ describes which of the LDSs is to be used.[1] The observation (or 'visible' variable) $v_t \in \mathcal{R}^V$ is linearly related to the hidden state $h_t \in \mathcal{R}^H$ by

$$v_t = B(s_t)h_t + \eta^v(s_t), \qquad \eta^v(s_t) \sim \mathcal{N}(\bar{v}(s_t), \Sigma^v(s_t)) \tag{1}$$

where $\mathcal{N}(\mu,\Sigma)$ denotes a Gaussian distribution with mean μ and covariance Σ. The transition dynamics of the continuous hidden state h_t is linear

$$h_t = A(s_t)h_{t-1} + \eta^h(s_t), \qquad \eta^h(s_t) \sim \mathcal{N}\left(\bar{h}(s_t), \Sigma^h(s_t)\right). \tag{2}$$

1. These systems also go under the names Jump Markov model/process, switching Kalman Filter, Switching Linear Gaussian State-Space model, Conditional Linear Gaussian Model.

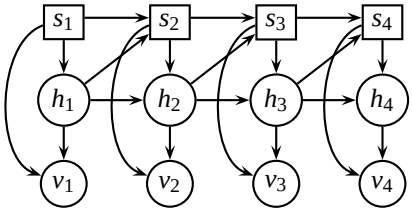


Figure 1: The independence structure of the aSLDS. Square nodes denote discrete variables, round nodes continuous variables. In the SLDS links from h to s are not normally considered.

The dynamics of the switch variables is Markovian, with transition $p(s_t|s_{t-1})$. The SLDS is used in many disciplines, from econometrics to machine learning (Bar-Shalom and Li, 1998; Ghahramani and Hinton, 1998; Lerner et al., 2000; Kitagawa, 1994; Kim and Nelson, 1999; Pavlovic et al., 2001). See Lerner (2002) and Zoeter (2005) for recent reviews of work.

AUGMENTED SWITCHING LINEAR DYNAMICAL SYSTEM

In this article, we will consider the more general model in which the switch s_t is dependent on both the previous s_{t-1} and h_{t-1}. We call this an *augmented Switching Linear Dynamical System*[2] (aSLDS), in keeping with the terminology in Lerner (2002). An equivalent probabilistic model is, as depicted in Figure (1),

$$p(v_{1:T}, h_{1:T}, s_{1:T}) = p(v_1|h_1, s_1)p(h_1|s_1)p(s_1)\prod_{t=2}^{T} p(v_t|h_t, s_t)p(h_t|h_{t-1}, s_t)p(s_t|h_{t-1}, s_{t-1}).$$

The notation $x_{1:T}$ is shorthand for $x_1, \ldots, x_T$. The distributions are parameterized as

$$p(v_t|h_t, s_t) = \mathcal{N}(\bar{v}(s_t) + B(s_t)h_t, \Sigma^v(s_t)), \qquad p(h_t|h_{t-1}, s_t) = \mathcal{N}\left(\bar{h}(s_t) + A(s_t)h_{t-1}, \Sigma^h(s_t)\right)$$

where $p(h_1|s_1) = \mathcal{N}(\mu(s_1), \Sigma(s_1))$. The aSLDS has been used, for example, in state-duration modeling in acoustics (Cemgil et al., 2006) and econometrics (Chib and Dueker, 2004).

INFERENCE

The aim of this article is to address how to perform inference in both the SLDS and aSLDS. In particular we desire the so-called *filtered* estimate $p(h_t, s_t|v_{1:t})$ and the *smoothed* estimate $p(h_t, s_t|v_{1:T})$, for any t, $1 \leq t \leq T$. Both exact filtered and smoothed inference in the SLDS is intractable, scaling exponentially with time (Lerner, 2002). To see this informally, consider the filtered posterior, which may be recursively computed using

$$p(s_t, h_t|v_{1:t}) = \sum_{s_{t-1}} \int_{h_{t-1}} p(s_t, h_t|s_{t-1}, h_{t-1}, v_t)p(s_{t-1}, h_{t-1}|v_{1:t-1}). \tag{3}$$

At timestep 1, $p(s_1, h_1|v_1) = p(h_1|s_1, v_1)p(s_1|v_1)$ is an indexed set of Gaussians. At time-step 2, due to the summation over the states s_1, $p(s_2, h_2|v_{1:2})$ will be an indexed set of S Gaussians; similarly at

2. These models are closely related to *Threshold Regression Models* (Tong, 1990).

time-step 3, it will be S^2 and, in general, gives rise to S^{t-1} Gaussians. More formally, in Lauritzen and Jensen (2001), a general exact method is presented for performing stable inference in such hybrid discrete models with conditional Gaussian potentials. The method requires finding a strong junction tree which, in the SLDS case, means that the discrete variables are placed in a single cluster, resulting in exponential complexity.

The key issue in the (a)SLDS, therefore, is how to perform *approximate* inference in a numerically stable manner. Our own interest in the SLDS stems primarily from acoustic modeling, in which the time-series consists of many thousands of time-steps (Mesot and Barber, 2006; Cemgil et al., 2006). For this, we require a stable and computationally feasible approximate inference, which is also able to deal with state-spaces of high hidden dimension, H.

2. Expectation Correction

Our approach to approximate $p(h_t, s_t|v_{1:T}) \approx \tilde{p}(h_t, s_t|v_{1:T})$ mirrors the Rauch-Tung-Striebel (RTS) 'correction' smoother for the LDS (Rauch et al., 1965; Bar-Shalom and Li, 1998). Readers unfamiliar with this approach will find a short explanation in Appendix (A), which defines the important functions LDSFORWARD and LDSBACKWARD, which we shall make use of for inference in the aSLDS. Our correction approach consists of a single Forward Pass to recursively find the filtered posterior $\tilde{p}(h_t, s_t|v_{1:t})$, followed by a single Backward Pass to correct this into a smoothed posterior $\tilde{p}(h_t, s_t|v_{1:T})$. The Forward Pass we use is equivalent to Assumed Density Filtering (Alspach and Sorenson, 1972; Boyen and Koller, 1998; Minka, 2001). The main contribution of this paper is a novel form of Backward Pass, based on collapsing the smoothed posterior to a mixture of Gaussians.

Unless stated otherwise, all quantities should be considered as approximations to their exact counterparts, and we will therefore usually omit the tildes˜throughout the article.

2.1 Forward Pass (Filtering)

Readers familiar with Assumed Density Filtering (ADF) may wish to continue directly to Section (2.2). The basic idea is to represent the (intractable) posterior using a simpler distribution. This is then propagated forwards through time, conditioned on the new observation, and subsequently collapsed back to the tractable distribution representation—see Figure (2). Our aim is to form a recursion for $p(s_t, h_t|v_{1:t})$, based on a Gaussian mixture approximation of $p(h_t|s_t, v_{1:t})$. Without loss of generality, we may decompose the filtered posterior as

$$p(h_t, s_t|v_{1:t}) = p(h_t|s_t, v_{1:t})p(s_t|v_{1:t}).$$

We will first form a recursion for $p(h_t|s_t, v_{1:t})$, and discuss the switch recursion $p(s_t|v_{1:t})$ later. The full procedure for computing the filtered posterior is presented in Algorithm (1).

The exact representation of $p(h_t|s_t, v_{1:t})$ is a mixture with $O(S^t)$ components. We therefore approximate this with a smaller I_t-component mixture

$$p(h_t|s_t, v_{1:t}) \approx \tilde{p}(h_t|s_t, v_{1:t}) \equiv \sum_{i_t=1}^{I_t} \tilde{p}(h_t|i_t, s_t, v_{1:t})\tilde{p}(i_t|s_t, v_{1:t})$$

where $\tilde{p}(h_t|i_t, s_t, v_{1:t})$ is a Gaussian parameterized with mean[3] $f(i_t, s_t)$ and covariance $F(i_t, s_t)$. The Gaussian mixture weights are given by $\tilde{p}(i_t|s_t, v_{1:t})$. In the above, $\tilde{p}$ represent approximations to the

3. Strictly speaking, we should use the notation $f_t(i_t, s_t)$ since, for each time t, we have a set of means indexed by i_t, s_t. This mild abuse of notation is used elsewhere in the paper.

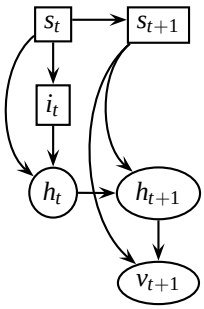


Figure 2: Structure of the mixture representation of the Forward Pass. Essentially, the Forward Pass defines a 'prior' distribution at time t which contains all the information from the variables $v_{1:t}$. This prior is propagated forwards through time using the exact dynamics, conditioned on the observation, and then collapsed back to form a new prior approximation at time $t+1$.

corresponding exact p distributions. To find a recursion for these parameters, consider

$$\begin{aligned}\tilde{p}(h_{t+1}|s_{t+1},v_{1:t+1}) &= \sum_{s_t,i_t} \tilde{p}(h_{t+1},s_t,i_t|s_{t+1},v_{1:t+1}) \\ &= \sum_{s_t,i_t} \tilde{p}(h_{t+1}|i_t,s_t,s_{t+1},v_{1:t+1})\tilde{p}(s_t,i_t|s_{t+1},v_{1:t+1}) \end{aligned} \tag{4}$$

where each of the factors can be recursively computed on the basis of the previous filtered results (see below). However, this recursion suffers from an exponential increase in mixture components. To deal with this, we will later collapse $\tilde{p}(h_{t+1}|s_{t+1},v_{1:t+1})$ back to a smaller mixture. For the remainder, we drop the $\tilde{p}$ notation, and concentrate on computing the r.h.s of Equation (4).

EVALUATING $p(h_{t+1}|s_t,i_t,s_{t+1},v_{1:t+1})$

We find $p(h_{t+1}|s_t,i_t,s_{t+1},v_{1:t+1})$ from the joint distribution $p(h_{t+1},v_{t+1}|s_t,i_t,s_{t+1},v_{1:t})$, which is a Gaussian with covariance and mean elements[4]

$$\begin{aligned}&\Sigma_{hh} = A(s_{t+1})F(i_t,s_t)A^{\mathsf{T}}(s_{t+1}) + \Sigma^h(s_{t+1}),\ \ \Sigma_{vv} = B(s_{t+1})\Sigma_{hh}B^{\mathsf{T}}(s_{t+1}) + \Sigma^v(s_{t+1}) \\ &\Sigma_{vh} = B(s_{t+1})F(i_t,s_t),\ \ \mu_v = B(s_{t+1})A(s_{t+1})f(i_t,s_t),\ \ \mu_h = A(s_{t+1})f(i_t,s_t).\end{aligned} \tag{5}$$

These results are obtained from integrating the forward dynamics, Equations (1,2) over h_t, using the results in Appendix (B). To find $p(h_{t+1}|s_t,i_t,s_{t+1},v_{1:t+1})$ we may then condition $p(h_{t+1},v_{t+1}|s_t,i_t,s_{t+1},v_{1:t})$ on v_{t+1} using the results in Appendix (C)—see also Algorithm (4).

EVALUATING $p(s_t,i_t|s_{t+1},v_{1:t+1})$

Up to a trivial normalization constant the mixture weight in Equation (4) can be found from the decomposition

$$p(s_t,i_t|s_{t+1},v_{1:t+1}) \propto p(v_{t+1}|i_t,s_t,s_{t+1},v_{1:t})p(s_{t+1}|i_t,s_t,v_{1:t})p(i_t|s_t,v_{1:t})p(s_t|v_{1:t}). \tag{6}$$

4. We derive this for $\bar{h}_{t+1},\bar{v}_{t+1} \equiv 0$, to ease notation.

Algorithm 1 aSLDS Forward Pass. Approximate the filtered posterior $p(s_t|v_{1:t}) \equiv \rho_t$, $p(h_t|s_t, v_{1:t}) \equiv \sum_{i_t} w_t(i_t, s_t)\mathcal{N}(f_t(i_t, s_t), F_t(i_t, s_t))$. Also we return the approximate log-likelihood $\log p(v_{1:T})$. We require $I_1 = 1, I_2 \leq S, I_t \leq S \times I_{t-1}$. $\theta_t(s) = A(s), B(s), \Sigma^h(s), \Sigma^v(s), \bar{h}(s), \bar{v}(s)$ for $t > 1$. $\theta_1(s) = A(s), B(s), \Sigma(s), \Sigma^v(s), \mu(s), \bar{v}(s)$

for $s_1 \leftarrow 1$ **to** S **do**
  $\{f_1(1, s_1), F_1(1, s_1), \hat{p}\} = \text{LDSFORWARD}(0, 0, v_1; \theta(s_1))$
  $\rho_1 \leftarrow p(s_1)\hat{p}$
end for

for $t \leftarrow 2$ **to** T **do**
  for $s_t \leftarrow 1$ **to** S **do**
    for $i \leftarrow 1$ **to** I_{t-1}, **and** $s \leftarrow 1$ **to** S **do**
      $\{\mu_{x|y}(i, s), \Sigma_{x|y}(i, s), \hat{p}\} = \text{LDSFORWARD}(f_{t-1}(i, s), F_{t-1}(i, s), v_t; \theta_t(s_t))$
      $p^*(s_t|i, s) \equiv \langle p(s_t|h_{t-1}, s_{t-1} = s)\rangle_{p(h_{t-1}|i_{t-1}=i, s_{t-1}=s, v_{1:t-1})}$
      $p'(s_t, i, s) \leftarrow w_{t-1}(i, s)p^*(s_t|i, s)\rho_{t-1}(s)\hat{p}$
    end for
    Collapse the $I_{t-1} \times S$ mixture of Gaussians defined by $\mu_{x|y}$,$\Sigma_{x|y}$, and weights $p(i, s|s_t) \propto p'(s_t, i, s)$ to a Gaussian with I_t components, $p(h_t|s_t, v_{1:t}) \approx \sum_{i_t=1}^{I_t} p(i_t|s_t, v_{1:t})p(h_t|s_t, i_t, v_{1:t})$. This defines the new means $f_t(i_t, s_t)$, covariances $F_t(i_t, s_t)$ and mixture weights $w_t(i_t, s_t) \equiv p(i_t|s_t, v_{1:t})$.
    Compute $\rho_t(s_t) \propto \sum_{i,s} p'(s_t, i, s)$
  end for
  normalize $\rho_t \equiv p(s_t|v_{1:t})$
  $L \leftarrow L + \log \sum_{s_t, i, s} p'(s_t, i, s)$
end for

The first factor in Equation (6), $p(v_{t+1}|i_t, s_t, s_{t+1}, v_{1:t})$, is a Gaussian with mean μ_v and covariance Σ_{vv}, as given in Equation (5). The last two factors $p(i_t|s_t, v_{1:t})$ and $p(s_t|v_{1:t})$ are given from the previous iteration. Finally, $p(s_{t+1}|i_t, s_t, v_{1:t})$ is found from

$$p(s_{t+1}|i_t, s_t, v_{1:t}) = \langle p(s_{t+1}|h_t, s_t)\rangle_{p(h_t|i_t, s_t, v_{1:t})} \tag{7}$$

where $\langle\cdot\rangle_p$ denotes expectation with respect to p. In the standard SLDS, Equation (7) is replaced by the Markov transition $p(s_{t+1}|s_t)$. In the aSLDS, however, Equation (7) will generally need to be computed numerically. A simple approximation is to evaluate Equation (7) at the mean value of the distribution $p(h_t|i_t, s_t, v_{1:t})$. To take covariance information into account an alternative would be to draw samples from the Gaussian $p(h_t|i_t, s_t, v_{1:t})$ and thus approximate the average of $p(s_{t+1}|h_t, s_t)$ by sampling.[5]

CLOSING THE RECURSION

We are now in a position to calculate Equation (4). For each setting of the variable s_{t+1}, we have a mixture of $I_t \times S$ Gaussians. In order to avoid an exponential explosion in the number of mixture

5. Whilst we suggest sampling as part of the aSLDS update procedure, this does not render the Forward Pass as a form of sequential sampling procedure, such as Particle Filtering. The sampling here is a form of exact sampling, for which no convergence issues arise, being used only to numerically evaluate Equation (7).

components, we numerically collapse this back to I_{t+1} Gaussians to form

$$p(h_{t+1}|s_{t+1},v_{1:t+1}) \approx \sum_{i_{t+1}=1}^{I_{t+1}} p(h_{t+1}|i_{t+1},s_{t+1},v_{1:t+1})p(i_{t+1}|s_{t+1},v_{1:t+1}).$$

Hence the Gaussian components and corresponding mixture weights $p(i_{t+1}|s_{t+1},v_{1:t+1})$ are defined implicitly through a numerical (Gaussian-Mixture to smaller Gaussian-Mixture) collapse procedure, for which any method of choice may be supplied. A straightforward approach that we use in our code is based on repeatedly merging low-weight components, as explained in Appendix (D).

A RECURSION FOR THE SWITCH VARIABLES

A recursion for the switch variables can be found by considering

$$p(s_{t+1}|v_{1:t+1}) \propto \sum_{i_t,s_t} p(i_t,s_t,s_{t+1},v_{t+1},v_{1:t}).$$

The r.h.s. of the above equation is proportional to

$$\sum_{s_t,i_t} p(v_{t+1}|i_t,s_t,s_{t+1},v_{1:t})p(s_{t+1}|i_t,s_t,v_{1:t})p(i_t|s_t,v_{1:t})p(s_t|v_{1:t})$$

where all terms have been computed during the recursion for $p(h_{t+1}|s_{t+1},v_{1:t+1})$.

THE LIKELIHOOD $p(v_{1:T})$

The likelihood $p(v_{1:T})$ may be found by recursing $p(v_{1:t+1}) = p(v_{t+1}|v_{1:t})p(v_{1:t})$, where

$$p(v_{t+1}|v_{1:t}) = \sum_{i_t,s_t,s_{t+1}} p(v_{t+1}|i_t,s_t,s_{t+1},v_{1:t})p(s_{t+1}|i_t,s_t,v_{1:t})p(i_t|s_t,v_{1:t})p(s_t|v_{1:t}).$$

In the above expression, all terms have been computed in forming the recursion for the filtered posterior $p(h_{t+1},s_{t+1}|v_{1:t+1})$.

2.2 Backward Pass (Smoothing)

The main contribution of this paper is to find a suitable way to 'correct' the filtered posterior $p(s_t,h_t|v_{1:t})$ obtained from the Forward Pass into a smoothed posterior $p(s_t,h_t|v_{1:T})$. We initially derive this for the case of a single Gaussian representation—the extension to the mixture case is straightforward and given in Section (2.3). Our derivation holds for both the SLDS and aSLDS. We approximate the smoothed posterior $p(h_t|s_t,v_{1:T})$ by a Gaussian with mean $g(s_t)$ and covariance $G(s_t)$, and our aim is to find a recursion for these parameters. A useful starting point is the exact relation:

$$p(h_t,s_t|v_{1:T}) = \sum_{s_{t+1}} p(s_{t+1}|v_{1:T})p(h_t|s_t,s_{t+1},v_{1:T})p(s_t|s_{t+1},v_{1:T}).$$

The term $p(h_t|s_t, s_{t+1}, v_{1:T})$ may be computed as

$$\begin{aligned} p(h_t|s_t, s_{t+1}, v_{1:T}) &= \int_{h_{t+1}} p(h_t, h_{t+1}|s_t, s_{t+1}, v_{1:T}) \\ &= \int_{h_{t+1}} p(h_t|h_{t+1}, s_t, s_{t+1}, v_{1:T}) p(h_{t+1}|s_t, s_{t+1}, v_{1:T}) \\ &= \int_{h_{t+1}} p(h_t|h_{t+1}, s_t, s_{t+1}, v_{1:t}) p(h_{t+1}|s_t, s_{t+1}, v_{1:T}) \end{aligned} \tag{8}$$

which is in the form of a recursion. This recursion therefore requires $p(h_{t+1}|s_t, s_{t+1}, v_{1:T})$, which we can write as

$$p(h_{t+1}|s_t, s_{t+1}, v_{1:T}) \propto p(h_{t+1}|s_{t+1}, v_{1:T}) p(s_t|s_{t+1}, h_{t+1}, v_{1:t}). \tag{9}$$

The above recursions represent the exact computation of the smoothed posterior. In our approximate treatment, we replace all quantities p with their corresponding approximations $\tilde{p}$. A difficulty is that the functional form of $\tilde{p}(s_t|s_{t+1}, h_{t+1}, v_{1:t})$ in the approximation of Equation (9) is not squared exponential in h_{t+1}, so that $\tilde{p}(h_{t+1}|s_t, s_{t+1}, v_{1:T})$ will not be a mixture of Gaussians.[6] One possibility would be to approximate the non-Gaussian $p(h_{t+1}|s_t, s_{t+1}, v_{1:T})$ (dropping the $\tilde{p}$ notation) by a Gaussian (mixture) by minimizing the Kullback-Leilbler divergence between the two, or performing moment matching in the case of a single Gaussian. A simpler alternative is to make the assumption $p(h_{t+1}|s_t, s_{t+1}, v_{1:T}) \approx p(h_{t+1}|s_{t+1}, v_{1:T})$, see Figure (3). This is a considerable simplification since $p(h_{t+1}|s_{t+1}, v_{1:T})$ is already known from the previous backward recursion. Under this assumption, the recursion becomes

$$p(h_t, s_t|v_{1:T}) \approx \sum_{s_{t+1}} p(s_{t+1}|v_{1:T}) p(s_t|s_{t+1}, v_{1:T}) \left\langle p(h_t|h_{t+1}, s_t, s_{t+1}, v_{1:t}) \right\rangle_{p(h_{t+1}|s_{t+1}, v_{1:T})}. \tag{10}$$

We call the procedure based on Equation (10) Expectation Correction (EC) since it 'corrects' the filtered results which themselves are formed from propagating expectations. In Appendix (E) we show how EC is equivalent to a partial Discrete-Continuous factorized approximation.

Equation (10) forms the basis of the the EC Backward Pass. However, similar to the ADF Forward Pass, the number of mixture components needed to represent the posterior in this recursion grows exponentially as we go backwards in time. The strategy we take to deal with this is a form of Assumed Density Smoothing, in which Equation (10) is interpreted as a propagated dynamics reversal, which will subsequently be collapsed back to an assumed family of distributions—see Figure (4). How we implement the recursion for the continuous and discrete factors is detailed below.[7]

6. In the *exact* calculation, $p(h_{t+1}|s_t, s_{t+1}, v_{1:T})$ *is* a mixture of Gaussians since $p(s_t|s_{t+1}, h_{t+1}, v_{1:t}) = p(s_t, s_{t+1}, h_{t+1}, v_{1:T})/p(s_{t+1}, h_{t+1}, v_{1:T})$ so that the mixture of Gaussians denominator $p(s_{t+1}, h_{t+1}, v_{1:T})$ cancels with the first term in Equation (9), leaving a mixture of Gaussians. However, since in Equation (9) the two terms $p(h_{t+1}|s_{t+1}, v_{1:T})$ and $p(s_t|s_{t+1}, h_{t+1}, v_{1:t})$ are replaced by approximations, this cancellation is not guaranteed.

7. Equation (10) has the pleasing form of an RTS Backward Pass for the continuous part (analogous to LDS case), and a discrete smoother (analogous to a smoother recursion for the HMM). In the Forward-Backward algorithm for the HMM (Rabiner, 1989), the posterior $\gamma_t \equiv p(s_t|v_{1:T})$ is formed from the product of $\alpha_t \equiv p(s_t|v_{1:t})$ and $\beta_t \equiv p(v_{t+1:T}|s_t)$. This approach is also analogous to EP (Heskes and Zoeter, 2002). In the correction approach, a direct recursion for γ_t in terms of γ_{t+1} and α_t is formed, without explicitly defining β_t. The two approaches to inference are known as $\alpha - \beta$ and $\alpha - \gamma$ recursions.

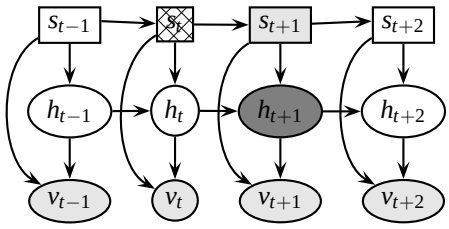


Figure 3: Our Backward Pass approximates $p(h_{t+1}|s_{t+1},s_t,v_{1:T})$ by $p(h_{t+1}|s_{t+1},v_{1:T})$. Motivation for this is that s_t only influences h_{t+1} through h_t. However, h_t will most likely be heavily influenced by $v_{1:t}$, so that not knowing the state of s_t is likely to be of secondary importance. The darker shaded node is the variable we wish to find the posterior state of. The lighter shaded nodes are variables in known states, and the hashed node a variable whose state is indeed known but assumed unknown for the approximation.

EVALUATING $\langle p(h_t|h_{t+1},s_t,s_{t+1},v_{1:t})\rangle_{p(h_{t+1}|s_{t+1},v_{1:T})}$

$\langle p(h_t|h_{t+1},s_t,s_{t+1},v_{1:t})\rangle_{p(h_{t+1}|s_{t+1},v_{1:T})}$ is a Gaussian in h_t, whose statistics we will now compute. First we find $p(h_t|h_{t+1},s_t,s_{t+1},v_{1:t})$ which may be obtained from the joint distribution

$$p(h_t,h_{t+1}|s_t,s_{t+1},v_{1:t}) = p(h_{t+1}|h_t,s_{t+1})p(h_t|s_t,v_{1:t}) \tag{11}$$

which itself can be found using the forward dynamics from the filtered estimate $p(h_t|s_t,v_{1:t})$. The statistics for the marginal $p(h_t|s_t,s_{t+1},v_{1:t})$ are simply those of $p(h_t|s_t,v_{1:t})$, since s_{t+1} carries no extra information about h_t.[8] The remaining statistics are the mean of h_{t+1}, the covariance of h_{t+1} and cross-variance between h_t and h_{t+1},

$$\langle h_{t+1}\rangle = A(s_{t+1})\,f_t(s_t)$$
$$\Sigma_{t+1,t+1} = A(s_{t+1})F_t(s_t)A^{\mathsf{T}}(s_{t+1}) + \Sigma^h(s_{t+1}), \qquad \Sigma_{t+1,t} = A(s_{t+1})F_t(s_t).$$

Given the statistics of Equation (11), we may now condition on h_{t+1} to find $p(h_t|h_{t+1},s_t,s_{t+1},v_{1:t})$. Doing so effectively constitutes a reversal of the dynamics,

$$h_t = \overleftarrow{A}(s_t,s_{t+1})h_{t+1} + \overleftarrow{\eta}(s_t,s_{t+1})$$

where $\overleftarrow{A}(s_t,s_{t+1})$ and $\overleftarrow{\eta}(s_t,s_{t+1}) \sim \mathcal{N}(\overleftarrow{m}(s_t,s_{t+1}),\overleftarrow{\Sigma}(s_t,s_{t+1}))$ are easily found using the conditioned Gaussian results in Appendix (C)—see also Algorithm (5). Averaging the reversed dynamics we obtain a Gaussian in h_t for $\langle p(h_t|h_{t+1},s_t,s_{t+1},v_{1:t})\rangle_{p(h_{t+1}|s_{t+1},v_{1:T})}$ with statistics

$$\mu_t = \overleftarrow{A}(s_t,s_{t+1})g(s_{t+1}) + \overleftarrow{m}(s_t,s_{t+1}), \quad \Sigma_{t,t} = \overleftarrow{A}(s_t,s_{t+1})G(s_{t+1})\overleftarrow{A}^{\mathsf{T}}(s_t,s_{t+1}) + \overleftarrow{\Sigma}(s_t,s_{t+1}).$$

These equations directly mirror the RTS Backward Pass, see Algorithm (5).

8. Integrating over h_{t+1} means that the information from s_{t+1} passing through h_{t+1} via the term $p(h_{t+1}|s_{t+1},h_t)$ vanishes. Also, since s_t is known, no information from s_{t+1} passes through s_t to h_t.

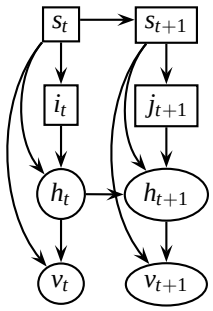


Figure 4: Structure of the Backward Pass for mixtures. Given the smoothed information at time-step $t+1$, we need to work backwards to 'correct' the filtered estimate at time t.

EVALUATING $p(s_t|s_{t+1}, v_{1:T})$

The main departure of EC from previous methods is in treating the term

$$p(s_t|s_{t+1}, v_{1:T}) = \langle p(s_t|h_{t+1}, s_{t+1}, v_{1:t})\rangle_{p(h_{t+1}|s_{t+1}, v_{1:T})}. \tag{12}$$

The term $p(s_t|h_{t+1}, s_{t+1}, v_{1:t})$ is given by

$$p(s_t|h_{t+1}, s_{t+1}, v_{1:t}) = \frac{p(h_{t+1}|s_t, s_{t+1}, v_{1:t})p(s_t, s_{t+1}|v_{1:t})}{\sum_{s'_t} p(h_{t+1}|s'_t, s_{t+1}, v_{1:t})p(s'_t, s_{t+1}|v_{1:t})}. \tag{13}$$

Here $p(s_t, s_{t+1}|v_{1:t}) = p(s_{t+1}|s_t, v_{1:t})p(s_t|v_{1:t})$, where $p(s_{t+1}|s_t, v_{1:t})$ occurs in the Forward Pass, Equation (7). In Equation (13), $p(h_{t+1}|s_{t+1}, s_t, v_{1:t})$ is found by marginalizing Equation (11).

Performing the average over $p(h_{t+1}|s_{t+1}, v_{1:T})$ in Equation (12) may be achieved by any numerical integration method desired. Below we outline a crude approximation that is fast and often performs surprisingly well.

MEAN APPROXIMATION

A simple approximation of Equation (12) is to evaluate the integrand at the mean value of the averaging distribution. Replacing h_{t+1} in Equation (13) by its mean gives the simple approximation

$$\langle p(s_t|h_{t+1}, s_{t+1}, v_{1:t})\rangle_{p(h_{t+1}|s_{t+1}, v_{1:T})} \approx \frac{1}{Z}\frac{e^{-\frac{1}{2}z_{t+1}^{\mathsf{T}}(s_t, s_{t+1})\Sigma^{-1}(s_t, s_{t+1}|v_{1:t})z_{t+1}(s_t, s_{t+1})}}{\sqrt{\det\Sigma(s_t, s_{t+1}|v_{1:t})}}p(s_t|s_{t+1}, v_{1:t})$$

where $z_{t+1}(s_t, s_{t+1}) \equiv \langle h_{t+1}|s_{t+1}, v_{1:T}\rangle - \langle h_{t+1}|s_t, s_{t+1}, v_{1:t}\rangle$ and Z ensures normalization over s_t. This result comes simply from the fact that in Equation (12) we have a Gaussian with a mean $\langle h_{t+1}|s_t, s_{t+1}, v_{1:t}\rangle$ and covariance $\Sigma(s_t, s_{t+1}|v_{1:t})$, being the filtered covariance of h_{t+1} given s_t, s_{t+1} and the observations $v_{1:t}$, which may be taken from Σ_{hh} in Equation (5). Then evaluating this Gaussian at the specific point $\langle h_{t+1}|s_{t+1}, v_{1:T}\rangle$, we arrive at the above expression. An alternative to this simple mean approximation is to sample from the Gaussian $p(h_{t+1}|s_{t+1}, v_{1:T})$, which has the potential advantage that covariance information is used.[9] Other methods such as variational

9. This is a form of exact sampling since drawing samples from a Gaussian is easy. This should not be confused with meaning that this use of sampling renders EC a sequential Monte-Carlo sampling scheme.

Algorithm 2 aSLDS: EC Backward Pass (Single Gaussian case $I = J = 1$). Approximates $p(s_t|v_{1:T})$ and $p(h_t|s_t, v_{1:T}) \equiv \mathcal{N}(g_t(s_t), G_t(s_t))$. This routine needs the results from Algorithm (1) for $I = 1$.

$G_T \leftarrow F_T$, $g_T \leftarrow f_T$,
for $t \leftarrow T-1$ **to** 1 **do**
  for $s \leftarrow 1$ **to** S, $s' \leftarrow 1$ **to** S **do**,
    $(\mu, \Sigma)(s, s') = \text{LDSBACKWARD}(g_{t+1}(s'), G_{t+1}(s'), f_t(s), F_t(s), \theta_{t+1}(s'))$
    $p(s|s') = \langle p(s_t = s|h_{t+1}, s_{t+1} = s', v_{1:t}) \rangle_{p(h_{t+1}|s_{t+1}=s', v_{1:T})}$
    $p(s, s'|v_{1:T}) \leftarrow p(s_{t+1} = s'|v_{1:T}) p(s|s')$
  end for
  for $s_t \leftarrow 1$ **to** S **do**
    Collapse the mixture defined by weights $p(s_{t+1} = s'|s_t, v_{1:T}) \propto p(s_t, s'|v_{1:T})$, means $\mu(s_t, s')$ and covariances $\Sigma(s_t, s')$ to a single Gaussian. This defines the new means $g_t(s_t)$, covariances $G_t(s_t)$.
    $p(s_t|v_{1:T}) \leftarrow \sum_{s'} p(s_t, s'|v_{1:T})$
  end for
end for

approximations to this average (Jaakkola and Jordan, 1996) or the unscented transform (Julier and Uhlmann, 1997) may be employed if desired.

CLOSING THE RECURSION

We have now computed both the continuous and discrete factors in Equation (10), which we wish to use to write the smoothed estimate in the form $p(h_t, s_t|v_{1:T}) = p(s_t|v_{1:T}) p(h_t|s_t, v_{1:T})$. The distribution $p(h_t|s_t, v_{1:T})$ is readily obtained from the joint Equation (10) by conditioning on s_t to form the mixture

$$p(h_t|s_t, v_{1:T}) = \sum_{s_{t+1}} p(s_{t+1}|s_t, v_{1:T}) p(h_t|s_t, s_{t+1}, v_{1:T})$$

which may be collapsed to a single Gaussian (or mixture if desired). As in the Forward Pass, this collapse implicitly defines the Gaussian mean $g(s_t)$ and covariance $G(s_t)$. The smoothed posterior $p(s_t|v_{1:T})$ is given by

$$\begin{aligned} p(s_t|v_{1:T}) &= \sum_{s_{t+1}} p(s_{t+1}|v_{1:T}) p(s_t|s_{t+1}, v_{1:T}) \\ &= \sum_{s_{t+1}} p(s_{t+1}|v_{1:T}) \left\langle p(s_t|h_{t+1}, s_{t+1}, v_{1:t}) \right\rangle_{p(h_{t+1}|s_{t+1}, v_{1:T})} . \end{aligned} \tag{14}$$

The algorithm for the single Gaussian case is presented in Algorithm (2).

NUMERICAL STABILITY

Numerical stability is a concern even in the LDS, and the same is to be expected for the aSLDS. Since the LDS recursions LDSFORWARD and LDSBACKWARD are embedded within the EC algorithm, we may immediately take advantage of the large body of work on stabilizing the LDS recursions, such as the Joseph form (Grewal and Andrews, 1993), or the square root forms (Park and Kailath, 1996; Verhaegen and Van Dooren, 1986).

RELAXING EC

The conditional independence assumption $p(h_{t+1}|s_t, s_{t+1}, v_{1:T}) \approx p(h_{t+1}|s_{t+1}, v_{1:T})$ is not strictly necessary in EC. We motivate it by computational simplicity, since finding an appropriate moment matching approximation of $p(h_{t+1}|s_t, s_{t+1}, v_{1:T})$ in Equation (9) requires a relatively expensive non-Gaussian integration. If we therefore did treat $p(h_{t+1}|s_t, s_{t+1}, v_{1:T})$ more correctly, the central assumption in this relaxed version of EC would be a collapse to a mixture of Gaussians (the additional computation of Equation (12) may usually be numerically evaluated to high precision). Whilst we did not do so, implementing this should not give rise to numerical instabilities since no potential divisions are required, merely the estimation of moments. In the experiments presented here, we did not pursue this option, since we believe that the effect of this conditional independence assumption is relatively weak.

INCONSISTENCIES IN THE APPROXIMATION

The recursion Equation (8), upon which EC depends, makes use of the Forward Pass results, and a subtle issue arises about possible inconsistencies in the Forward and Backward approximations. For example, under the conditional independence assumption in the Backward Pass, $p(h_T|s_{T-1}, s_T, v_{1:T}) \approx p(h_T|s_T, v_{1:T})$, which is in contradiction to Equation (5) which states that the approximation to $p(h_T|s_{T-1}, s_T, v_{1:T})$ *will* depend on s_{T-1}. Similar contradictions occur also for the relaxed version of EC. Such potential inconsistencies arise because of the approximations made, and should not be considered as separate approximations in themselves. Furthermore, these inconsistencies will most likely be strongest at the end of the chain, $t \approx T$, since only then is Equation (8) in direct contradiction to Equation (5). Such potential inconsistencies arise since EC is not founded on a consistency criterion, unlike EP—see Section (3)—but rather an approximation of the exact recursions. Our experience is that compared to EP, which attempts to ensure consistency based on multiple sweeps through the graph, such inconsistencies are a small price to pay compared to the numerical stability advantages of EC.

2.3 Using Mixtures in the Backward Pass

The extension to the mixture case is straightforward, based on the representation

$$p(h_t|s_t, v_{1:T}) \approx \sum_{j_t=1}^{J_t} p(h_t|s_t, j_t, v_{1:T}) p(j_t|s_t, v_{1:T}).$$

Analogously to the case with a single component,

$$p(h_t, s_t|v_{1:T}) = \sum_{i_t, j_{t+1}, s_{t+1}} p(s_{t+1}|v_{1:T}) p(j_{t+1}|s_{t+1}, v_{1:T}) p(h_t|j_{t+1}, s_{t+1}, i_t, s_t, v_{1:T}) \cdot \left\langle p(i_t, s_t|h_{t+1}, j_{t+1}, s_{t+1}, v_{1:t}) \right\rangle_{p(h_{t+1}|j_{t+1}, s_{t+1}, v_{1:T})}.$$

The average in the last line of the above equation can be tackled using the same techniques as outlined in the single Gaussian case. To approximate $p(h_t|j_{t+1}, s_{t+1}, i_t, s_t, v_{1:T})$ we consider this as the marginal of the joint distribution

$$p(h_t, h_{t+1}|i_t, s_t, j_{t+1}, s_{t+1}, v_{1:T}) = p(h_t|h_{t+1}, i_t, s_t, j_{t+1}, s_{t+1}, v_{1:t}) p(h_{t+1}|i_t, s_t, j_{t+1}, s_{t+1}, v_{1:T}).$$

Algorithm 3 aSLDS: EC Backward Pass. Approximates $p(s_t|v_{1:T})$ and $p(h_t|s_t,v_{1:T}) \equiv \sum_{j_t=1}^{J_t} u_t(j_t,s_t)\mathcal{N}(g_t(j_t,s_t),G_t(j_t,s_t))$ using a mixture of Gaussians. $J_T = I_T, J_t \leq S \times I_t \times J_{t+1}$. This routine needs the results from Algorithm (1).

$G_T \leftarrow F_T$, $g_T \leftarrow f_T$, $u_T \leftarrow w_T$ (*)
for $t \leftarrow T-1$ **to** 1 **do**
  for $s \leftarrow 1$ **to** S, $s' \leftarrow 1$ **to** S, $i \leftarrow 1$ **to** I_t, $j' \leftarrow 1$ **to** J_{t+1} **do**
    $(\mu,\Sigma)(i,s,j',s') = \text{LDSBACKWARD}(g_{t+1}(j',s'),G_{t+1}(j',s'),f_t(i,s),F_t(i,s),\theta_{t+1}(s'))$
    $p(i,s|j',s') = \langle p(s_t = s, i_t = i|h_{t+1},s_{t+1} = s', j_{t+1} = j', v_{1:t})\rangle_{p(h_{t+1}|s_{t+1}=s',j_{t+1}=j',v_{1:T})}$
    $p(i,s,j',s'|v_{1:T}) \leftarrow p(s_{t+1} = s'|v_{1:T})u_{t+1}(j',s')p(i,s|j',s')$
  end for
  for $s_t \leftarrow 1$ **to** S **do**
    Collapse the mixture defined by weights $p(i_t = i, s_{t+1} = s', j_{t+1} = j'|s_t,v_{1:T}) \propto p(i,s_t,j',s'|v_{1:T})$, means $\mu(i_t,s_t,j',s')$ and covariances $\Sigma(i_t,s_t,j',s')$ to a mixture with J_t components. This defines the new means $g_t(j_t,s_t)$, covariances $G_t(j_t,s_t)$ and mixture weights $u_t(j_t,s_t)$.
    $p(s_t|v_{1:T}) \leftarrow \sum_{i_t,j',s'} p(i_t,s_t,j',s'|v_{1:T})$
  end for
end for

(*) If $J_T < I_T$ then the initialization is formed by collapsing the Forward Pass results at time T to J_T components.

As in the case of a single mixture, the problematic term is $p(h_{t+1}|i_t,s_t,j_{t+1},s_{t+1},v_{1:T})$. Analogously to before, we may make the assumption

$$p(h_{t+1}|i_t,s_t,j_{t+1},s_{t+1},v_{1:T}) \approx p(h_{t+1}|j_{t+1},s_{t+1},v_{1:T})$$

meaning that information about the current switch state s_t, i_t is ignored.[10] We can then form

$$p(h_t|s_t,v_{1:T}) = \sum_{i_t,j_{t+1},s_{t+1}} p(i_t,j_{t+1},s_{t+1}|s_t,v_{1:T})p(h_t|i_t,s_t,j_{t+1},s_{t+1},v_{1:T}).$$

This mixture can then be collapsed to smaller mixture using any method of choice, to give

$$p(h_t|s_t,v_{1:T}) \approx \sum_{j_t=1}^{J_t} p(h_t|j_t,s_t,v_{1:T})p(j_t|s_t,v_{1:T})$$

The collapse procedure implicitly defines the means $g(j_t,s_t)$ and covariances $G(j_t,s_t)$ of the smoothed approximation. A recursion for the switches follows analogously to the single component Backward Pass. The resulting algorithm is presented in Algorithm (3), which includes using mixtures in both Forward and Backward Passes. Note that if $J_T < I_T$, an extra initial collapse is required of the I_T component Forward Pass Gaussian mixture at time T to J_T components.

EC has time complexity $O(S^2IJK)$ where S are the number of switch states, I and J are the number of Gaussians used in the Forward and Backward passes, and K is the time to compute the exact Kalman smoother for the system with a single switch state.

10. As in the single component case, in principle, this assumption may be relaxed and a moment matching approximation be performed instead.

3. Relation to Other Methods

Approximate inference in the SLDS is a long-standing research topic, generating an extensive literature. See Lerner (2002) and Zoeter (2005) for reviews of previous work. A brief summary of some of the major existing approaches follows.

Assumed Density Filtering Since the exact filtered estimate $p(h_t|s_t, v_{1:t})$ is an (exponentially large) mixture of Gaussians, a useful remedy is to project at each stage of the recursion Equation (3) back to a limited set of K Gaussians. This is a *Gaussian Sum Approximation* (Alspach and Sorenson, 1972), and is a form of *Assumed Density Filtering* (ADF) (Minka, 2001). Similarly, Generalized Pseudo Bayes2 (GPB2) (Bar-Shalom and Li, 1998) also performs filtering by collapsing to a mixture of Gaussians. This approach to filtering is also taken in Lerner et al. (2000) which performs the collapse by removing spatially similar Gaussians, thereby retaining diversity.

Several smoothing approaches directly use the results from ADF. The most popular is Kim's method, which updates the filtered posterior weights to form the smoother (Kim, 1994; Kim and Nelson, 1999). In both EC and Kim's method, the approximation $p(h_{t+1}|s_t, s_{t+1}, v_{1:T}) \approx p(h_{t+1}|s_{t+1}, v_{1:T})$, is used to form a numerically simple Backward Pass. The other approximation in EC is to numerically compute the average in Equation (14). In Kim's method, however, an update for the discrete variables is formed by replacing the required term in Equation (14) by

$$\left\langle p(s_t|h_{t+1}, s_{t+1}, v_{1:t})\right\rangle_{p(h_{t+1}|s_{t+1}, v_{1:T})} \approx p(s_t|s_{t+1}, v_{1:t}). \tag{15}$$

This approximation[11] decouples the discrete Backward Pass in Kim's method from the continuous dynamics, since $p(s_t|s_{t+1}, v_{1:t}) \propto p(s_{t+1}|s_t)p(s_t|v_{1:t})/p(s_{t+1}|v_{1:t})$ can be computed simply from the filtered results alone (the continuous Backward Pass in Kim's method, however, does depend on the discrete Backward Pass). The fundamental difference between EC and Kim's method is that the approximation (15) is not required by EC. The EC Backward Pass therefore makes fuller use of the future information, resulting in a recursion which intimately couples the continuous and discrete variables. The resulting effect on the quality of the approximation can be profound, as we will see in the experiments.

Kim's smoother corresponds to a potentially severe loss of future information and, in general, cannot be expected to improve much on the filtered results from ADF. The more recent work of Lerner et al. (2000) is similar in spirit to Kim's method, whereby the contribution from the continuous variables is ignored in forming an approximate recursion for the smoothed $p(s_t|v_{1:T})$. The main difference is that for the discrete variables, Kim's method is based on a correction smoother (Rauch et al., 1965), whereas Lerner's method uses a Belief Propagation style Backward Pass (Jordan, 1998). Neither method correctly integrates information from the continuous variables. How to form a recursion for a mixture approximation which does not ignore information coming through the continuous hidden variables is a central contribution of our work.

Kitagawa (1994) used a two-filter method in which the dynamics of the chain are reversed. Essentially, this corresponds to a Belief Propagation method which defines a Gaussian sum

11. In the HMM this is exact, but in the SLDS the future observations carry information about s_t.

	EC	Relaxed EC	EP	Kim
Mixture Collapsing to Single			x	
Mixture Collapsing to Mixture	x	x		x
Cond. Indep. $p(h_{t+1}\|s_t,s_{t+1},v_{1:T}) \approx p(h_{t+1}\|s_{t+1},v_{1:T})$	x			x
Approx. of $p(s_t\|s_{t+1},v_{1:T})$, average Equation (12)	x	x		
Kim's Backward Pass				x
Mixture approx. of $p(h_{t+1}\|s_t,s_{t+1},v_{1:T})$, Equation (9)		x		

Table 1: Relation between methods. In the EC methods, the mean approximation may be replaced by an essentially exact Monte Carlo approximation to Equation (12). EP refers to the Single Gaussian approximation in Heskes and Zoeter (2002). In the case of using Relaxed EC with collapse to a single Gaussian, EC and EP are not equivalent, since the underlying recursions on which the two methods are based are fundamentally different.

approximation for $p(v_{t+1:T}|h_t,s_t)$. However, since this is not a density in h_t,s_t, but rather a conditional likelihood, formally one cannot treat this using density propagation methods. In Kitagawa (1994), the singularities resulting from incorrectly treating $p(v_{t+1:T}|h_t,s_t)$ as a density are heuristically finessed.

Expectation Propagation EP (Minka, 2001), as applied to the SLDS, corresponds to an approximate implementation of Belief Propagation[12] (Jordan, 1998; Heskes and Zoeter, 2002). EP is the most sophisticated rival to Kim's method and EC, since it makes the least assumptions. For this reason, we'll explain briefly how EP works. Unlike EC, which is based on an approximation of the exact filtering and smoothing recursions, EP is based on a consistency criterion.

First, let's simplify the notation, and write the distribution as $p=\prod_t \phi(x_{t-1},v_{t-1},x_t,v_t)$, where $x_t \equiv h_t \otimes s_t$, and $\phi(x_{t-1},v_{t-1},x_t,v_t) \equiv p(x_t|x_{t-1})p(v_t|x_t)$. EP defines 'messages' ρ, λ[13] which contain information from past and future observations respectively.[14] Explicitly, we define $\rho_t(x_t) \propto p(x_t|v_{1:t})$ to represent knowledge about x_t given all information from time 1 to t. Similarly, $\lambda_t(x_t)$ represents knowledge about state x_t given all observations from time T to time $t+1$. In the sequel, we drop the time suffix for notational clarity. We define $\lambda(x_t)$ implicitly through the requirement that the marginal smoothed inference is given by

$$p(x_t|v_{1:T}) \propto \rho(x_t)\lambda(x_t). \tag{16}$$

Hence $\lambda(x_t) \propto p(v_{t+1:T}|x_t,v_{1:t}) = p(v_{t+1:T}|x_t)$ and represents all future knowledge about $p(x_t|v_{1:T})$. From this

$$p(x_{t-1},x_t|v_{1:T}) \propto \rho(x_{t-1})\phi(x_{t-1},v_{t-1},x_t,v_t)\lambda(x_t). \tag{17}$$

12. Non-parametric belief propagation (Sudderth et al., 2003), which performs approximate inference in general continuous distributions, is also related to EP applied to the aSLDS, in the sense that the messages cannot be represented easily, and are approximated by mixtures of Gaussians.
13. These correspond to the α and β messages in the Hidden Markov Model framework (Rabiner, 1989).
14. In this Belief Propagation/EP viewpoint, the backward messages, traditionally labeled as β, correspond to conditional likelihoods, and not distributions. In contrast, in the EC approach, which is effectively a so-called $\alpha-\gamma$ recursion, the backward γ messages correspond to posterior distributions.

Taking the above equation as a starting point, we have

$$p(x_t|v_{1:T}) \propto \int_{x_{t-1}} \rho(x_{t-1})\,\phi(x_{t-1}, v_{t-1}, x_t, v_t)\,\lambda(x_t).$$

Consistency with Equation (16) requires (neglecting irrelevant scalings)

$$\rho(x_t)\,\lambda(x_t) \propto \int_{x_{t-1}} \rho(x_{t-1})\,\phi(x_{t-1}, v_{t-1}, x_t, v_t)\,\lambda(x_t).$$

Similarly, we can integrate Equation (17) over x_t to get the marginal at time x_{t-1} which, by consistency, should be proportional to $\rho(x_{t-1})\,\lambda(x_{t-1})$. Hence

$$\rho(x_t) \propto \frac{\int_{x_{t-1}} \rho(x_{t-1})\,\phi(x_{t-1}, x_t)\,\lambda(x_t)}{\lambda(x_t)}, \lambda(x_{t-1}) \propto \frac{\int_{x_t} \rho(x_{t-1})\,\phi(x_{t-1}, x_t)\,\lambda(x_t)}{\rho(x_{t-1})} \tag{18}$$

where the divisions can be interpreted as preventing over-counting of messages. In an exact implementation, the common factors in the numerator and denominator cancel. EP addresses the fact that $\lambda(x_t)$ is not a distribution by using Equation (18) to form the projection (or 'collapse'). In the numerator, $\int_{x_{t-1}} \rho(x_{t-1})\,\phi(x_{t-1}, x_t)\,\lambda(x_t)$ and $\int_{x_t} \rho(x_{t-1})\,\phi(x_{t-1}, x_t)\,\lambda(x_t)$ represent $p(x_t|v_{1:T})$ and $p(x_{t-1}|v_{1:T})$. Since these *are* distributions (an indexed mixture of Gaussians in the SLDS), they may be projected/collapsed to a single indexed Gaussian. The update for the ρ message is then found from division by the λ potential, and vice versa. In EP the explicit division of potentials only makes sense for members of the exponential family. More complex methods could be envisaged in which, rather than an explicit division, the new messages are defined by minimizing some measure of divergence between $\rho(x_t)\lambda(x_t)$ and $\int_{x_{t-1}} \rho(x_{t-1})\,\phi(x_{t-1}, x_t)\,\lambda(x_t)$, such as the Kullback-Leibler divergence. In this way, non-exponential family approximations (such as mixtures of Gaussians) may be considered. Whilst this is certainly feasible, it is somewhat unattractive computationally since this would require for each time-step an expensive minimization.

For the single Gaussian case, in order to perform the division, the potentials in the numerator and denominator are converted to their canonical representations. To form the ρ update, the result of the division is then reconverted back to a moment representation. The resulting recursions, due to the approximation, are no longer independent and Heskes and Zoeter (2002) show that using more than a single Forward and Backward sweep often improves on the quality of the approximation. This coupling is a departure from the exact recursions, which should remain independent.

Applied to the SLDS, EP suffers from severe numerical instabilities (Heskes and Zoeter, 2002) and finding a way to minimize the corresponding EP free energy in an efficient, robust and guaranteed way remains an open problem. Our experience is that current implementations of EP are unsuitable for large scale time-series applications. Damping the parameter updates is one suggested approach to heuristically improve convergence. The source of these numerical instabilities is not well understood since, even in cases when the posterior appears uni-modal, the method is problematic. The frequent conversions between moment and canonical parameterizations of Gaussians are most likely at the root of the difficulties. An interesting comparison here is between Lauritzen's original method for exact computation on conditional Gaussian distributions (for which the SLDS is a special case) Lauritzen (1992),

which is numerically unstable due to conversion between moment and canonical representations, and Lauritzen and Jensen (2001), which improves stability by avoiding using canonical parameterizations.

Variational Methods Ghahramani and Hinton (1998) used a variational method which approximates the joint distribution $p(h_{1:T}, s_{1:T}|v_{1:T})$ rather than the marginal $p(h_t, s_t|v_{1:T})$—related work is presented in Lee et al. (2004). This is a disadvantage when compared to other methods that directly approximate the marginal. The variational methods are nevertheless potentially attractive since they are able to exploit structural properties of the distribution, such as a factored discrete state-transition. In this article, we concentrate on the case of a small number of states S and hence will not consider variational methods further here.[15]

Sequential Monte Carlo (Particle Filtering) These methods form an approximate implementation of Equation (3), using a sum of delta functions to represent the posterior—see, for example, Doucet et al. (2001). Whilst potentially powerful, these non-analytic methods typically suffer in high-dimensional hidden spaces since they are often based on naive importance sampling, which restricts their practical use. ADF is generally preferential to Particle Filtering, since in ADF the approximation is a mixture of non-trivial distributions, which is better at capturing the variability of the posterior. Rao-Blackwellized Particle Filters (Doucet et al., 2000) are an attempt to alleviate the difficulty of sampling in high-dimensional state spaces by explicitly integrating over the continuous state.

Non-Sequential Monte Carlo

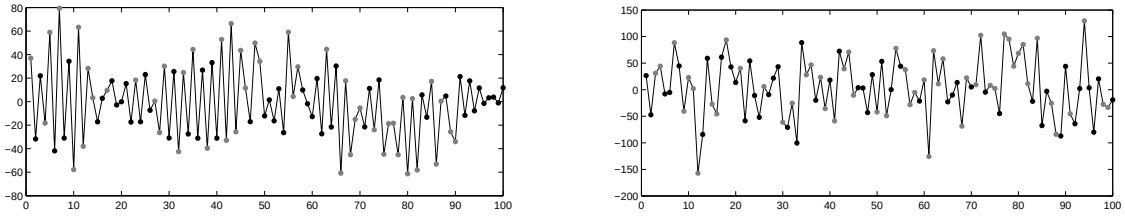
For fixed switches $s_{1:T}$, $p(v_{1:T}|s_{1:T})$ is easily computable since this is just the likelihood of an LDS. This observation raises the possibility of sampling from the posterior $p(s_{1:T}|v_{1:T}) \propto p(v_{1:T}|s_{1:T})p(s_{1:T})$ directly. Many possible sampling methods could be applied in this case, and the most immediate is Gibbs sampling, in which a sample for each t is drawn from $p(s_t|s_{\setminus t}, v_{1:T})$—see Neal (1993) for a general reference and Carter and Kohn (1996) for an application to the SLDS. This procedure may work well in practice provided that the initial setting of $s_{1:T}$ is in a region of high probability mass—otherwise, sampling by such individual coordinate updates may be extremely inefficient.

4. Experiments

Our experiments examine the stability and accuracy of EC against several other methods on long time-series. In addition, we will compare the absolute accuracy of EC as a function of the number of mixture components on a short time-series, where exact inference may be explicitly evaluated.

Testing EC in a problem with a reasonably long temporal sequence, T, is important since numerical stabilities may not be apparent in time-series of just a few time-steps. To do this, we sequentially generate hidden states h_t, s_t and observations v_t from a given model. Then, given only the parameters of the model and the observations (but not any of the hidden states), the task is to infer $p(h_t|s_t, v_{1:T})$ and $p(s_t|v_{1:T})$. Since the exact computation is exponential in T, a formally exact evaluation of the method is infeasible. A simple alternative is to assume that the original sample states $s_{1:T}$ are the 'correct' inferred states, and compare our most probable posterior smoothed

15. Lerner (2002) discusses an approach in the case of a large structured discrete state transition. Related ideas could also be used in EC.



(a) Easy problem (b) Hard problem

Figure 5: SLDS: Throughout, $S = 2$, $V = 1$ (scalar observations), $T = 100$, with zero output bias. $\underline{A}(s) = 0.9999 * \text{orth}(\text{randn}(\text{H},\text{H}))$, $B(s) = \text{randn}(\text{V},\text{H})$, $\bar{v}_t \equiv 0$, $\bar{h}_1 = 10 * \text{randn}(\text{H},1)$, $\bar{h}_{t>1} = 0$, $\Sigma_1^h = I_H$, $p_1 = \text{uniform}$. The figures show typical examples for each of the two problems: (a) Easy problem. $H = 3$, $\Sigma^h(s) = I_H$, $\Sigma^v(s) = 0.1 I_V$, $p(s_{t+1}|s_t) \propto 1_{S\times S} + I_S$. (b) Hard problem. $H = 30$, $\Sigma^v(s) = 30 I_V$,$\Sigma^h(s) = 0.01 I_H$, $p(s_{t+1}|s_t) \propto 1_{S\times S}$.

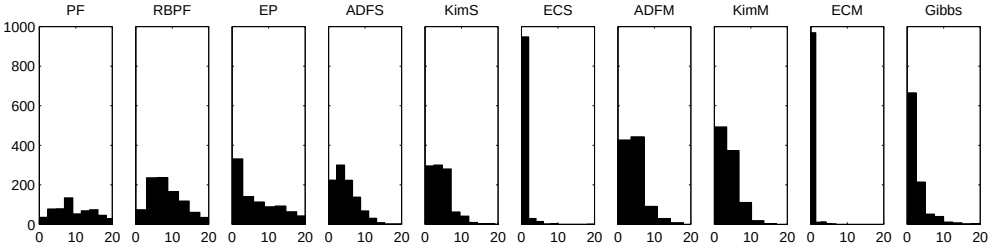


Figure 6: SLDS 'Easy' problem: The number of errors in estimating a binary switch $p(s_t|v_{1:T})$ over a time series of length $T = 100$. Hence 50 errors corresponds to random guessing. Plotted are histograms of the errors over 1000 experiments. The histograms have been cutoff at 20 errors in order to improve visualization. (PF) Particle Filter. (RBPF) Rao-Blackwellized PF. (EP) Expectation Propagation. (ADFS) Assumed Density Filtering using a Single Gaussian. (KimS) Kim's smoother using the results from ADFS. (ECS) Expectation Correction using a Single Gaussian ($I = J = 1$). (ADFM) ADF using a multiple of $I = 4$ Gaussians. (KimM) Kim's smoother using the results from ADFM. (ECM) Expectation Correction using a mixture with $I = J = 4$ components. In Gibbs sampling, we use the initialization from ADFM.

estimates $\arg\max_{s_t} p(s_t|v_{1:T})$ with the assumed correct sample s_t.[16] We look at two sets of experiments, one for the SLDS and one for the aSLDS. In both cases, scalar observations are used so that the complexity of the inference problem can be visually assessed.

16. We could also consider performance measures on the accuracy of $p(h_t|s_t, v_{1:T})$. However, we prefer to look at approximating $\arg\max_{s_t} p(s_t|v_{1:T})$ since the sampled discrete states are likely to correspond to the exact $\arg\max_{s_t} p(s_t|v_{1:T})$. In addition, if the posterior switch distribution is dominated by a single state $s^*_{1:T}$, then provided they are correctly estimated, the model reduces to an LDS, for which inference of the continuous hidden state is trivial.

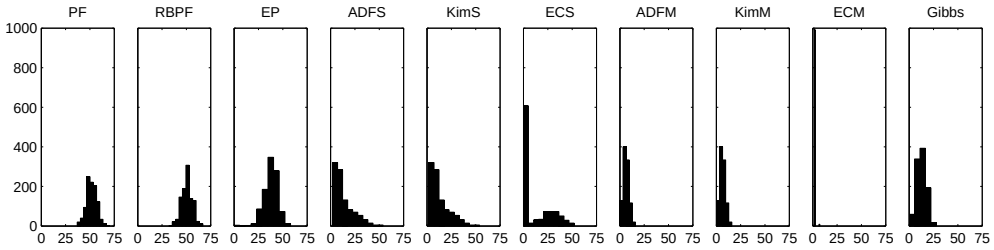


Figure 7: SLDS 'Hard' problem: The number of errors in estimating a binary switch $p(s_t|v_{1:T})$ over a time series of length $T = 100$. Hence 50 errors corresponds to random guessing. Plotted are histograms of the errors over 1000 experiments.

SLDS EXPERIMENTS

We chose experimental conditions that, from the viewpoint of classical signal processing, are difficult, with changes in the switches occurring at a much higher rate than the typical frequencies in the signal. We consider two different toy SLDS experiments : The 'easy' problem corresponds to a low hidden dimension, $H = 3$, with low observation noise; The 'hard' problem corresponds to a high hidden dimension, $H = 30$, and high observation noise. See Figure (5) for details of the experimental setup.

We compared methods using a single Gaussian, and methods using multiple Gaussians, see Figure (6) and Figure (7). For EC we use the mean approximation for the numerical integration of Equation (12). For the Particle Filter 1000 particles were used, with Kitagawa re-sampling (Kitagawa, 1996). For the Rao-Blackwellized Particle Filter (Doucet et al., 2000), 500 particles were used, with Kitagawa re-sampling. We included the Particle Filter merely for a point of comparison with ADF, since they are not designed to approximate the smoothed estimate.

An alternative MCMC procedure is to perform Gibbs sampling of $p(s_{1:T}|v_{1:T})$ using $p(s_t|s_{\setminus t}, v_{1:T}) \propto p(v_{1:T}|s_{1:T})p(s_{1:T})$, where $p(v_{1:T}|s_{1:T})$ is simply the likelihood of an LDS—see for example Carter and Kohn (1996).[17] We initialize the state $s_{1:T}$ by using the most likely states s_t from the filtered results using a Gaussian mixture (ADFM), and then swept forwards in time, sampling from the state $p(s_t|s_{\setminus t}, v_{1:T})$ until the end of the chain. We then reversed direction, sampling from time T back to time 1, and continued repeating this procedure 100 times, with the mean over the last 80 sweeps used as the posterior mean approximation. This procedure is expensive since each sample requires computing the likelihood of an LDS defined on the whole time-series. The procedure therefore scales with GT^2 where G is the number of sweeps over the time series. Despite using a reasonable initialization, Gibbs sampling struggles to improve on the filtered results.

We found that EP was numerically unstable and often struggled to converge. To encourage convergence, we used the damping method in Heskes and Zoeter (2002), performing 20 iterations with a damping factor of 0.5. The disappointing performance of EP is most likely due to conflicts

17. Carter and Kohn (1996) proposed an overly complex procedure for computing the likelihood $p(v_{1:T}|s_{1:T})$. This is simply the likelihood of an LDS (since $s_{1:T}$ are assumed known), and is readily computable using any of the standard procedures in the literature.

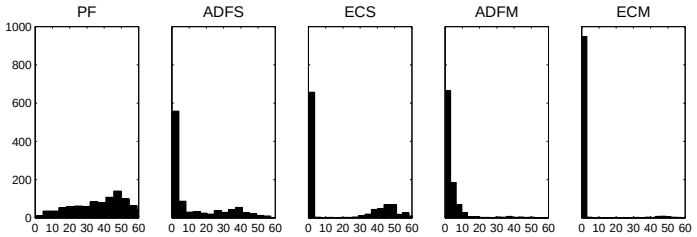


Figure 8: aSLDS: Histogram of the number of errors in estimating a binary switch $p(s_t|v_{1:T})$ over a time series of length $T = 100$. Hence 50 errors corresponds to random guessing. Plotted are histograms of the errors over 1000 experiments. Augmented SLDS results. ADFM used $I = 4$ Gaussians, and ECM used $I = J = 4$ Gaussians. We used 1000 samples to approximate Equation (12).

I	1	4	4	16	16	64	64	256	256
J	1	1	4	1	16	1	64	1	256
error	0.0989	0.0624	0.0365	0.0440	0.0130	0.0440	4.75e-4	0.0440	3.40e-8

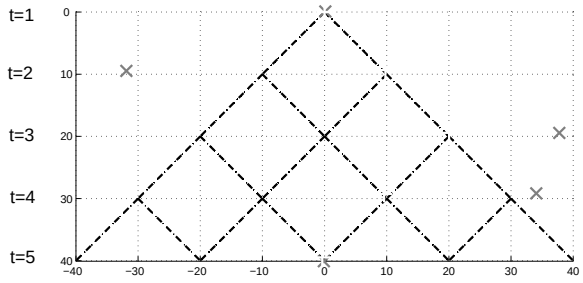
Table 2: Errors in approximating the states for the multi-path problem, see Figure (9). The mean absolute deviation $|p^{ec}(s_t|v_{1:T}) - p^{exact}(s_t|v_{1:T})|$ averaged over the $S = 4$ states of s_t and over the times $t = 1,\ldots,5$, computed for different numbers of mixture components in EC. The mean approximation of Equation (12) is used. The exact computation uses $S^{T-1} = 256$ mixtures.

resulting from numerical instabilities introduced by the frequent conversions between moment and canonical representations.

The various algorithms differ widely in performance, see Figures (6,7). Not surprisingly, the best filtered results are given using ADF, since this is better able to represent the variance in the filtered posterior than the sampling methods. Unlike Kim's method, EC makes good use of the future information to clean up the filtered results considerably. One should bear in mind that both EC, Kim's method and the Gibbs initialization use the same ADF results. These results show that EC may dramatically improve on Kim's method, so that the small amount of extra work in making a numerical approximation of $p(s_t|s_{t+1}, v_{1:T})$, Equation (12), may bring significant benefits.

AUGMENTED SLDS EXPERIMENTS

In Figure (8), we chose a simple two state $S = 2$ transition distribution $p(s_{t+1} = 1|s_t, h_t) = \sigma\left(h_t^\mathsf{T} w(s_t)\right)$, where $\sigma(x) \equiv 1/(1+e^{-x})$. Some care needs to be taken to make a model so for which even exact inference would produce posterior switches close to the sampled switches. If the switch variables s_{t+1} changes wildly (which is possible given the above formula since the hidden state h may have a large projected change if the hidden state changes) essentially no information is left in the signal for any inference method to produce reasonable results. We therefore set $w(s_t)$ to a zero vector except for the first two components, which are independently sampled from a zero mean Gaussian with standard deviation 5. For each of the two switch states, s, we have a transition matrix $A(s)$, which



(a)

	t=1	t=2	t=3	t=4	t=5
s=1	0.495	0.000	0.000	0.000	0.269
s=2	0.495	0.000	0.000	0.000	0.682
s=3	0.005	0.507	0.713	0.509	0.022
s=4	0.005	0.493	0.287	0.491	0.027

(b)

Figure 9: (a) The multi-path problem. The particle starts from $(0,0)$ at time $t = 1$. Subsequently, at each time-point, either the vector $(10,10)$ (corresponding to states $s = 1$ and $s = 3$) or $(-10,10)$ (corresponding to states $s = 2$ and $s = 4$), is added to the hidden dynamics, perturbed by a small amount of noise, $\Sigma^h = 0.1$. The observations are $v = h + \eta^v(s)$. For states $s = 1,2$ the observation noise is small, $\Sigma^v = 0.1I$, but for $s = 3,4$ the noise in the horizontal direction has variance 1000. The visible observations are given by the x'. The true hidden states are given by '+'. (b) The exact smoothed state posteriors $p^{exact}(s_t|v_{1:T})$ computed by enumerating all paths (given by the dashed lines).

we set to be block diagonal. The first 2×2 block is set to $0.9999R_\theta$, where R_θ is a 2×2 rotation matrix with angle θ chosen uniformly from 0 to 1 radians. This means that s_{t+1} is dependent on the first two components of h_t which are rotating at a restricted rate. The remaining $H-2 \times H-2$ block of $A(s)$ is chosen as (using MATLAB notation) $0.9999 * \text{orth}(\text{rand}(\text{H}-2))$, which means a scaled randomly chosen orthogonal matrix. Throughout, $S = 2$, $V = 1$, $H = 30$, $T = 100$, with zero output bias. Using partly MATLAB notation, $B(s) = \text{randn}(\text{V},\text{H})$, $\bar{v}_t \equiv 0$, $\bar{h}_1 = 10 * \text{randn}(\text{H},1)$, $\bar{h}_{t>1} = 0$, $\Sigma_1^h = I_H$, $p_1 = \text{uniform}$. $\Sigma^v = 30I_V$, $\Sigma^h = 0.1I_H$.

We compare EC only against Particle Filters using 1000 particles, since other methods would require specialized and novel implementations. In ADFM, $I = 4$ Gaussians were used, and for ECM, $I = J = 4$ Gaussians were used. Looking at the results in Figure (8), we see that EC performs well, with some improvement in using the mixture representation $I, J = 4$ over a single Gaussian $I = J = 1$. The Particle Filter most likely failed since the hidden dimension is too high to be explored well with only 1000 particles.

EFFECT OF USING MIXTURES

Our claim is that EC should cope in situations where the smoothed posterior $p(h_t|s_t, v_{1:T})$ is multi-modal and, consequently, cannot be well represented by a single Gaussian.[18] We therefore constructed an SLDS which exhibits multi-modality to see the effect of using EC with both I and J greater than 1. The 'multi-path' scenario is described in Figure (9), where a particle traces a path through a two dimensional space. A small number of time-steps was chosen so that the exact $p(s_t|v_{1:T})$ can be computed by direct enumeration. The observation of the particle is at times extremely noisy in the horizontal direction. This induces multi-modality of $p(h_t|s_t, v_{1:T})$ since there

18. This should not be confused with the multi-modality of $p(h_t|v_{1:T}) = \sum_{s_t} p(h_t|s_t, v_{1:T})p(s_t|v_{1:T})$.

are several paths that might plausibly have been taken to give rise to the observations. The accuracy with which EC predicts the exact smoothed posterior is given in Table (2). For this problem we see that both the number of Forward (I) and Backward components (J) affects the accuracy of the approximation, generally with improved accuracy as the number of mixture components increases. For a 'perfect' approximation method, one would expect that when $I = J = S^{T-1} = 256$, then the approximation should become exact. The small error for this case in Table (2) may arise for several reasons: the extra independence assumption used in EC, or the simple mean approximation used to compute Equation (12), or numerical roundoff. However, at least in this case, the effect of these assumptions on the performance is very small.

5. Discussion

Expectation Correction is a novel form of Backward Pass which makes less approximations than the widely used approach from Kim (1994). In Kim's method, potentially important future information channeled through the continuous hidden variables is lost. EC, along with Kim's method, makes the additional assumption $p(h_{t+1}|s_t, s_{t+1}, v_{1:T}) \approx p(h_{t+1}|s_{t+1}, v_{1:T})$. However, our experience is that this assumption is rather mild, since the state of h_{t+1} will be most heavily influenced by its immediate parent s_{t+1}.

Our approximation is based on the idea that, although exact inference will consist of an exponentially large number of mixture components, due to the forgetting which commonly occurs in Markovian models, a finite number of mixture components may provide a reasonable approximation. In tracking situations where the visible information is (temporarily) not enough to specify accurately the hidden state, then representing the posterior $p(h_t|s_t, v_{1:T})$ using a mixture of Gaussians may improve results significantly. Clearly, in systems with very long correlation times our method may require too many mixture components to produce a satisfactory result, although we are unaware of other techniques that would be able to cope well in that case.

We hope that the straightforward ideas presented here may help facilitate the practical application of dynamic hybrid networks to machine learning and related areas. Whilst models with Gaussian emission distributions such as the SLDS are widespread, the extension of this method to non-Gaussian emissions $p(v_t|h_t, s_t)$ would clearly be of considerable interest.

Software for Expectation Correction for this augmented class of Switching Linear Gaussian models is available from `www.idiap.ch/~barber`.

Acknowledgments

I would like to thank Onno Zoeter and Tom Heskes for kindly providing their Expectation Propagation code, Silvia Chiappa for helpful discussions, and Bertrand Mesot for many discussions, help with the simulations and for suggesting the relationship between the partial factorization and independence viewpoints of EC. I would also like to thank the reviewers for their many helpful comments and suggestions.

Algorithm 4 LDS Forward Pass. Compute the filtered posteriors $p(h_t|v_{1:t}) \equiv \mathcal{N}(f_t, F_t)$ for a LDS with parameters $\theta_t = A, B, \Sigma^h, \Sigma^v, \bar{h}, \bar{v}$, for $t > 1$. At time $t = 1$, we use parameters $\theta_1 = A, B, \Sigma, \Sigma^v, \mu, \bar{v}$, where Σ and μ are the prior covariance and mean of h. The log-likelihood $L = \log p(v_{1:T})$ is also returned.

$F_0 \leftarrow 0$, $f_0 \leftarrow 0$, $L \leftarrow 0$
for $t \leftarrow 1, T$ **do**
  $\{f_t, F_t, p_t\} = \text{LDSFORWARD}(f_{t-1}, F_{t-1}, v_t; \theta_t)$
  $L \leftarrow L + \log p_t$
end for
function LDSFORWARD$(f, F, v; \theta)$
  Compute joint $p(h_t, v_t|v_{1:t-1})$:
  $\mu_h \leftarrow Af + \bar{h}$, $\mu_v \leftarrow B\mu_h + \bar{v}$
  $\Sigma_{hh} \leftarrow AFA^\mathsf{T} + \Sigma^h$, $\Sigma_{vv} \leftarrow B\Sigma_{hh}B^\mathsf{T} + \Sigma^v$, $\Sigma_{vh} \leftarrow B\Sigma_{hh}$
  Find $p(h_t|v_{1:t})$ by conditioning:
  $f' \leftarrow \mu_h + \Sigma_{vh}^\mathsf{T}\Sigma_{vv}^{-1}(v - \mu_v)$, $F' \leftarrow \Sigma_{hh} - \Sigma_{vh}^\mathsf{T}\Sigma_{vv}^{-1}\Sigma_{vh}$
  Compute $p(v_t|v_{1:t-1})$:
  $p' \leftarrow \exp\left(-\frac{1}{2}(v - \mu_v)^\mathsf{T}\Sigma_{vv}^{-1}(v - \mu_v)\right) / \sqrt{\det 2\pi\Sigma_{vv}}$
  return f', F', p'
end function

Appendix A. Inference in the LDS

The LDS is defined by Equations (1,2) in the case of a single switch $S = 1$. The LDS Forward and Backward passes define the important functions LDSFORWARD and LDSBACKWARD, which we shall make use of for inference in the aSLDS.

FORWARD PASS (FILTERING)

The filtered posterior $p(h_t|v_{1:t})$ is a Gaussian which we parameterize with mean f_t and covariance F_t. These parameters can be updated recursively using $p(h_t|v_{1:t}) \propto p(h_t, v_t|v_{1:t-1})$, where the joint distribution $p(h_t, v_t|v_{1:t-1})$ has statistics (see Appendix (B))

$$\mu_h = Af_{t-1} + \bar{h}, \quad \mu_v = B\mu_h + \bar{v}$$

$$\Sigma_{hh} = AF_{t-1}A^\mathsf{T} + \Sigma^h, \quad \Sigma_{vv} = B\Sigma_{hh}B^\mathsf{T} + \Sigma^v, \quad \Sigma_{vh} = B\Sigma_{hh}.$$

We may then find $p(h_t|v_{1:t})$ by conditioning $p(h_t, v_t|v_{1:t-1})$ on v_t, see Appendix (C). This gives rise to Algorithm (4).

BACKWARD PASS

The smoothed posterior $p(h_t|v_{1:T}) \equiv \mathcal{N}(g_t, G_t)$ can be computed recursively using:

$$p(h_t|v_{1:T}) = \int_{h_{t+1}} p(h_t|h_{t+1}, v_{1:T})p(h_{t+1}|v_{1:T}) = \int_{h_{t+1}} p(h_t|h_{t+1}, v_{1:t})p(h_{t+1}|v_{1:T})$$

where $p(h_t|h_{t+1}, v_{1:t})$ may be obtained from the joint distribution

$$p(h_t, h_{t+1}|v_{1:t}) = p(h_{t+1}|h_t)p(h_t|v_{1:t}) \tag{19}$$

Algorithm 5 LDS Backward Pass. Compute the smoothed posteriors $p(h_t|v_{1:T})$. This requires the filtered results from Algorithm (4).

$G_T \leftarrow F_T$, $g_T \leftarrow f_T$
for $t \leftarrow T-1, 1$ **do**
  $\{g_t, G_t\}$ = LDSBACKWARD$(g_{t+1}, G_{t+1}, f_t, F_t; \theta_{t+1})$
end for
function LDSBACKWARD$(g, G, f, F; \theta)$
  $\mu_h \leftarrow Af + \bar{h}$, $\quad \Sigma_{h'h'} \leftarrow AFA^\mathsf{T} + \Sigma^h$, $\quad \Sigma_{h'h} \leftarrow AF$
  $\overleftarrow{\Sigma} \leftarrow F_t - \Sigma_{h'h}^\mathsf{T}\Sigma_{h'h'}^{-1}\Sigma_{h'h}$, $\quad \overleftarrow{A} \leftarrow \Sigma_{h'h}^\mathsf{T}\Sigma_{h'h'}^{-1}$, $\quad \overleftarrow{m} \leftarrow f - \overleftarrow{A}\mu_h$
  $g' \leftarrow \overleftarrow{A}g + \overleftarrow{m}$, $\quad G' \leftarrow \overleftarrow{A}G\overleftarrow{A}^\mathsf{T} + \overleftarrow{\Sigma}$
  return g', G'
end function

which itself can be obtained by forward propagation from $p(h_t|v_{1:t})$. Conditioning Equation (19) to find $p(h_t|h_{t+1}, v_{1:t})$ effectively reverses the dynamics,

$$h_t = \overleftarrow{A_t}h_{t+1} + \overleftarrow{\eta_t}$$

where $\overleftarrow{A_t}$ and $\overleftarrow{\eta}_t \sim \mathcal{N}(\overleftarrow{m_t}, \overleftarrow{\Sigma_t})$ are found using the conditioned Gaussian results in Appendix (C)—these are explicitly given in Algorithm (5). Then averaging the reversed dynamics over $p(h_{t+1}|v_{1:T})$ we find that $p(h_t|v_{1:T})$ is a Gaussian with statistics

$$g_t = \overleftarrow{A_t}g_{t+1} + \overleftarrow{m_t}, \quad G_t = \overleftarrow{A_t}G_{t+1}\overleftarrow{A_t}^\mathsf{T} + \overleftarrow{\Sigma_t}.$$

This Backward Pass is given in Algorithm (5). For parameter learning of the A matrix, the smoothed statistic $\left\langle h_t h_{t+1}^\mathsf{T}\right\rangle$ is required. Using the above formulation, this is given by $\overleftarrow{A_t}G_{t+1} + \langle h_t\rangle\left\langle h_{t+1}^\mathsf{T}\right\rangle$. This is much simpler than the standard expressions cited in Shumway and Stoffer (2000) and Roweis and Ghahramani (1999).

Appendix B. Gaussian Propagation

Let y be linearly related to x through $y = Mx + \eta$, where $\eta \sim \mathcal{N}(\mu, \Sigma)$, and $x \sim \mathcal{N}(\mu_x, \Sigma_x)$. Then $p(y) = \int_x p(y|x)p(x)$ is a Gaussian with mean $M\mu_x + \mu$ and covariance $M\Sigma_x M^\mathsf{T} + \Sigma$.

Appendix C. Gaussian Conditioning

For a joint Gaussian distribution over the vectors x and y with means μ_x, μ_y and covariance elements Σ_{xx},Σ_{xy},Σ_{yy}, the conditional $p(x|y)$ is a Gaussian with mean $\mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y)$ and covariance $\Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}$.

Appendix D. Collapsing Gaussians

The user may provide any algorithm of their choice for collapsing a set of Gaussians to a smaller set of Gaussians (Titterington et al., 1985). Here, to be explicit, we present a simple one which is fast, but has the disadvantage that no spatial information about the mixture is used.

First, we describe how to collapse a mixture to a *single* Gaussian: We may collapse a mixture of Gaussians $p(x) = \sum_i p_i \mathcal{N}(x|\mu_i, \Sigma_i)$ to a single Gaussian with mean $\sum_i p_i \mu_i$ and covariance $\sum_i p_i \left(\Sigma_i + \mu_i \mu_i^\mathsf{T}\right) - \mu\mu^\mathsf{T}$.

To collapse a mixture to a K-component *mixture* we retain the $K-1$ Gaussians with the largest mixture weights—the remaining $N-K$ Gaussians are simply merged to a single Gaussian using the above method. The alternative of recursively merging the two Gaussians with the lowest mixture weights gave similar experimental performance.

More sophisticated methods which retain some spatial information would clearly be potentially useful. The method presented in Lerner et al. (2000) is a suitable approach which considers removing Gaussians which are spatially similar (and not just low-weight components), thereby retaining diversity over the possible solutions.

Appendix E. The Discrete-Continuous Factorization Viewpoint

An alternative viewpoint is to proceed analogously to the Rauch-Tung-Striebel correction method for the LDS (Grewal and Andrews, 1993):

$$\begin{aligned}
p(h_t, s_t|v_{1:T}) &= \sum_{s_{t+1}} \int_{h_{t+1}} p(s_t, h_t, h_{t+1}, s_{t+1}|v_{1:T}) \\
&= \sum_{s_{t+1}} p(s_{t+1}|v_{1:T}) \int_{h_{t+1}} p(h_t, s_t|h_{t+1}, s_{t+1}, v_{1:t}) p(h_{t+1}|s_{t+1}, v_{1:T}) \\
&= \sum_{s_{t+1}} p(s_{t+1}|v_{1:T}) \left\langle p(h_t|h_{t+1}, s_{t+1}, s_t, v_{1:t}) p(s_t|h_{t+1}, s_{t+1}, v_{1:t}) \right\rangle \\
&\approx \sum_{s_{t+1}} p(s_{t+1}|v_{1:T}) \left\langle p(h_t|h_{t+1}, s_{t+1}, s_t, v_{1:t}) \right\rangle \underbrace{\left\langle p(s_t|s_{t+1}, v_{1:T}) \right\rangle}_{p(s_t|s_{t+1}, v_{1:T})}
\end{aligned} \tag{20}$$

where angled brackets $\langle \cdot \rangle$ denote averages with respect to $p(h_{t+1}|s_{t+1}, v_{1:T})$. Whilst the factorized approximation in Equation (20) may seem severe, by comparing Equations (20) and (10) we see that it is equivalent to the apparently milder assumption $p(h_{t+1}|s_t, s_{t+1}, v_{1:T}) \approx p(h_{t+1}|s_{t+1}, v_{1:T})$. Hence this factorized approximation is equivalent to the 'standard' EC approach in which the dependency on s_t is dropped.

References

D. L. Alspach and H. W. Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448, 1972.

Y. Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking : Principles, Techniques and Software*. Artech House, Norwood, MA, 1998.

X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence—UAI 1998*, pages 33–42. Morgan Kaufmann, 1998.

C. Carter and R. Kohn. Markov chain Monte Carlo in conditionally Gaussian state space models. *Biometrika*, 83:589–601, 1996.

A. T. Cemgil, B. Kappen, and D. Barber. A Generative Model for Music Transcription. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):679 – 694, 2006.

S. Chib and M. Dueker. Non-Markovian regime switching with endogenous states and time-varying state strengths. *Econometric Society 2004 North American Summer Meetings 600*, 2004.

A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. *Uncertainty in Artificial Intelligence*, 2000.

A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996, 1998.

M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice*. Prentice-Hall, 1993.

T. Heskes and O. Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In A. Darwiche and N. Friedman, editors, *Uncertainty in Artificial Intelligence*, pages 216–223, 2002.

T. Jaakkola and M. Jordan. A variational approach to Bayesian logistic regression problems and their extensions. In *Artificial Intelligence and Statistics*, 1996.

M. I. Jordan. *Learning in Graphical Models*. MIT Press, 1998.

S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.

C-J. Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60:1–22, 1994.

C-J. Kim and C. R. Nelson. *State-Space Models with Regime Switching*. MIT Press, 1999.

G. Kitagawa. The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4):605–623, 1994.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

S. Lauritzen and F. Jensen. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11:191–203, 2001.

S. L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108, 1992.

L. J. Lee, H. Attias, Li Deng, and P. Fieguth. A multimodal variational approach to learning and inference in switching state space models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP 04)*, volume 5, pages 505–8, 2004.

U. Lerner, R. Parr, D. Koller, and G. Biswas. Bayesian fault detection and diagnosis in dynamic systems. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AIII-00)*, pages 531–537, 2000.

U. N. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems*. PhD thesis, Stanford University, 2002.

B. Mesot and D. Barber. Switching linear dynamical systems for noise robust speech recognition. IDIAP-RR 08, 2006.

T. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, MIT Media Lab, 2001.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993.

P. Park and T. Kailath. New square-root smoothing algorithms. *IEEE Transactions on Automatic Control*, 41:727–732, 1996.

V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *Advances in Neural Information Processing systems (NIPS 13)*, pages 981–987, 2001.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

H. E. Rauch, G. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *American Institute of Aeronautics and Astronautics Journal (AIAAJ)*, 3(8):1445–1450, 1965.

S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345, 1999.

R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications*. Springer, 2000.

E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 605–612, 2003.

D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, 1985.

H. Tong. *Nonlinear Time Series Analysis: A Dynamical Systems Approach*. Oxford Univ. Press, 1990.

M. Verhaegen and P. Van Dooren. Numerical aspects of different Kalman filter implementations. *IEEE Transactions of Automatic Control*, 31(10):907–917, 1986.

M. West and J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer, 1999.

O. Zoeter. *Monitoring Non-Linear and Switching Dynamical Systems*. PhD thesis, Radboud University Nijmegen, 2005.

Journal of Machine Learning Research 7 (2006) 2541-2563 Submitted 4/06; Revised 9/06; Published 11/06

On Model Selection Consistency of Lasso

Peng Zhao PENGZHAO@STAT.BERKELEY.EDU
Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics
University of California, Berkeley
367 Evans Hall Berkeley, CA 94720-3860, USA

Editor: David Madigan

Abstract

Sparsity or parsimony of statistical models is crucial for their proper interpretations, as in sciences and social sciences. Model selection is a commonly used method to find such models, but usually involves a computationally heavy combinatorial search. Lasso (Tibshirani, 1996) is now being used as a computationally feasible alternative to model selection. Therefore it is important to study Lasso for model selection purposes.

In this paper, we prove that a single condition, which we call the Irrepresentable Condition, is almost necessary and sufficient for Lasso to select the true model both in the classical fixed p setting and in the large p setting as the sample size n gets large. Based on these results, sufficient conditions that are verifiable in practice are given to relate to previous works and help applications of Lasso for feature selection and sparse representation.

This Irrepresentable Condition, which depends mainly on the covariance of the predictor variables, states that Lasso selects the true model consistently if and (almost) only if the predictors that are not in the true model are "irrepresentable" (in a sense to be clarified) by predictors that are in the true model. Furthermore, simulations are carried out to provide insights and understanding of this result.

Keywords: Lasso, regularization, sparsity, model selection, consistency

1. Introduction

A vastly popular and successful approach in statistical modeling is to use regularization penalties in model fitting (Hoerl and Kennard, 1970). By jointly minimizing the empirical error and penalty, one seeks a model that not only fits well and is also "simple" to avoid large variation which occurs in estimating complex models. Lasso (Tibshirani, 1996) is a successful idea that falls into this category. Its popularity is largely because the regularization resulting from Lasso's L_1 penalty leads to sparse solutions, that is, there are few nonzero estimates (among all possible choices). Sparse models are more interpretable and often preferred in the sciences and social sciences. However, obtaining such models through classical model selection methods usually involves heavy combinatorial search. Lasso, of which the entire regularization path can be computed in the complexity of one linear regression (Efron et al., 2004; Osborne et al., 2000b), provides a computationally feasible way for model selection (also see, for example, Zhao and Yu, 2004; Rosset, 2004). However, in order to use Lasso for model selection, it is necessary to assess how well the sparse model given by Lasso relates to the true model. We make this assessment by investigating Lasso's model selection consistency

under linear models, that is, when given a large amount of data under what conditions Lasso does and does not choose the true model.

Assume our data is generated by a linear regression model

$$Y_n = \mathbf{X_n}\beta^n + \varepsilon_\mathbf{n}.$$

where $\varepsilon_\mathbf{n} = (\varepsilon_1, ..., \varepsilon_n)^T$ is a vector of i.i.d. random variables with mean 0 and variance σ^2. Y_n is an $n \times 1$ response and $\mathbf{X_n} = (X_1^n, ..., X_p^n) = ((x_1^n)^T, ..., (x_n^n)^T)^T$ is the $n \times p$ design matrix where X_i^n is its ith column (ith predictor) and x_j^n is its jth row (jth sample). β^n is the vector of model coefficients. The model is assumed to be "sparse", that is, some of the regression coefficients β^n are exactly zero corresponding to predictors that are irrelevant to the response. Unlike classical fixed p settings, the data and model parameters β are indexed by n to allow them to change as n grows.

The Lasso estimates $\hat{\beta}^n = (\hat{\beta}_1^n, ..., \hat{\beta}_j^n, ...)^T$ are defined by

$$\hat{\beta}^n(\lambda) = \arg\min_\beta \|Y_n - \mathbf{X_n}\beta\|_2^2 + \lambda\|\beta\|_1, \tag{1}$$

where $\|\cdot\|_1$ stands for the L_1 norm of a vector which equals the sum of absolute values of the vector's entries.

The parameter $\lambda \geq 0$ controls the amount of regularization applied to the estimate. Setting $\lambda = 0$ reverses the Lasso problem to Ordinary Least Squares which minimizes the unregularized empirical loss. On the other hand, a very large λ will completely shrink $\hat{\beta}^n$ to 0 thus leading to the empty or null model. In general, moderate values of λ will cause shrinkage of the solutions towards 0, and some coefficients may end up being exactly 0.

Under some regularity conditions on the design, Knight and Fu (2000) have shown estimation consistency for Lasso for fixed p and fixed β^n (i.e., p and β^n are independent of n) as $n \to \infty$. In particular, they have shown that $\hat{\beta}^n(\lambda_n) \to_p \beta$ and asymptotic normality of the estimates provided that $\lambda_n = o(n)$. In addition, it is shown in the work that for $\lambda_n \propto n^{\frac{1}{2}}$ (on the same order of $n^{\frac{1}{2}}$), as $n \to \infty$ there is a non-vanishing positive probability for lasso to select the true model.

On the model selection consistency front, Meinshausen and Buhlmann (2006) have shown that under a set of conditions, Lasso is consistent in estimating the dependency between Gaussian variables even when the number of variables p grows faster than n. Addressing a slightly different but closely related problem, Leng et al. (2004) have shown that for a fixed p and orthogonal designs, the Lasso estimate that is optimal in terms of parameter estimation does not give consistent model selection. Furthermore, Osborne et al. (1998), in their work of using Lasso for knot selection for regression splines, noted that Lasso tend to pick up knots in close proximity to one another. In general, as we will show, if an irrelevant predictor is highly correlated with the predictors in the true model, Lasso may not be able to distinguish it from the true predictors with any amount of data and any amount of regularization.

Since using the Lasso estimate involves choosing the appropriate amount of regularization, to study the model selection consistency of the Lasso, we consider two problems: whether there exists a deterministic amount of regularization that gives consistent selection; or, for each random realization whether there exists a correct amount of regularization that selects the true model. Our main result shows there exists an **Irrepresentable Condition** that, except for a minor technicality, is almost necessary and sufficient for both types of consistency. Based on this condition, we give sufficient conditions that are verifiable in practice. In particular, in one example our condition coincides with the "Coherence" condition in Donoho et al. (2004) where the L_2 distance between the Lasso estimate and true model is studied in a non-asymptotic setting.

After we had obtained our almost necessary and sufficient condition result, it was brought to our attention of an independent result in Meinshausen and Buhlmann (2006) where a similar condition to the Irrepresentable Condition was obtained to prove a model selection consistency result for Gaussian graphical model selection using the Lasso. Our result is for linear models (with fixed p and p growing with n) and it could accommodate non-Gaussian errors and non-Gaussian designs. Our analytical approach is direct and we thoroughly explain through special cases and simulations the meaning of this condition in various cases. We also make connections to previous theoretical studies and simulations on Lasso (e.g., Donoho et al., 2004; Zou et al., 2004; Tibshirani, 1996).

The rest of the paper is organized as follows. In Section 2, we describe our main result—the Irrepresentable Condition for Lasso to achieve consistent selection and prove that it is almost necessary and sufficient. We then elaborate on the condition by extending to other sufficient conditions that are more intuitive and verifiable to relate to previous theoretical and simulation studies of Lasso. Sections 3 contains simulation results to illustrate our result and to build heuristic sense of how strong the condition is. To conclude, Section 4 compares Lasso with thresholding and discusses alternatives and possible modifications of Lasso to achieve selection consistency when Irrepresentable Condition fails.

2. Model Selection Consistency and Irrepresentable Conditions

An estimate which is consistent in term of parameter estimation does not necessarily consistently select the correct model (or even attempt to do so) where the reverse is also true. The former requires

$$\hat{\beta}^n - \beta^n \to_p 0, \text{ as } n \to \infty$$

while the latter requires

$$P(\{i: \ \hat{\beta}_i^n \neq 0\} = \{i: \ \beta_i^n \neq 0\}) \to 1, \text{ as } n \to \infty.$$

In general, we desire our estimate to have both consistencies. However, to separate the selection aspect of the consistency from the parameter estimation aspect, we make the following definitions about Sign Consistency that does not assume the estimates to be estimation consistent.

Definition 1 An estimate $\hat{\beta}^n$ is equal in sign with the true model β^n which is written

$$\hat{\beta}^n =_s \beta^n$$

if and only if

$$\text{sign}(\hat{\beta}^n) = \text{sign}(\beta^n)$$

where $\text{sign}(\cdot)$ maps positive entry to 1, negative entry to -1 and zero to zero, that is, $\hat{\beta}^n$ matches the zeros and signs of β.

Sign consistency is stronger than the usual selection consistency which only requires the zeros to be matched, but not the signs. The reason for using sign consistency is technical. It is needed for proving the necessity of the Irrepresentable Condition (to be defined) to avoid dealing with situations where a model is estimated with matching zeros but reversed signs. We also argue that an estimated model with reversed signs can be misleading and hardly qualifies as a correctly selected model.

Now we define two kinds of sign consistencies for Lasso depending on how the amount of regularization is determined.

Definition 2 Lasso is **Strongly Sign Consistent** if there exists $\lambda_n = f(n)$, that is, a function of n and independent of Y_n or $\mathbf{X_n}$ such that

$$\lim_{n\to\infty} P(\hat{\beta}^n(\lambda_n) =_s \beta^n) = 1.$$

Definition 3 The Lasso is **General Sign Consistent** if

$$\lim_{n\to\infty} P(\exists\lambda \geq 0, \hat{\beta}^n(\lambda) =_s \beta^n) = 1.$$

Strong Sign Consistency implies one can use a preselected λ to achieve consistent model selection via Lasso. General Sign Consistency means for a random realization there exists a correct amount of regularization that selects the true model. Obviously, strong sign consistency implies general sign consistency. Surprisingly, as implied by our results, the two kinds of sign consistencies are almost equivalent to one condition. To define this condition we need the following notations on the design.

Without loss of generality, assume $\beta^n = (\beta_1^n, ..., \beta_q^n, \beta_{q+1}^n, ... \beta_p^n)^T$ where $\beta_j^n \neq 0$ for $j = 1, .., q$ and $\beta_j^n = 0$ for $j = q+1, ..., p$. Let $\beta_{(1)}^n = (\beta_1^n, ..., \beta_q^n)^T$ and $\beta_{(2)}^n = (\beta_{q+1}^n, ..., \beta_p^n)$. Now write $\mathbf{X_n}(1)$ and $\mathbf{X_n}(2)$ as the first q and last $p-q$ columns of $\mathbf{X_n}$ respectively and let $C^n = \frac{1}{n}\mathbf{X_n}^T\mathbf{X_n}$. By setting $C_{11}^n = \frac{1}{n}\mathbf{X_n}(1)'\mathbf{X_n}(1)$, $C_{22}^n = \frac{1}{n}\mathbf{X_n}(2)'\mathbf{X_n}(2)$, $C_{12}^n = \frac{1}{n}\mathbf{X_n}(\mathbf{1})'\mathbf{X_n}(\mathbf{2})$ and $C_{21}^n = \frac{1}{n}\mathbf{X_n}(2)'\mathbf{X_n}(1)$. C^n can then be expressed in a block-wise form as follows:

$$C^n = \begin{pmatrix} C_{11}^n & C_{12}^n \\ C_{21}^n & C_{22}^n \end{pmatrix}.$$

Assuming C_{11}^n is invertible, we define the following Irrepresentable Conditions
Strong Irrepresentable Condition. There exists a positive constant vector η

$$|C_{21}^n (C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)| \leq \mathbf{1} - \eta,$$

where $\mathbf{1}$ is a $p-q$ by 1 vector of 1's and the inequality holds element-wise.
Weak Irrepresentable Condition.

$$|C_{21}^n (C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)| < \mathbf{1},$$

where the inequality holds element-wise.

Weak Irrepresentable Condition is slightly weaker than Strong Irrepresentable Condition. C^n can converge in ways that entries of $|C_{21}^n (C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)|$ approach 1 from the below so that Weak Condition holds but the strict inequality fails in the limit. For a fixed p and $\beta^n = \beta$, the distinction disappears for random designs when, for example, x_i^n's are i.i.d. realizations with covariance matrix C, since then the two conditions are equivalent to $|C_{21}(C_{11})^{-1}\text{sign}(\beta(1))| < \mathbf{1}$ almost surely.

The Irrepresentable Conditions closely resembles a regularization constraint on the regression coefficients of the irrelevant covariates ($\mathbf{X_n}(2)$) on the relevant covariates ($\mathbf{X_n}(1)$). In particular, when signs of the true β are unknown, for the Irrepresentable Condition to hold for all possible signs, we need the L_1 norms of the regression coefficients to be smaller than 1. To see this, recall for (2) to hold for all possible sign$(\beta(1))$, we need

$$|((\mathbf{X_n}(1)^T\mathbf{X_n}(1))^{-1}\mathbf{X_n}(1)^T\mathbf{X_n}(2)| = |(C_{11}^n)^{-1}C_{12}^n| < \mathbf{1} - \eta, \tag{2}$$

that is, the total amount of an irrelevant covariate represented by the covariates in the true model is not to reach 1 (therefore the name "irrepresentable").

As a preparatory result, the following proposition puts a lower bound on the probability of Lasso picking the true model which quantitatively relates the probability of Lasso selecting the correct model and how well Strong Irrepresentable Condition holds:

Proposition 1. Assume Strong Irrepresentable Condition holds with a constant $\eta > 0$ then

$$P(\hat{\beta}^n(\lambda_n)) =_s \beta^n) \geq P(A_n \cap B_n)$$

for

$$\begin{aligned} A_n &= \{|(C^n_{11})^{-1}W^n(1)| < \sqrt{n}(|\beta^n_{(1)}| - \frac{\lambda_n}{2n}|(C^{11}_n)^{-1}\text{sign}(\beta^n_{(1)})|)\}, \\ B_n &= \{|C^n_{21}(C^n_{11})^{-1}W^n(1) - W^n(2)| \leq \frac{\lambda_n}{2\sqrt{n}}\eta\}, \end{aligned}$$

where

$$W^n(1) = \frac{1}{\sqrt{n}}\mathbf{X_n}(1)'\varepsilon_n \text{ and } \frac{1}{\sqrt{n}}W^n(2) = \mathbf{X_n}(2)'\varepsilon_n.$$

It can be argued (see the proof of Proposition 1 in the appendix) that A_n implies the signs of of those of $\beta^n_{(1)}$ are estimated correctly. And given A_n, B_n further imply $\hat{\beta}^n_{(2)}$ are shrunk to zero. The regularization parameter λ_n trades off the size of these two events. Smaller λ_n leads to larger A_n but smaller B_n which makes it likely to have Lasso pick more irrelevant variables. On the other hand, larger constant η always leads to larger B_n and have no impact on A_n. So when Strong Irrepresentable Condition holds with a larger constant η, it is easier for Lasso to pick up the true model. This is quantitatively illustrated in Simulation 3.2.

Our main results relate Strong and Weak Irrepresentable Conditions with strong and general sign consistency. We describe the results for small q and p case next followed by results for large q and p in Section 2.2. Then, analysis and sufficient conditions are given in Section 2.3 to achieve a better understanding of the Irrepresentable Conditions and relate to previous works.

2.1 Model Selection Consistency for Small q and p

In this section, we work under the classical setting where q, p and β^n are all fixed as $n \to \infty$. In this setting, it is natural to assume the following regularity conditions:

$$C^n \to C, \text{ as } n \to \infty. \tag{3}$$

where C is a positive definite matrix. And,

$$\frac{1}{n}\max_{1\leq i\leq n}((x^n_i)^T x^n_i) \to 0, \text{ as } n \to \infty. \tag{4}$$

In practice, the covariates are usually scaled so that the diagonal elements of C^n are all 1's. The convergence in (3) and (4) are deterministic. However, the results in this is section also holds quite generally for random designs. Specifically, in the case of a random design, X can be conditioned on and the asymptotic results still apply if the probability of the set where (3) and (4) hold is 1. In general, (3) and (4) are weak in the sense that if one assumes x_i are i.i.d. with finite second moments then $C = E((x^n_i)^T x^n_i)$, $\frac{1}{n}\mathbf{X_n}^T\mathbf{X_n} \to_{a.s.} C$ and $\max_{1\leq i\leq n} x_i^T x_i = o_p(n)$, thus (3) and (4) hold naturally.

Under these conditions we have the following result.

Theorem 1. For fixed q, p and $\beta^n = \beta$, under regularity conditions (3) and (4), Lasso is strongly sign consistent *if* Strong Irrepresentable Condition holds. That is, when Strong Irrepresentable Condition holds, for $\forall \lambda_n$ that satisfies $\lambda_n/n \to 0$ and $\lambda_n/n^{\frac{1+c}{2}} \to \infty$ with $0 \le c < 1$, we have

$$P(\hat{\beta}^n(\lambda_n) =_s \beta^n) = 1 - o(e^{-n^c}).$$

A proof of Theorem 1 can be found in the appendix.

Theorem 1 states that, if Strong Irrepresentable Condition holds, then the probability of Lasso selecting the true model approaches 1 at an exponential rate while only the finite second moment of the noise terms is assumed. In addition, from Knight and Fu (2000) we know that for $\lambda_n = o(n)$ Lasso also has consistent estimation and asymptotic normality. Therefore Strong Irrepresentable Condition allows for consistent model selection and parameter estimation simultaneously. On the other hand, Theorem 2 shows that Weak Irrepresentable Condition is also necessary even for the weaker general sign consistency.

Theorem 2. For fixed p, q and $\beta_n = \beta$, under regularity conditions (3) and (4), Lasso is general sign consistent *only if* there exists N so that Weak Irrepresentable Condition holds for $n > N$.

A proof of Theorem 2 can be found in the appendix.

Therefore, Strong Irrepresentable Condition implies strong sign consistency implies general sign consistency implies Weak Irrepresentable Condition. So except for the technical difference between the two conditions, Irrepresentable Condition is almost necessary and sufficient for both strong sign consistency and general sign consistency.

Furthermore, under additional regularity conditions on the noise terms ε_i^n, this "small" p result can be extended to the "large" p case. That is, when p also tends to infinity "not too fast" as n tends to infinity, we show that Strong Irrepresentable Condition, again, implies Strong Sign Consistency for Lasso.

2.2 Model Selection Consistency for Large p and q

In the large p and q case, we allow the dimension of the designs C^n and model parameters β_n grow as n grows, that is, $p = p_n$ and $q = q_n$ are allowed to grow with n. Consequently, the assumptions and regularity conditions in Section 2.1 becomes inappropriate as C^n do not converge and β^n may change as n grows. Thus we need to control the size of the smallest entry of $\beta^n_{(1)}$, bound the eigenvalues of C^n_{11} and have the design scale properly. Specifically, we assume:

There exists $0 \le c_1 < c_2 \le 1$ and $M_1, M_2, M_3, M_4 > 0$ so the following holds:

$$\frac{1}{n}(X_i^n)' X_i^n \le M_1 \text{ for } \forall i, \tag{5}$$

$$\alpha' C_{11}^n \alpha \ge M_2, \text{ for } \forall \|\alpha\|_2^2 = 1, \tag{6}$$

$$q_n = O(n^{c_1}), \tag{7}$$

$$n^{\frac{1-c_2}{2}} \min_{i=1,..,q} |\beta_i^n| \ge M_3. \tag{8}$$

Condition (5) is trivial since it can always be achieved by normalizing the covariates. (6) requires the design of the relevant covariates have eigenvalues bounded from below so that the inverse

of C_{11}^n behaves well. For a random design, if the eigenvalues of the population covariance matrix are bounded from below and $q_n/n \to \rho < 1$ then (6) usually follows Bai (1999).

The main conditions are (7) and (8) which are similar to the ones in Meinshausen (2005) for Gaussian graphical models. (8) requires a gap of size n^{c_2} between the decay rate of $\beta_{(1)}^n$ and $n^{-\frac{1}{2}}$. Since the noise terms aggregate at a rate of $n^{-\frac{1}{2}}$, this prevents the estimation to be dominated by the noise terms. Condition (7) is a sparsity assumption which requires square root of the size of the true model $\sqrt{q_n}$ to grow at a rate slower than the rate gap which consequently prevents the estimation bias of the Lasso solutions from dominating the model parameters.

Under these conditions, we have the following result:

Theorem 3. Assume ε_i^n are i.i.d. random variables with finite $2k$'th moment $E(\varepsilon_i^n)^{2k} < \infty$ for an integer $k > 0$. Under conditions (5), (6), (7) and (8), Strong Irrepresentable Condition implies that Lasso has strong sign consistency for $p_n = o(n^{(c_2-c_1)k})$. In particular, for $\forall \lambda_n$ that satisfies $\frac{\lambda_n}{\sqrt{n}} = o(n^{\frac{c_2-c_1}{2}})$ and $\frac{1}{p_n}(\frac{\lambda_n}{\sqrt{n}})^{2k} \to \infty$, we have

$$P(\hat{\beta}^n(\lambda_n) =_s \beta^n) \geq 1 - O(\frac{p_n n^k}{\lambda_n^{2k}}) \to 1 \text{ as } n \to \infty.$$

A proof of Theorem 3 can be found in the appendix.

Theorem 3 states that Lasso can select the true model consistently given that Strong Irrepresentable Condition holds and the noise terms have some finite moments. For example, if only the second moment is assumed, p is allowed to grow slower than $n^{c_2-c_1}$. If all moments of the noise exist then, by Theorem 3, p can grow at any polynomial rate and the probability of Lasso selecting the true model converges to 1 at a faster rate than any polynomial rate. In particular, for Gaussian noises, we have:

Theorem 4 (Gaussian Noise). Assume ε_i^n are i.i.d. Gaussian random variables. Under conditions (5), (6), (7) and (8), if there exists $0 \leq c_3 < c_2 - c_1$ for which $p_n = O(e^{n^{c_3}})$ then strong Irrepresentable Condition implies that Lasso has strong sign consistency. In particular, for $\lambda_n \propto n^{\frac{1+c_4}{2}}$ with $c_3 < c_4 < c_2 - c_1$,

$$P(\hat{\beta}^n(\lambda_n) =_s \beta^n) \geq 1 - o(e^{-n^{c_3}}) \to 1 \text{ as } n \to \infty.$$

A proof of Theorem 4 can be found in the appendix. As discussed in the introduction, this result has also been obtained independently by Meinshausen and Buhlmann (2006) in their study of high dimensional multivariate Gaussian random variables. This result is obtained more directly for linear models and differs from theirs by the use of fixed designs to accommodate non-Gaussian designs. p_n is also allowed to grow slightly faster than the polynomial rates used in that work.

It is an encouraging result that using Lasso we can allow p to grow much faster than n (up to exponentially fast) while still allow for fast convergence of the probability of correct model selection to 1. However, we note that this fast rate is not achievable for all noise distributions. In general, the result of Theorem 3 is tight in the sense that if higher moments of the noise distribution do not exist then the tail probability of the noise terms does not vanish quick enough to allow p to grow at higher degree polynomial rates.

Through Theorem 3 and 4, we have shown, for cases with large p—(polynomial in n given that noise have finite moments, exponential in n for Gaussian noises), Strong Irrepresentable Condition still implies the probability of Lasso selecting the true model converges to 1 at a fast rate. We have found it difficult to show necessariness of Irrepresentable Condition for the large p setting in

a meaningful way. This is mainly due to the technical difficulty that arises from dealing with high dimensional design matrices. However, by the results for the small p case, the necessariness of Irrepresentable Condition is implied to some extent.

2.3 Analysis and Sufficient Conditions for Strong Irrepresentable Condition

In general, the Irrepresentable Condition is non-trivial when the numbers of zeros and nonzeros are of moderate sizes, for example, 3. Particularly since we do not know sign(β) before hand, we need the Irrepresentable Condition to hold for every possible combination of different signs and placement of zeros. A closer look discloses that (2) does not depend on C^n_{22}, that is, the covariance of the covariates that are not in the true model. It linearly depends on C^n_{21}, the correlations between the covariates that are in the model and the ones that are not. For the C^n_{11} part, except for special cases (Corollary 1) we also want the correlations between covariates that are in the model to be small otherwise C^n_{11} may contain small eigenvalues which leads to large eigenvalues for $(C^n_{11})^{-1}$ and results in the violation of (2).

To further elaborate and relate to previous works, we give some sufficient conditions in the following corollaries such that Strong Irrepresentable Condition is guaranteed. All diagonal elements of C^n are assumed to be 1 which is equivalent to normalizing all covariates in the model to the same scale since Strong Irrepresentable Condition is invariant under any common scaling of C^n. Proofs of the corollaries are included in the appendix.

Corollary 1. (Constant Positive Correlation) Suppose

$$C^n = \begin{pmatrix} 1 & \dots & r_n \\ \vdots & \ddots & \vdots \\ r_n & \dots & 1 \end{pmatrix}$$

and there exists $c > 0$ such that $0 < r_n \leq \frac{1}{1+cq}$, then Strong Irrepresentable Condition holds.

Corollary 1 has particularly strong implications for applications of Lasso where the covariates of the regression are designed with a symmetry so that the covariates share a constant correlation. Under such a design, this result implies that Strong Irrepresentable Condition holds even for p growing with n as long as q remains fixed and consequently ensures that Lasso selects the true model asymptotically. However, when the design is random or, for example, arises from an observational study we usually do not have the constant correlation. Correspondingly, we have the following result on bounded correlations.

Corollary 2. (Bounded Correlation) Suppose β has q nonzero entries. C^n has 1's on the diagonal and bounded correlation $|r_{ij}| \leq \frac{c}{2q-1}$ for a constant $0 \leq c < 1$ then Strong Irrepresentable Condition holds.

Corollary 2 verifies the common intuition that when the design matrix is slightly correlated Lasso works consistently. And the larger q is, the smaller the bound on correlation becomes. For a q of considerable size, the bound becomes too small to meet in practice. Unfortunately, this bound is also tight in the following sense: when the bound is violated, one can construct

$$C^n = \begin{pmatrix} 1 & \dots & r \\ \vdots & \ddots & \vdots \\ r & \dots & 1 \end{pmatrix}$$

with $r \leq -\frac{1}{2q-1}$ and make the nonzero β_i's all positive then $|C_{21}(C_{11})^{-1}\text{sign}(\beta(1))| \geq \mathbf{1}$ holds element-wise which fails Strong Irrepresentable Condition.

In comparison, Donoho et al. (2004) showed that, in a non-asymptotic setup, the L^2 distance between the sparsest estimate and the true model is bounded by a linear multiple of the noise level if

$$q < (1/r+1)/2,$$

where $r = \max_{i,j}|C^n_{ij}|$ (called Coherence). This is equivalent to

$$\max_{i,j}|C^n_{ij}| < \frac{1}{2q-1}$$

which coincides with the condition of Corollary 2. Interestingly, for the same result to apply to the Lasso estimates, Donoho et al. (2004) required tighter bound on the correlation, that is, $\max_{i,j}|C^n_{ij}| < \frac{1}{4q-1}$.

Another typical design used for Lasso simulations (e.g., Tibshirani, 1996; Zou et al., 2004) is setting the correlation between X^n_i and X^n_j to be $\rho^{|i-j|}$ with an constant $0 < \rho < 1$. Although this design introduces more sophisticated correlation structure between the predictors and does not seem restrictive, the following corollary states under this design Strong Irrepresentable Condition holds for any q.

Corollary 3. (Power Decay Correlation) Suppose for any $i, j = 1, ..., p$, $C^n_{ij} = (\rho_n)^{|i-j|}$, for $|\rho_n| \leq c < 1$, then Strong Irrepresentable Condition holds.

In addition, as instances of Corollary 2, under some simplified designs which are often used for theoretical studies, Lasso is consistent for model selection.

Corollary 4. If

- the design is orthogonal, or
- $q = 1$ and the predictors are normalized with correlations bounded from 1, or
- $p = 2$ and the predictors are normalized with correlations bounded from 1

then Strong Irrepresentable Condition holds.

One additional informative scenario to consider is a block-wise design. As it is commonly assumed in practice, this assumed scenario is a hybrid between the most highly structured designs like the orthogonal design and a general design. For this design, it can be shown that

Corollary 5. For a block-wise design such that

$$C^n = \begin{pmatrix} B^n_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & B^n_k \end{pmatrix}$$

with β^n written as $\beta^n = (b^n_1, ..., b^n_k)$ to correspond to different blocks, Strong Irrepresentable Condition holds if and only if there exists a common $0 < \eta \leq 1$ for which Strong Irrepresentable Condition holds for all B^n_j and b^n_j, $j = 1, ..., k$.

Combinations of Corollary 5 and Corollary 1-4 cover some interesting cases such as models with 2×2 design blocks and models where 0, 1 or all parameters out of each block are nonzero.

Through Corollaries 1 - 5, we have shown that under specific designs, which are commonly used or assumed in previous works, Irrepresentable Condition holds which leads to Lasso's consistency in model selection. Next, we demonstrate Lasso's model selection consistency and the Irrepresentable Conditions using simulations.

3. Simulation Studies

In this section, we give simulation examples to illustrate the established results. The first simulation illustrates the simplest case ($p = 3$, $q = 2$, cf. Corollary 4) under which Lasso is inconsistent for model selection. We also analyze the Lasso algorithm to explain how Lasso is misled into inconsistency when Irrepresentable Conditions fail. The second simulation quantitatively relates the consistency (inconsistency) of Lasso to how well the Strong Irrepresentable Condition holds (fails) by counting the percentages of Lasso selecting the true model and comparing it to $\eta_\infty = 1 - \|C_{21}^n (C_{11}^n)^{-1} \mathrm{sign}(\beta_{(1)}^n)\|_\infty$. In the last simulation, we establish a heuristic sense of how strong our Strong Irrepresentable Condition is for different values of p and q by observing how often the condition holds when C is sampled from Wishart(p, p) distribution.

3.1 Simulation Example 1: Consistency and Inconsistency with 3 Variables

In this simple example, we aim to give some practical sense of the Lasso algorithm's behaviors when Strong Irrepresentable Condition holds and fails. We first generate i.i.d. random variables x_{i1}, x_{i2}, e_i and ε_i with variance 1 and mean 0 for $i = 1, ..., n$ and $n = 1000$. A third predictor x_{i3} is generated to be correlated with x_{i1} and x_{i2} by

$$x_{i3} = \frac{2}{3} x_{i1} + \frac{2}{3} x_{i2} + \frac{1}{3} e_i,$$

then by construction, x_{i3} is also i.i.d. with mean 0 and variance 1.

The response is generated by

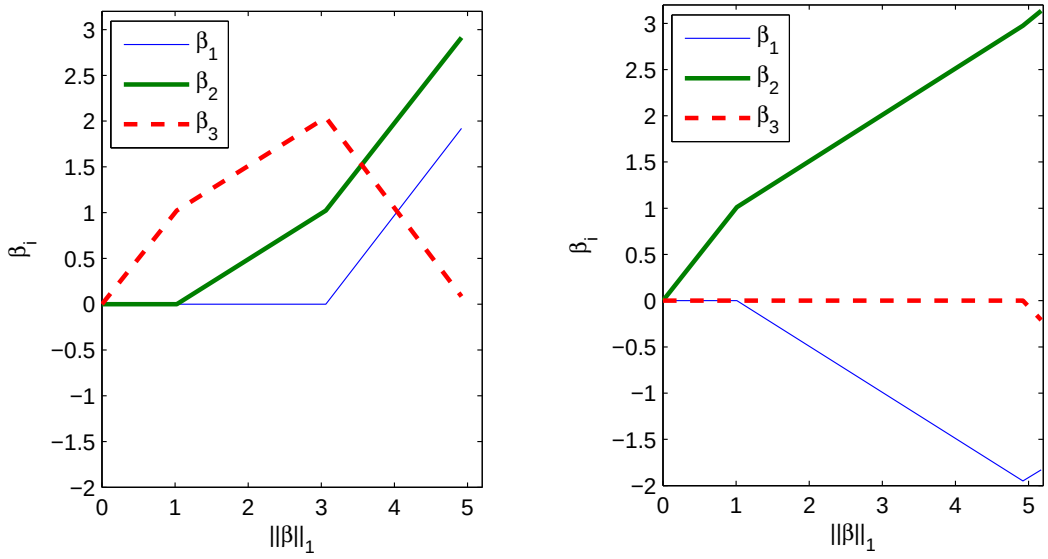
$$Y_i = x_{i1}\beta_1 + x_{i2}\beta_2 + \varepsilon_i.$$

Lasso is applied (through the LARS algorithm by Efron et al., 2004) on Y, X_1, X_2 and X_3 in two settings: (a) $\beta_1 = 2$, $\beta_2 = 3$; and (b) $\beta_1 = -2$, $\beta_2 = 3$. In both settings, $\mathbf{X}(1) = (X_1, X_2)$, $\mathbf{X}(2) = X_3$ and through (2), it is easy to get $C_{21}C_{11}^{-1} = (\frac{2}{3}, \frac{2}{3})$. Therefore Strong Irrepresentable Condition fails for setting (a) and holds for setting (b).

Now we investigate how these two set-ups lead to Lasso's sign consistency and inconsistency respectively. As we vary the amount of regularization (controlled by λ), we get different Lasso solutions which form the Lasso path (as illustrated by the left and right panels of Figure 1). This Lasso path follows the least angle direction (as described in for example, Efron et al. (2004) and Zhao and Yu (2004)), that is, $\hat{\beta}(\lambda)$ progresses in coordinates on which the absolute values of inner products between $Y^*(\lambda) := Y - X\hat{\beta}(\lambda)$ and the predictors are the largest while entries of $\hat{\beta}(\lambda)$ corresponding to smaller inner products are left at zero.

In this example,

$$\begin{aligned} |X_3' Y^*| &= |(\frac{2}{3}X_1 + \frac{2}{3}X_2 + \frac{1}{3}e)' Y^*| \\ &\geq \frac{4}{3} \min(|X_1' Y^*|, |X_2' Y^*|)(\frac{\mathrm{sign}(X_1' Y^*) + \mathrm{sign}(X_2' Y^*)}{2}) - \frac{1}{3}|e' Y^*|. \end{aligned}$$



(a) $\beta_1 = 2, \beta_2 = 3$ (b) $\beta_1 = -2, \beta_2 = 3$

Figure 1: An example to illustrate Lasso's (in)consistency in Model Selection. The Lasso paths for settings (a) and (b) are plotted in the left and right panel respectively.

For large n, $e' * Y^*$ is on a smaller order than the rest of the terms. If $\hat{\beta}_3$ is zero, the signs of X_1's and X_2's inner products with Y agree with the signs of $\hat{\beta}_1$ and $\hat{\beta}_2$. Therefore for Lasso to be sign consistent, the signs of β_1 and β_2 has to disagree which happens in setting (b) but not setting (a).

Consequently, in setting (a) Lasso does not shrink $\hat{\beta}_3$ to 0. Instead, the L_1 regularization prefers X_3 over X_1 and X_2 as Lasso picks up X_3 first and never shrinks it back to zero. For setting (b), Strong Irrepresentable Condition holds and with a proper amount of regularization, Lasso correctly shrinks $\hat{\beta}_3$ to 0.

3.2 Simulation Example 2: Quantitative Evaluation of Impact of Strong Irrepresentable Condition on Model Selection

In this example, we give some quantitative sense on the relationship between the probability of Lasso selecting the correct model and how well Strong Irrepresentable Condition holds (or fails). First, we take $n = 100$,$p = 32$, $q = 5$ and $\beta_1 = (7,4,2,1,1)^T$ and choose a small $\sigma^2 = 0.1$ to allow us to go into asymptotic quickly.

Then we would like to generate 100 designs of X as follows. We first sample a covariance matrix S from Wishart(p,p) (see section 3.3 for details), then take n samples of X_i from $N(0,S)$, and finally normalize them to have mean squares 1 as in common applications of Lasso. Such generated samples represent a variety of designs: some satisfy Strong Irrepresentable Condition with a large η, while others fail the condition badly. To evaluate how well the Irrepresentable condition holds we calculate $\eta_\infty = 1 - \|C_{21}^n (C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)\|_\infty$. So if $\eta_\infty > 0$, Strong Irrepresentable holds otherwise it

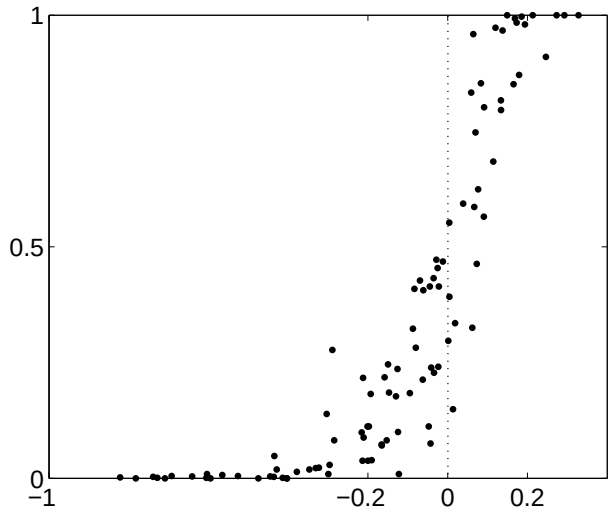


Figure 2: Comparison of Percentage of Lasso Selecting the Correct Model and η_∞. X-axis: η_∞. Y-axis: Percentage of Lasso Selecting the Correct Model.

fails. The η_∞'s of the 100 simulated designs are within $[-1.02, 0.33]$ with 67 of them being smaller than 0 and 33 of them bigger than 0.

For each design, we run the simulation 1000 times and examine general sign consistencies. Each time, n samples of ε are generated from $N(0,\sigma^2)$ and $Y = X\beta + \varepsilon$ are calculated. We then run Lasso (through the LARS algorithm by Efron et al., 2004) to calculate the Lasso path. The entire path is examined to see if there exists a model estimate that matches the signs of the true model. Then we compute the percentage of runs that generated matched models for each design and compare it to η_∞ as shown in Figure 2.

As can be seen from Figure 2, when η_∞ gets large, the percentage of Lasso selecting the correct model goes up with the steepest increase happening around 0. For η_∞ considerably larger than 0 (> 0.2) the percentage is close to 1. On the other hand, for η_∞ considerably smaller than 0 (< -0.3) there is little chance for Lasso to select the true model. In general, this is consistent with our result (Proposition 1 and Theorem 1 to 4), as for $\eta_\infty > 0$, if n is large enough, the probability of Lasso selects the true model gets close to 1 which does not happen if $\eta_\infty < 0$. This quantitatively illustrates the importance of Strong Irrepresentable Condition for Lasso's model selection performance.

3.3 Simulation Example 3: How Strong is Irrepresentable Condition?

As illustrated by Corollaries 1 to 4, Strong Irrepresentable Condition holds for some constrained special settings. While in Section 3.1 and 3.2, we have seen cases where Irrepresentable Condition fails. In this simulation, we establish some heuristic sense of how strong our Strong Irrepresentable Condition is for different values of p and q.

For a given p, the set of C^n is the set of nonnegative definite matrix of size p. To measure the size of the subset of C^n's on which Irrepresentable Condition holds, the Wishart measure family can be used. Since Strong Irrepresentable Condition holds for designs that are close to orthogonal

	$p=2^3$	$p=2^4$	$p=2^5$	$p=2^6$	$p=2^7$	$p=2^8$
$q=\frac{1}{8}p$	100%	93.7%	83.1%	68.6%	43.0%	19.5%
$q=\frac{2}{8}p$	72.7%	44.9%	22.3%	4.3%	$<1\%$	0%
$q=\frac{3}{8}p$	48.3%	19.2%	3.4%	$<1\%$	0%	0%
$q=\frac{4}{8}p$	33.8%	8.9%	1.3%	0%	0%	0%
$q=\frac{5}{8}p$	23.8%	6.7%	$<1\%$	0%	0%	0%
$q=\frac{6}{8}p$	26.4%	7.1%	$<1\%$	0%	0%	0%
$q=\frac{7}{8}p$	36.3%	12.0%	1.8%	0%	0%	0%

Table 1: Percentage of Simulated C^n that meet Strong Irrepresentable Condition.

(Corollary 2), we take the Wishart(p,p) measure which centers but does not concentrate around the identity matrix.

In this simulation study, we sample C^n's from white Wishart(p,p) and examine how often Irrepresentable Condition holds. For each $p=2^3,2^4,2^5,2^6,2^7,2^8$ and correspondingly $q=\frac{1}{8}p,\frac{2}{8}p,...,\frac{7}{8}p$ we generate 1000 C^n's from Wishart and re-normalize it to have 1's on the diagonal. Then we examine how often Irrepresentable Condition holds. The entries of $\beta(1)$ are assumed to be positive, otherwise a sign flip of the corresponding X_i's can make the corresponding β_i positive. The result is shown in Table 1.

Table 1 shows that, when the true model is very sparse (q small), Strong Irrepresentable Condition has some probability to hold which illustrates Corollary 2's conclusion. For the extreme case, $q=1$, it has been proved to hold (see Corollary 4). However, in general, for large p and q, Irrepresentable Condition rarely (measured by Wishart(p,p)) holds.

4. Discussions

In this paper, we have provided Strong and Weak Irrepresentable Conditions that are almost necessary and sufficient for model selection consistency of Lasso under both small p and large p settings. We have explored the meaning of the conditions through theoretical and empirical studies. Although much of Lasso's strength lies in its finite sample performance which is not the focus here, our asymptotic results offer insights and guidance to applications of Lasso as a feature selection tool, assuming that the typical regularity conditions are satisfied on the design matrix as in Knight and Fu (2000). As a precaution, for data sets that can not be verified to satisfy the Irrepresentable Conditions, Lasso may not select the model correctly. In comparison, traditional all subset methods like BIC and MDL are always consistent but computationally intractable for p of moderate sizes. Thus, alternative computationally feasible methods that lead to selection consistency when the condition fails are of interest.

In particular, for small p cases, if consistency is the only concern then thresholding (either hard or soft) is an obvious choice that guarantees consistent selection. Since the OLS estimate $\hat{\beta}_{OLS}$ converges at a $1/\sqrt{n}$ rate, therefore a threshold that satisfies $t_n/\sqrt{n}\to\infty$ and $t_n\to 0$ leads to consistent selection. However, as emphasized earlier, consistency does not mean good performance in finite sample which is what matters in many applications where Lasso-type of technique is used. In particular, when the linear system is over determined $p>n$, the approach is no longer applicable

since the OLS estimates are not well defined. On the other hand, Theorem 3 and Theorem 4 indicate that for cases where p may grow much faster then n, the Lasso still perform well.

To get some intuitive sense of how the thresholding performs comparing to the Lasso in finite sample, we ran the same simulations as in Section 3.2 and examined the sign matching rate of thresholding and compare it to the Lasso's performance. Our observation is, when the sample size is large, that is, in the asymptotic domain, even when Strong Irrepresentable Condition holds, Lasso does not perform better than simple thresholding in term of variable selection. In the small sample domain, however, Lasso seems to show an advantage which is consistent with the results reported in other publications (e.g., Tibshirani, 1996).

Another alternative that selects model consistently in our simulations is given by Osborne et al. (1998). They advise to use Lasso to do initial selection. Then a best subset selection (or a similar procedure, for example, forward selection) should be performed on the initial set selected by Lasso. This is loosely justified since, for instance, from Knight and Fu (2000) we know Lasso is consistent for $\lambda = o(n)$ and therefore can pick up all the true predictors if the amount of data is sufficient (although it may over-select).

Finally, we think it is possible to directly construct an alternative regularization to Lasso that selects model consistently under much weaker conditions and at the same time remains computationally feasible. This relies on understanding why Lasso is inconsistent when Strong Irrepresentable Condition fails: to induce sparsity, Lasso shrinks the estimates for the nonzero β's too heavily. When Strong Irrepresentable Condition fails, the irrelevant covariates are correlated with the relevant covariates enough to be picked up by Lasso to compensate the over-shrinkage of the nonzero β's. Therefore, to get universal consistency, we need to reduce the amount of shrinkage on the β estimates that are away from zero and regularize in a more similar fashion as l_0 penalty. However, as a consequence, this breaks the convexity of the Lasso penalty, therefore more sophisticated algorithms are needed for solving the minimization problems. A different set of analysis is also needed to deal with the local minima. This points towards our future work.

Acknowledgments

This research is partly supported by NSF Grant DMS-03036508 and ARO Grant W911NF-05-1-0104. Yu is also very grateful for the support of a 2006 Guggenheim Fellowship.

Appendix A. Proofs

To prove Proposition 1 and the rest of the theorems, we state Lemma 1 which is a direct consequence of KKT (Karush-Kuhn-Tucker) conditions:

Lemma 1. $\hat{\beta}^n(\lambda) = (\hat{\beta}_1^n, ..., \hat{\beta}_j^n, ...)$ are the Lasso estimates as defined by (1) if and only if

$$\begin{aligned} \frac{d\|Y_n - \mathbf{X_n}\beta\|_2^2}{d\beta_j}|_{\beta_j=\hat{\beta}_j^n} &= \lambda \text{sign}(\hat{\beta}_j^n) \quad \text{for } j \text{ s.t. } \hat{\beta}_j^n \neq 0 \\ |\frac{d\|Y_n - \mathbf{X_n}\beta\|_2^2}{d\beta_j}|_{\beta_j=\hat{\beta}_j^n}| &\leq \lambda \qquad \text{for } j \text{ s.t. } \hat{\beta}_j^n = 0. \end{aligned}$$

With Lemma 1, we now prove Proposition 1.

Proof of Proposition 1. First, by definition

$$\hat{\beta}^n = \arg\min_{\beta}[\sum_{i=1}^{n}(Y_i - X_i\beta)^2 + \lambda_n\|\beta\|_1].$$

Let $\hat{u}^n = \hat{\beta}^n - \beta^n$, and define

$$V_n(u^n) = \sum_{i=1}^{n}[(\varepsilon_i - X_i u^n)^2 - \varepsilon_i^2] + \lambda_n\|u^n + \beta\|_1,$$

we have

$$\begin{aligned}\hat{u}^n &= \arg\min_{u^n}[\sum_{i=1}^{n}(\varepsilon_i - X_i u^n)^2 + \lambda_n\|u^n + \beta\|_1]. \\ &= \arg\min_{u^n} V_n(u^n). \end{aligned} \tag{9}$$

The first summation in $V_n(u^n)$ can be simplified as follows:

$$\begin{aligned}&\sum_{i=1}^{n}[(\varepsilon_i - X_i u^n)^2 - \varepsilon_i^2] \\ =\ &\sum_{i=1}^{n}[-2\varepsilon_i X_i u^n + (u^n)^T X_i^T X_i u^n], \\ =\ &-2W^n(\sqrt{n}u^n) + (\sqrt{n}u^n)^T C^n(\sqrt{n}u^n), \end{aligned} \tag{10}$$

where $W^n = (X^n)^T\varepsilon^n/\sqrt{n}$. Notice that (10) is always differentiable w.r.t. u^n and

$$\frac{d[-2W^n(\sqrt{n}u^n) + (\sqrt{n}u^n)^T C_n(\sqrt{n}u^n)]}{du^n} = 2\sqrt{n}(C^n(\sqrt{n}u^n) - W^n). \tag{11}$$

Let $\hat{u}^n(1)$, $W^n(1)$ and $\hat{u}^n(2)$, $W^n(2)$ denote the first q and last $p-q$ entries of $\hat{u}^n$ and W^n respectively. Then by definition we have:

$$\{\text{sign}(\hat{\beta}_j^n) = \text{sign}(\beta_j^n), \text{ for } j = 1,...,q.\} \in \{\text{sign}(\beta_{(1)}^n)\hat{u}^n(1) > -|\beta_{(1)}^n|\}.$$

Then by Lemma 1, (9), (11) and uniqueness of Lasso solutions, if there exists $\hat{u}^n$, the following holds

$$C_{11}^n(\sqrt{n}\hat{u}^n(1)) - W^n(1) = -\frac{\lambda_n}{2\sqrt{n}}\text{sign}(\beta_{(1)}^n),$$

$$|\hat{u}^n(1)| < |\beta_{(1)}^n|,$$

$$-\frac{\lambda_n}{2\sqrt{n}}\mathbf{1} \le C_{21}^n(\sqrt{n}\hat{u}^n(1)) - W^n(2) \le \frac{\lambda_n}{2\sqrt{n}}\mathbf{1}.$$

then $\text{sign}(\hat{\beta}_{(1)}^n) = \text{sign}(\beta_{(1)}^n)$ and $\hat{\beta}_{(2)}^n = u^n(2) = 0$.

Substitute $\hat{u}^n(1)$, $\hat{u}^n(2)$ and bound the absolute values, the existence of such $\hat{\mu}^n$ is implied by

$$|(C_{11}^n)^{-1}W^n(1)| < \sqrt{n}(|\beta_{(1)}^n| - \frac{\lambda_n}{2n}|(C_{11}^n)^{-1}\text{sign}(\beta_{(1)}^n)|), \tag{12}$$

$$|C_{21}^n(C_{11}^n)^{-1}W^n(1)-W^n(2)| \leq \frac{\lambda_n}{2\sqrt{n}}(\mathbf{1}-|C_{21}^n(C_{11}^n)^{-1}\text{sign}(\beta_{(1)}^n)|) \tag{13}$$

(12) coincides with A_n and (13)$\in B_n$. This proves Proposition 1.

Using Proposition 1, we now prove Theorem 1.

Proof of Theorem 1. First, by Proposition 1 we have By Proposition 1, we have

$$P(\hat{\beta}^n(\lambda_n) =_s \beta) \geq P(A_n \cap B_n).$$

Whereas

$$\begin{aligned} 1-P(A_n\cap B_n) &\leq P(A_n^c)+P(B_n^c) \\ &\leq \sum_{i=1}^{q} P(|z_i^n| \geq \sqrt{n}(|\beta_i^n| - \frac{\lambda_n}{2n}b_i^n) + \sum_{i=1}^{p-q} P(|\zeta_i^n| \geq \frac{\lambda_n}{2\sqrt{n}}\eta_i). \end{aligned}$$

where $z^n = (z_1^n,...,z_p^n)' = (C_{11}^n)^{-1}W^n(1)$, $\zeta^n = (\zeta_1^n,...,\zeta_{p-q}^n)' = C_{21}^n(C_{11}^n)^{-1}W^n(1) - W^n(2)$ and $b = (b_1^n,...,b^n) = (C_{11}^n)^{-1}\text{sign}(\beta_{(1)}^n)$.

It is standard result (see for example, Knight and Fu, 2000) that under regularity conditions (3) and (4),

$$(C_{11}^n)^{-1}W^n(1) \to_d N(0, C_{11}^{-1}),$$

and

$$C_{21}^n(C_{11}^n)^{-1}W^n(1) - W^n(2) \to_d N(0, C_{22} - C_{21}C_{11}^{-1}C_{12}).$$

Therefore all z_i^n's and ζ_i^n's converge in distribution to Gaussian random variables with mean 0 and finite variance $E(z_i^n)^2, E(\zeta_i^n)^2 \leq s^2$ for some constant $S > 0$.

For $t > 0$, the Gaussian distribution has its tail probability bounded by

$$1-\Phi(t) < t^{-1}e^{-\frac{1}{2}t^2}. \tag{14}$$

Since $\frac{\lambda_n}{n} \to 0$, $\frac{\lambda_n}{n^{\frac{1+c}{2}}} \to \infty$ with $0 \leq c < 1$, p, q and β^n are all fixed, therefore

$$\begin{aligned} &\sum_{i=1}^{q} P(|z_i^n| \geq \sqrt{n}(|\beta_i^n| - \frac{\lambda_n}{2n}b_i^n) \\ \leq\ & (1+o(1))\sum_{i=1}^{q}(1-\Phi((1+o(1))\frac{1}{s}n^{\frac{1}{2}}|\beta_i|)) \\ =\ & o(e^{-n^c}), \end{aligned}$$

and

$$\sum_{i=1}^{p-q} P(|\zeta_i^n| \geq \frac{\lambda_n}{2\sqrt{n}}\eta_i) = \sum_{i=1}^{p-q}(1-\Phi(\frac{1}{s}\frac{\lambda_n}{2\sqrt{n}}\eta_i)) = o(e^{-n^c}).$$

Theorem 1 follows immediately.

Proof of Theorem 2. Consider the set F_1^n, on which there exists λ_n such that,

$$\begin{aligned} \text{sign}(\hat{\beta}_{(1)}^n) &= \text{sign}(\beta_{(1)}^n) \\ (\hat{\beta}_{(2)}^n) &= 0. \end{aligned}$$

General Sign Consistency implies that $P(F_1^n) \to 1$ as $n \to 1$.

Conditions of F_1^n imply that $\hat{\beta}^n_{(1)} \neq 0$ and $\hat{\beta}^n_{(2)} = 0$. Therefore by Lemma 1 and (11) from the proof of Proposition 1, we have

$$C_{11}^n(\sqrt{n}\hat{u}^n(1)) - W^n(1) = -\frac{\lambda_n}{2\sqrt{n}}\text{sign}(\hat{\beta}^n_{(1)}) = -\frac{\lambda_n}{2\sqrt{n}}\text{sign}(\beta^n_{(1)}) \quad (15)$$

$$|C_{21}^n(\sqrt{n}\hat{u}^n(1)) - W^n(2)| \leq \frac{\lambda_n}{2\sqrt{n}}\mathbf{1} \quad (16)$$

which hold over F_1^n.

Re-write (16) by replacing $\hat{u}^n(1)$ using (15), we get

$$F_1^n \subset F_2^n := \{(\lambda_n/2\sqrt{n})L^n \leq C_{21}^n(C_{11}^n)^{-1}W^n(1) - W^n(2) \leq (\lambda_n/2\sqrt{n})U^n\}$$

where

$$\begin{aligned} L^n &= -1 + C_{21}^n(C_{11}^n)^{-1}\text{sign}(\beta^n_{(1)}), \\ U^n &= 1 + C_{21}^n(C_{11}^n)^{-1}\text{sign}(\beta^n_{(1)}). \end{aligned}$$

To prove by contradiction, if Weak Irrepresentable Condition fails, then for any N there always exists $n > N$ such that at least one element of $|C_{21}^n(C_{11}^n)^{-1}\text{sign}(\beta^n_{(1)})| \geq 1$. Without loss of generality, assume the first element of $C_{21}^n(C_{11}^n)^{-1}\text{sign}(\beta^n_{(1)}) \geq 1$, then

$$[(\lambda_n/2\sqrt{n})L_1^n, (\lambda_n/2\sqrt{n})U_1^n] \subset [0, +\infty),$$

for any $\lambda_n \geq 0$. Since $C_{21}^n(C_{11}^n)^{-1}W^n(1) - W^n(2) \to_d N(0, C_{22} - C_{21}C_{11}^{-1}C_{12})$, there is a non-vanishing probability that the first element is negative, then the probability of F_2^n holds does not go to 1, therefore

$$\liminf P(F_1^n) \leq \liminf P(F_2^n) < 1.$$

This contradicts with the General Sign Consistency assumption. Therefore Weak Irrepresentable Condition is necessary for General Sign Consistency.

This completes the proof.

Proofs of Theorem 3 and 4 are similar to that of Theorem 1. The goal is to bound the tail probabilities in Proposition 1 using different conditions on the noise terms. We first derive the following inequalities for both Theorem 3 and 4.

Proof of Theorem 3 and Theorem 4. As in the proof of Theorem 1, we have

$$\begin{aligned} 1 - P(A_n \cap B_n) &\leq P(A_n^c) + P(B_n^c) \\ &\leq \sum_{i=1}^{q} P(|z_i^n| \geq \sqrt{n}(|\beta_i^n| - \frac{\lambda_n}{2n}b_i^n) + \sum_{i=1}^{p-q} P(|\zeta_i^n| \geq \frac{\lambda_n}{2\sqrt{n}}\eta_i). \end{aligned}$$

where $z^n = (z_1^n, ..., z_p^n)' = (C_{11}^n)^{-1}W^n(1)$, $\zeta^n = (\zeta_1^n, ..., \zeta_{p-q}^n)' = C_{21}^n(C_{11}^n)^{-1}W^n(1) - W^n(2)$ and $b = (b_1^n, ..., b^n) = (C_{11}^n)^{-1}\text{sign}(\beta^n_{(1)})$.

Now if we write $z_i^n = H_A'\varepsilon^n$ where $H_A' = (h_1^a, ..., h_q^a)' = (C_{11}^n)^{-1}(n^{-\frac{1}{2}}\mathbf{X^n}(1))$, then

$$H_A'H_A = (C_{11}^n)^{-1}(n^{-\frac{1}{2}}\mathbf{X^n}(1)')((C_{11}^n)^{-1}(n^{-\frac{1}{2}}\mathbf{X^n}(1))')' = (C_{11}^n)^{-1}.$$

Therefore $z_i^n = (h_i^a)'\varepsilon$ with

$$\|h_i^a\|_2^2 \leq \frac{1}{M_2} \text{ for } \forall i = 1,..,q. \tag{17}$$

Similarly if we write $\zeta^n = H_B'\varepsilon^n$ where $H_B' = (h_1^b,...,h_{p-q}^b)' = C_{21}^n(C_{11}^n)^{-1}(n^{-\frac{1}{2}}\mathbf{X^n}(1)')$ $-n^{-\frac{1}{2}}\mathbf{X^n}(2)'$, then

$$H_B'H_B = \frac{1}{n}(\mathbf{X^n}(2))'(I - \mathbf{X^n}(1)((\mathbf{X^n}(1)'(\mathbf{X^n}(1))^{-1}\mathbf{X^n}(1)')\mathbf{X^n}(2).$$

Since $I - \mathbf{X^n}(1)((\mathbf{X^n}(1)'(\mathbf{X^n}(1))^{-1}\mathbf{X^n}(1)'$ has eigenvalues between 0 and 1, therefore $\zeta_i^n = (h_i^b)'\varepsilon$ with

$$\|h_i^b\|_2^2 \leq M_1 \text{ for } \forall i = 1,..,q. \tag{18}$$

Also notice that,

$$|\frac{\lambda_n}{n}b_n| = \frac{\lambda_n}{n}|(C_{11}^n)^{-1}\text{sign}(\beta_{(1)}^n)| \leq \frac{\lambda_n}{nM_2}\|\text{sign}(\beta_{(1)}^n)\|_2 = \frac{\lambda_n}{nM_2}\sqrt{q} \tag{19}$$

Proof of Theorem 3. Now, given (17) and (18), it can be shown that $E(\varepsilon_i^n)^{2k} < \infty$ implies $E(z_i^n)^{2k} < \infty$ and $E(\zeta_i^n)^{2k} < \infty$. In fact, given constant n-dimensional vector α,

$$E(\alpha'\varepsilon^n)^{2k} \leq (2k-1)!!\|\alpha\|_2^2 E(\varepsilon_i^n)^{2k}.$$

For radome variables with bounded $2k$'th moments, we have their tail probability bounded by

$$P(z_i^n > t) = O(t^{-2k}).$$

Therefore, for $\lambda/\sqrt{n} = o(n^{\frac{c_2-c_1}{2}})$, using (19), we get

$$\begin{aligned} & \sum_{i=1}^{q} P(|z_i^n| > \sqrt{n}(|\beta_i^n| - \frac{\lambda_n}{2n}b_i^n) \\ = \quad & qO(n^{-kc_2}) = o(\frac{pn^k}{\lambda_n^{2k}}). \end{aligned}$$

Whereas

$$\begin{aligned} & \sum_{i=1}^{p-q} P(|\zeta_i^n| > \frac{\lambda_n}{2\sqrt{n}}\eta_i) \\ = \quad & (p-q)O(\frac{n^k}{\lambda_n^{2k}}) = O(\frac{pn^k}{\lambda_n^{2k}}). \end{aligned}$$

Sum these two terms and notice for $p = o(n^{c_2-c_1})$, there exists a sequence of λ_n s.t. $\lambda/\sqrt{n} = o(n^{\frac{c_2-c_1}{2}})$ and $= o(\frac{pn^k}{\lambda_n^{2k}})$. This completes the proof for Theorem 3.

Proof of Theorem 4. Since ε_i^n's are i.i.d. Gaussian, therefore by (17) and (18), z_i's and ζ_i's are Gaussian with bounded second moments.

Using the tail probability bound (14) on Gaussian random variables , for $\lambda_n \propto n^{\frac{1+c_4}{2}}$, by (19) we immediately have

$$\begin{aligned} & \sum_{i=1}^{q} P(|z_i^n| > \sqrt{n}(|\beta_i^n| - \frac{\lambda_n}{2n} b_i^n) \\ = \quad & q \cdot O(1 - \Phi(\ (1+o(1))M_3 M_2 n^{c_2/2}\ ) = o(e^{-n^{c_3}}). \end{aligned}$$

(since $q < n = e^{\log n}$) and

$$\begin{aligned} & \sum_{i=1}^{p-q} P(|\zeta_i^n| > \frac{\lambda_n}{2\sqrt{n}} \eta_i) \\ = \quad & (p-q) \cdot O(1 - \Phi(\frac{1}{M_1} \frac{\lambda_n}{\sqrt{n}})\eta) = o(e^{-n^{c_3}}). \end{aligned}$$

This completes the proof for Theorem 4.

Proof of Corollary 1. First we recall, for a positive definite matrix of the form

$$\begin{pmatrix} a & b & \cdots & b & b \\ b & a & \cdots & b & b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b & b & \cdots & a & b \\ b & b & \cdots & b & a \end{pmatrix}_{q \times q}.$$

The eigenvalues are $e_1 = a + (q-1)b$ and $e_i = a - b$ for $i \geq 2$. Therefore the inversion of

$$C_{11}^n = \begin{pmatrix} 1 & r_n & \cdots & r_n & r_n \\ r_n & 1 & \cdots & r_n & r_n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r_n & r_n & \cdots & 1 & r_n \\ r_n & r_n & \cdots & r_n & 1 \end{pmatrix}_{q \times q}$$

can be obtained by applying the formula and taking reciprocal of the eigenvalues which gets us

$$(C_{11}^n)^{-1} = \begin{pmatrix} c & d & \cdots & d & d \\ d & c & \cdots & d & d \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d & d & \cdots & c & d \\ d & d & \cdots & d & c \end{pmatrix}_{q \times q}$$

for which $e_1' = c + (q-1)d = \frac{1}{e_1} = \frac{1}{1+(q-1)r_n}$.

Now since $C_{21}^n = r_n \times \mathbf{1}_{(p-q)\times q}$ so

$$C_{21}^n (C_{11}^n)^{-1} = r_n(c + (q-1)d)\mathbf{1}_{(p-q)\times q} = \frac{r_n}{1+(q-1)r_n} \mathbf{1}_{(p-q)\times q}.$$

By which, we get

$$\begin{aligned} |C_{21}^n (C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)| &= \frac{r_n}{1+(q-1)r_n} |\sum \text{sign}(\beta_i)| \mathbf{1}_{q\times 1} \\ &\leq \frac{q r_n}{1+(q-1)r_n} \mathbf{1}_{q\times 1} \leq \frac{\frac{q}{1+cq}}{1+\frac{q-1}{1+cq}} = \frac{1}{1+c}, \end{aligned}$$

that is, Strong Irrepresentable Condition holds.

Proof of Corollary 2. Without loss of generality, consider the first entry of $|C_{21}^n (C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)|$ which takes the form $|\alpha'(C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)|$ where α' is the first row of C_{21}^n. After proper scaling, this is bounded by the largest eigenvalue of $(C_{11}^n)^{-1}$ or equivalently the reciprocal of the smallest eigenvalue of C_{11}^n, that is,

$$|\alpha'(C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)| \leq \|\alpha\| \|\text{sign}(\beta_{(1)}^n)\| \frac{1}{e_1} < \frac{cq}{2q-1} \frac{1}{e_1}. \tag{20}$$

To bound e_1, we assume $C_{11}^n = c_{ij}{}_{q\times q}$. Then for a unit length $q \times 1$ vector $x = (x_1, ..., x_q)'$, we consider

$$\begin{aligned} x' C_{11}^n x &= \sum_{i,j} x_i c_{ij} x_j = 1 + \sum_{i\neq j} x_i c_{ij} x_j \\ &\geq 1 - \sum_{i\neq j} |x_i||c_{ij}||x_j| \\ &\geq 1 - \frac{1}{2q-1} \sum_{i\neq j} |x_i||x_j| \\ &= 1 - \frac{1}{2q-1} (\sum_{i,j} |x_i||x_j| - 1) \\ &\geq 1 - \frac{q-1}{2q-1} = \frac{q}{2q-1}, \end{aligned} \tag{21}$$

where the last inequality is by Cauchy-Schwartz. Now put (21) through (20), we have $|C_{21}^n (C_{11}^n)^{-1} \text{sign}(\beta_{(1)}^n)| < c\mathbf{1}$. This completes the proof for Corollary 2.

Proof of Corollary 3. Without loss of generality, let us assume x_j, $j = 1, ..., n$ are i.i.d. $N(0, C^n)$ random variables. Then the power decay design implies an AR(1) model where

$$\begin{aligned} x_{j1} &= \eta_{j1} \\ x_{j2} &= \rho x_{j1} + (1-\rho^2)^{\frac{1}{2}} \eta_{j2} \\ &\vdots \\ x_{jp} &= \rho x_{j(p-1)} + (1-\rho^2)^{\frac{1}{2}} \eta_{jp} \end{aligned}$$

where η_{ij} are i.i.d. $N(0,1)$ random variables. Thus, the predictors follow a Markov Chain:

$$x_{j1} \to x_{j2} \to \cdots \to x_{jp}.$$

Now let

$$\begin{aligned} I_1 &= i : \beta_i \neq 0 \\ I_2 &= i : \beta_i = 0. \end{aligned}$$

For $\forall k \in I_2$, assume

$$\begin{aligned} k_l &= \{i : i < k\} \cap I_1 \\ k_h &= \{i : i > k\} \cap I_1. \end{aligned}$$

Then by the Markov property, we have

$$x_{jk} \perp x_{jg} | (x_{jk_l}, x_{jk_h})$$

for $j = 1, .., n$ and $\forall g \in I_1 / \{k_l, k_h\}$. Therefore by the regression interpretation as in (2), to check Strong Irrepresentable Condition for x_{jk} we only need to consider x_{jk_l} and x_{jk_h} since the rest of the entries are zero by the conditional independence. To further simplify, we assume $\rho \geq 0$ (otherwise ρ can be modified to be positive by flipping the signs of predictors $1, 3, 5, ...$). Now regressing x_{jk} on (x_{jk_l}, x_{jk_h}) we get

$$\begin{aligned} & \mathrm{Cov}(\begin{pmatrix} x_{jk_l} \\ x_{jk_h} \end{pmatrix})^{-1} \mathrm{Cov}(x_{jk}, \begin{pmatrix} x_{jk_l} \\ x_{jk_h} \end{pmatrix}) \\ = & \begin{pmatrix} 1 & \rho^{k_h - k_l} \\ \rho^{k_h - k_l} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \rho^{k_h - k} \\ \rho^{k - k_l} \end{pmatrix} \\ = & \begin{pmatrix} \frac{\rho^{k_l - k} - \rho^{k - k_l}}{\rho^{k_l - k_h} - \rho^{k_h - k_l}} \\ \frac{\rho^{k - k_h} - \rho^{k_h - k}}{\rho^{k_l - k_h} - \rho^{k_h - k_l}} \end{pmatrix}. \end{aligned}$$

Then sum of both entries follow

$$\frac{\rho^{k_l - k} - \rho^{k - k_l}}{\rho^{k_l - k_h} - \rho^{k_h - k_l}} + \frac{\rho^{k - k_h} - \rho^{k_h - k}}{\rho^{k_l - k_h} - \rho^{k_h - k_l}} = \frac{\rho^{k_l - k} + \rho^{k - k_h}}{1 + \rho^{k_l - k_h}} = 1 - \frac{(1 - \rho^{k_l - k})(1 - \rho^{k - k_h})}{1 + \rho^{k_l - k_h}} < 1 - \frac{(1 - c)^2}{2}.$$

Therefore Strong Irrepresentable Condition holds entry-wise. This completes the proof.

Proof of Corollary 4.

(a) Since the correlations are all zero so the condition of Corollary 2 holds for $\forall q$. Therefore Strong Irrepresentable Condition holds.

(b) Since $q = 1$, so $\frac{1}{2q-1} = 1$ therefore the condition of Corollary 2 holds. Therefore Strong Irrepresentable Condition holds.

(c) Since $p = 2$, therefore for $q = 0$ or 2, proof is trivial. When $q = 1$, result is implied by (b).

Proof of Corollary 5.

Let $\mathcal{M}$ be the set of indices of nonzero entries of β^n and $\mathcal{B}_j$, $j = 1,..,k$ be the set of indices of each block. Then the following holds

$$
\begin{aligned}
& C^n_{21}(C^n_{11})^{-1}\mathrm{sign}(\beta^n_{(1)}) \\
= & \begin{pmatrix} C^n_{\mathcal{M}^c\cap\mathcal{B}_1,\mathcal{M}\cap\mathcal{B}_1} & \cdots & 0 \\ \cdots & \ddots & \cdots \\ 0 & \cdots & C^n_{\mathcal{M}^c\cap\mathcal{B}_k,\mathcal{M}\cap\mathcal{B}_k} \end{pmatrix} \\
& \times \begin{pmatrix} C^n_{\mathcal{M}\cap\mathcal{B}_1,\mathcal{M}\cap\mathcal{B}_1} & \cdots & 0 \\ \cdots & \ddots & \cdots \\ 0 & \cdots & C^n_{\mathcal{M}\cap\mathcal{B}_k,\mathcal{M}\cap\mathcal{B}_k} \end{pmatrix}^{-1} \mathrm{sign}\left(\begin{pmatrix} \beta^n_{\mathcal{M}\cap\mathcal{B}_1} \\ \vdots \\ \beta^n_{\mathcal{M}\cap\mathcal{B}_k} \end{pmatrix}\right) \\
= & \begin{pmatrix} C^n_{\mathcal{M}^c\cap\mathcal{B}_1,\mathcal{M}\cap\mathcal{B}_1}(C^n_{\mathcal{M}\cap\mathcal{B}_1,\mathcal{M}\cap\mathcal{B}_1})^{-1}\mathrm{sign}(\beta^n_{\mathcal{M}\cap\mathcal{B}_1}) \\ \vdots \\ C^n_{\mathcal{M}^c\cap\mathcal{B}_k,\mathcal{M}\cap\mathcal{B}_k}(C^n_{\mathcal{M}\cap\mathcal{B}_k,\mathcal{M}\cap\mathcal{B}_k})^{-1}\mathrm{sign}(\beta^n_{\mathcal{M}\cap\mathcal{B}_k}) \end{pmatrix}.
\end{aligned}
$$

Corollary 5 is implied immediately from the shown equalities.

References

Z. D. Bai. Methodologies in spectral analysis of large dimensional random matrices: A review. *Statistica Sinica*, (9):611–677, 1999.

D. Donoho, M. Elad, and V. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *Preprint*, 2004.

B. Efron, T. Hastie, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

A.E. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

K. Knight and W. J. Fu. Asymptotics for Lasso-type estimators. *Annals of Statistics*, 28:1356–1378, 2000.

C. Leng, Y. Lin, and G. Wahba. A note on the lasso and related procedures in model selection. *Statistica Sinica*, (To appear), 2004.

N Meinshausen. Lasso with relaxation. *Technical Report*, 2005.

N. Meinshausen and P. Buhlmann. Consistent neighbourhood selection for high-dimensional graphs with the Lasso. *Annals of Statistics*, 34(3), 2006.

M.R. Osborne, B. Presnell, and B.A. Turlach. Knot selection for regression splines via the Lasso. *Computing Science and Statistics*, (30):44–49, 1998.

M.R. Osborne, B. Presnell, and B.A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, (9(2)):319–337, 2000b.

S. Rosset. Tracking curved regularized optimization solution paths. *NIPS*, 2004.

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

P. Zhao and B. Yu. Boosted Lasso. *Technical Report, Statistics Department, UC Berkeley*, 2004.

H. Zou, T. Hastie, and R. Tibshirani. On the "degrees of freedom" of the Lasso. *submitted*, 2004.

Journal of Machine Learning Research 7 (2006) 2565-2583 Submitted 8/05; Revised 9/06; Published 12/06

Stability Properties of Empirical Risk Minimization over Donsker Classes

Andrea Caponnetto CAPONNET@UCHICAGO.EDU
Alexander Rakhlin RAKHLIN@ALUM.MIT.EDU
Center for Biological and Computational Learning
Massachusetts Institute of Technology
Cambridge, MA, 02139, USA

Editor: Leslie Pack Kaelbling

Abstract

We study some stability properties of algorithms which minimize (or almost-minimize) empirical error over Donsker classes of functions. We show that, as the number n of samples grows, the L_2-diameter of the set of almost-minimizers of empirical error with tolerance $\xi(n) = o(n^{-\frac{1}{2}})$ converges to zero in probability. Hence, even in the case of multiple minimizers of expected error, as n increases it becomes less and less likely that adding a sample (or a number of samples) to the training set will result in a large jump to a new hypothesis. Moreover, under some assumptions on the entropy of the class, along with an assumption of Komlos-Major-Tusnady type, we derive a power rate of decay for the diameter of almost-minimizers. This rate, through an application of a uniform ratio limit inequality, is shown to govern the closeness of the expected errors of the almost-minimizers. In fact, under the above assumptions, the expected errors of almost-minimizers become closer with a rate strictly faster than $n^{-1/2}$.

Keywords: empirical risk minimization, empirical processes, stability, Donsker classes

1. Introduction

The empirical risk minimization (ERM) algorithm has been studied in learning theory to a great extent. Vapnik and Chervonenkis (1971, 1991) showed necessary and sufficient conditions for its consistency. In recent developments, Bartlett and Mendelson (2006); Bartlett et al. (2004); Koltchinskii (2006) proved sharp bounds on the performance of ERM. Tools from empirical process theory have been successfully applied, and, in particular, it has been shown that the *localized Rademacher averages* play an important role in studying the behavior of the ERM algorithm.

In this paper we are not directly concerned with rates of performance of ERM. Rather, we prove some properties of ERM algorithms, which, to our knowledge, do not appear in the literature. The analysis of this paper has been motivated by the study of *algorithmic stability*: the behavior of a learning algorithm with respect to perturbations of the training set. Algorithmic stability has been studied in the recent years as an alternative to the classical (complexity-oriented) approach to deriving generalization bounds (Bousquet and Elisseeff, 2002; Kutin and Niyogi, 2002; Mukherjee et al., 2006; Poggio et al., 2004; Rakhlin et al., 2005). Motivation for studying algorithmic stability comes, in part, from the work of Devroye and Wagner (1979). Their results indicate that for any algorithm, the performance of the leave-one-out estimator of expected error is bounded by L_1-stability of the algorithm, that is, by the average L_1 distance between hypotheses on similar samples. This

result can be used to derive bounds on the performance of the leave-one-out estimate for algorithms such as k-Nearest Neighbors. It is important to note that no class of finite complexity is searched by algorithms like k-NN, and so the classical approach of using complexity of the hypothesis space fails.

Further important results were proved by Bousquet and Elisseeff (2002), where a large family of algorithms (*Tikhonov regularization* based methods) has been shown to possess a strong L_∞ stability with respect to changes of single samples of the training set, and exponential bounds have been proved for the generalization error in terms of empirical error. Tikhonov regularization based algorithms minimize the empirical error plus a stabilizer, and are closely related to ERM. Though ERM is not, in general, L_∞-stable, it *is* L_1-stable over certain classes of functions, as one of the results of this paper shows. To the best of our knowledge, the outcomes of the present paper do not follow directly from results available in the machine learning literature. In fact we had to turn to empirical process theory for the mathematical tools necessary for studying stability of ERM.

Various assumptions on the function class, over which ERM is performed, have been considered recently to obtain fast rates on the performance of ERM. The importance of having a unique best function in the class has been shown by Lee et al. (1998): the difficult learning problems seem to be the ones where two minimizers of the expected error exist and are far apart. Although the present paper does not address the question of performance rates, it does shed some light on the behavior of ERM when two (or more) minimizers of expected error exist. Our results imply that, under a certain weak condition on the class, as the expected performance of empirical minimizers approaches the best in the class with the addition of new samples, a jump to a different part of the function class becomes less and less likely.

Since ERM minimizes empirical error instead of expected error, it is reasonable to require that the two quantities become close uniformly over the class, as the number of examples grows. Hence, ERM is a sound strategy only if the function class is uniform Glivenko-Cantelli, that is, it satisfies the uniform law of large numbers. In this paper we focus our attention on a more restricted family of function classes: Donsker classes (see for example, Dudley, 1999). These are classes satisfying not only the law of large numbers, but also a version of the central limit theorem. Though a more restricted family of classes, Donsker classes are still quite general. In particular, uniform Donsker and uniform Glivenko-Cantelli properties are equivalent in the case of binary-valued functions (and also equivalent to finiteness of VC dimension). The central limit theorem for Donsker classes states a form of convergence of the empirical process to a Gaussian process with a specific covariance structure (see for example, Dudley, 1999; van der Vaart and Wellner, 1996). This structure is used in the proof of the main result of the paper to control the correlation of the empirical errors of ERM minimizers on similar samples.

The paper is organized as follows. In Section 2 we introduce the notation and background results. Section 3 presents the main result of the paper, which is proved in the appendix using tools from empirical process theory. In Section 4, we show L_1-stability of ERM over Donsker classes as an application of the main result of Section 3. In Section 5 we show an improvement (in terms of the rates) of the main result under a suitable Komlos-Major-Tusnady condition and an assumption on entropy growth. Section 6 combines the results of Sections 4 and 5 and uses a uniform ratio limit theorem to obtain fast rates of decay on the deviations of expected errors of almost-ERM solutions, thus establishing *strong expected error stability* of ERM (see Mukherjee et al., 2006). Section 7 is a final summary of the results of the paper. Most of the proofs are postponed to the Appendix.

2. Notation and Background Results

Let $(\mathcal{Z},\mathcal{A})$ be a measurable space. Let P be a probability measure on $(\mathcal{Z},\mathcal{A})$ and $Z_1,\ldots,Z_n$ be independent copies of Z with distribution P. Let $\mathcal{F}$ be a class of functions from $\mathcal{Z}$ to $\mathbb{R}$. In the setting of learning theory, samples Z are input-output pairs (X,Y) and for $f\in\mathcal{F}$, $f(Z)$ measures how well the relationship between X and Y is captured by f. The goal is to minimize $Pf=\mathbb{E}f(Z)$ where information about the unknown P is given only through the finite sample $S=(Z_1,\ldots,Z_n)$. Define the empirical measure as $P_n=\frac{1}{n}\sum_{i=1}^n\delta_{Z_i}$.

Definition 1 *Given a sample S,*

$$f_S:=\operatorname*{argmin}_{f\in\mathcal{F}}P_nf=\operatorname*{argmin}_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^n f(Z_i)$$

is a minimizer of the empirical risk (empirical error), if the minimum exists.

Since an exact minimizer of the empirical risk might not exist, as well as for algorithmic reasons, we consider the set of almost-minimizers of empirical risk.

Definition 2 *Given $\xi\geq 0$ and S, define the set of almost empirical minimizers*

$$\mathcal{M}_S^\xi=\{f\in\mathcal{F}:P_nf-\inf_{g\in\mathcal{F}}P_ng\leq\xi\}$$

and define its diameter as

$$\operatorname{diam}\mathcal{M}_S^\xi=\sup_{f,g\in\mathcal{M}_S^\xi}\|f-g\|.$$

The $\|\cdot\|$ in the above definition is the seminorm on $\mathcal{F}$ induced by symmetric bilinear product

$$\langle f,f'\rangle=P\left((f-Pf)\left(f'-Pf'\right)\right),$$

hence $\|f\|$ is the standard deviation of f relative to P.

This is a natural measure of distance between functions, as will become apparent later, because of the central role of the covariance structure of Brownian bridges in our proofs. The results obtained for the seminorm $\|\cdot\|$ will be easily extended to the $L_2(P)$ norm, thanks to the close relation of these two notions of distance.

Definition 3 *The empirical process ν_n indexed by $\mathcal{F}$ is defined as the map*

$$f\mapsto\nu_n(f)=\sqrt{n}(P_n-P)f=\frac{1}{\sqrt{n}}\sum_{i=1}^n(f(Z_i)-Pf).$$

Definition 4 *A class $\mathcal{F}$ is called P-Donsker if*

$$\nu_n\rightsquigarrow\nu$$

in $\ell^\infty(\mathcal{F})$, where the limit ν is a tight Borel measurable element in $\ell^\infty(\mathcal{F})$ and "$\rightsquigarrow$" denotes weak convergence, as defined on p. 17 of van der Vaart and Wellner (1996).

In fact, it follows that the limit process ν must be a zero-mean Gaussian process with covariance function $\mathbb{E}\nu(f)\nu(f') = \langle f, f' \rangle$ (i.e., a Brownian bridge).

Various Donsker theorems provide sufficient conditions for a class being P-Donsker. Here we mention a few known results (see van der Vaart and Wellner 1996, Equation 2.1.7 and van de Geer 2000, Theorem 6.3) in terms of entropy and entropy with bracketing, which we define below (see van der Vaart and Wellner, 1996).

Definition 5 *The covering number $\mathcal{N}(\varepsilon, \mathcal{F}, \|\cdot\|)$ is the minimal number of balls $\{g : \|g - f\| < \varepsilon\}$ of radius ε needed to cover the set $\mathcal{F}$. The centers of the balls need not belong to $\mathcal{F}$, but they should have finite norms. The entropy is the logarithm of the covering number.*

Definition 6 *Given two functions l and u, the bracket $[l,u]$ is the set of all functions f with $l \leq f \leq u$. An ε-bracket is a bracket $[l,u]$ with $\|u - l\| < \varepsilon$. The bracketing number $\mathcal{N}_{[]}(\varepsilon, \mathcal{F}, \|\cdot\|)$ is the minimum number of ε-brackets needed to cover $\mathcal{F}$. The upper and lower bounds u and l need not belong to $\mathcal{F}$ but are assumed to have finite norms. The entropy with bracketing is the logarithm of the bracketing number.*

Definition 7 *An envelope function of a class $\mathcal{F}$ is any function $x \mapsto F(x)$ such that $|f(x)| \leq F(x)$ for every x and $f \in \mathcal{F}$.*

Proposition 8 *If the envelope F of $\mathcal{F}$ is square integrable and*

$$\int_0^\infty \sup_Q \sqrt{\log \mathcal{N}(\varepsilon \|F\|_{Q,2}, \mathcal{F}, L_2(Q))} d\varepsilon < \infty,$$

then $\mathcal{F}$ is P-Donsker for every P, that is, $\mathcal{F}$ is a universal Donsker class. Here the supremum is taken over all finitely discrete probability measures, and the $L_2(Q)$-norm is defined as $\|f\|_{Q,2} = \left(\int |f|^2\right)^{1/2}$.

Proposition 9 *If $\int_0^\infty \sqrt{\log \mathcal{N}_{[]}(\varepsilon, \mathcal{F}, L_2(P))} d\varepsilon < \infty$, then $\mathcal{F}$ is P-Donsker.*

From the learning theory perspective, however, the most interesting theorems are probably those relating the Donsker property to the VC-dimension. For example, if $\mathcal{F}$ is a $\{0,1\}$-valued class, then $\mathcal{F}$ is *universal* Donsker if and only if its VC dimension is finite (Theorem 10.1.4 of Dudley (1999) provides a more general result involving Pollard's entropy condition). As a corollary of their Proposition 3.1, Giné and Zinn (1991) show that under the Pollard's entropy condition, the $\{0,1\}$-valued class $\mathcal{F}$ is in fact *uniform* Donsker. Finally, Rudelson and Vershynin extended these results to the real-valued case: a class $\mathcal{F}$ is *uniform* Donsker if the square root of its scale-sensitive VC dimension is integrable.

3. Main Result

We now state the main result of this paper.

Theorem 10 *Let $\mathcal{F}$ be a P-Donsker class. For any sequence $\xi(n) = o(n^{-1/2})$,*

$$\mathrm{diam} \mathcal{M}_S^{\xi(n)} \xrightarrow{P^*} 0.$$

The outer probability P^* above is due to measurability issues. Definitions and results on various types of convergence, as well as ways to deal with measurability issues arising in the proofs, are based on the rigorous book of van der Vaart and Wellner (1996).

The following corollary, whose proof is given in Appendix A, extends the above result to L_2 (and thus L_1) diameters.

Corollary 11 *The result of Theorem 10 holds if the diameter is defined with respect to the $L_2(P)$ norm.*

It is easy to verify that the dependence $\xi(n) = o(n^{-1/2})$ of the tolerance, assumed in Theorem 10, is not improvable. In fact a simple example can show that if $\xi(n) \asymp n^{-1/2}$ the set of $\xi(n)$-almost minimizers may not shrink in probability.

Example 1 *Consider $\mathcal{Z} = \{x_1, x_2\}$ with $x_1 \neq x_2$, and $P = \frac{1}{2}(\delta_{x_1} + \delta_{x_2})$. Moreover let $\mathcal{F}$ be the set of functions $\{f_1, f_2\}$, with*

$$f_i(x) = \begin{cases} 0 & \text{if} \quad x = x_i, \\ 1 & \text{otherwise}. \end{cases}$$

Then it is clear that, given the finite sample $S = (Z_1, \ldots, Z_n)$, $\mathcal{M}_S^{\xi} = \mathcal{F}$ (and hence $\operatorname{diam}\mathcal{M}_S^{\xi(n)} = 1$) whenever $P_n f_1 - P_n f_2 = \frac{2}{n}|q - \mathbb{E}q| \leq \xi(n)$, where q is the binomial random variable

$$q = \#\{i | Z_i = x_1\}.$$

Now since the variance of q is $\frac{n}{4}$, it is clear that

$$\forall C > 0 \qquad \Pr\left\{|q - \mathbb{E}q| \leq C n^{\frac{1}{2}}\right\} = \Omega(1),$$

which shows that, if $\xi(n) \asymp n^{-1/2}$, with probability bounded away from zero, $\operatorname{diam}\mathcal{M}_S^{\xi(n)} = 1$.

The above example is very basic, yet provides important intuition. A class can contain two quite different functions with the smallest expectation, but it is unlikely that they both almost-minimize the empirical error to within $o(n^{-1/2})$. In fact, the above example suggests that the fluctuations of the difference in empirical performance of two functions is of the order $n^{-1/2}$. The extension of this result to more general function classes with possibly infinite number of expected minima is the main goal of Theorem 10.

Before diving into the proof of Theorem 10, let us state a few notions of stochastic convergence.

Definition 12 (Definition 1.9.1 in van der Vaart and Wellner (1996)) *Let $(\mathcal{Z}, \mathcal{A}, P)$ be a probability space. Let $Z_n, Z : \mathcal{Z} \mapsto \mathcal{D}$ be arbitrary maps and $(\mathcal{D}, d)$ be a metric space.*

- *Z_n converges in outer probability to Z if $d(Z_n, Z)^* \to 0$ in probability; this means that $P(d(Z_n, Z)^* > \varepsilon) = P^*(d(Z_n, Z) > \varepsilon) \to 0$, for every $\varepsilon > 0$, and is denoted by $Z_n \xrightarrow{P^*} 0$.*
- *Z_n converges almost uniformly to Z if, for every $\varepsilon > 0$, there exists a measurable set A with $P(A) \geq 1 - \varepsilon$ and $d(Z_n, Z) \to 0$ uniformly on A; this is denoted $Z_n \xrightarrow{au} Z$.*

The proof of Theorem 10 relies on the *almost sure representation theorem* (van der Vaart and Wellner, 1996, Theorem 1.10.4). Here we state the theorem applied to ν_n and ν.

Proposition 13 *Suppose $\mathcal{F}$ is P-Donsker. Let $\nu_n : \mathcal{Z}^n \mapsto \ell^\infty(\mathcal{F})$ be the empirical process. There exist a probability space $(\mathcal{Z}', \mathcal{A}', P')$ and maps $\nu', \nu'_n : \mathcal{Z}' \mapsto \ell^\infty(\mathcal{F})$ such that*

1. $\nu'_n \xrightarrow{au} \nu'$,
2. $\mathbb{E}^* f(\nu'_n) = \mathbb{E}^* f(\nu_n)$ *for every bounded* $f : \ell^\infty(\mathcal{F}) \mapsto \mathbb{R}$ *for all n.*

Lemma 14 is the main preliminary result used in the proof of Theorem 10 (and Theorem 17 in Section 5). We postpone its proof to Appendix A.

Lemma 14 *Let $\nu_n : \mathcal{Z}^n \mapsto \ell^\infty(\mathcal{F})$ be the empirical process. Fix n and assume that there exist a probability space $(\mathcal{Z}', \mathcal{A}', P')$ and a map $\nu'_n : \mathcal{Z}' \mapsto \ell^\infty(\mathcal{F})$ such that $\mathbb{E}^* f(\nu'_n) = \mathbb{E}^* f(\nu_n)$ for every bounded $f : \ell^\infty(\mathcal{F}) \mapsto \mathbb{R}$. Let ν' be a P-Brownian bridge defined on $(\mathcal{Z}', \mathcal{A}', P')$. Fix $C > 0$, $\varepsilon = \min(C^3/128, C/4)$ and suppose $\delta \geq \xi\sqrt{n}$ for a given $\xi > 0$. Then, if $\mathcal{F}$ is P-Donsker, the following inequality holds*

$$\Pr^*\left(\operatorname{diam}\mathcal{M}_S^\xi > C\right) \leq \mathcal{N}(\varepsilon, \mathcal{F}, \|\cdot\|)^2 \left(\frac{128\delta}{C^3} + \Pr^*\left(\sup_{\mathcal{F}} \left|\nu'_n - \nu'\right| \geq \delta/2\right)\right).$$

We are now ready to prove the main result of this section.

Proof [Theorem 10] Lemma 1.9.3 in van der Vaart and Wellner (1996) shows that when the limiting process is Borel measurable, almost uniform convergence implies convergence in outer probability. Therefore, the first implication of Proposition 13 states that for any $\delta > 0$

$$\Pr^*\left(\sup_{\mathcal{F}} |\nu'_n - \nu'| > \delta\right) \to 0.$$

By Lemma 14,

$$\Pr^*\left(\operatorname{diam}\mathcal{M}_S^{\xi(n)} > C\right) \leq \mathcal{N}(\varepsilon, \mathcal{F}, \|\cdot\|)^2 \left(\frac{128\delta}{C^3} + \Pr^*\left(\sup_{\mathcal{F}} \left|\nu'_n - \nu'\right| \geq \delta/2\right)\right)$$

for any $C > 0$, $\varepsilon = \min(C^3/128, C/4)$, and any $\delta \geq \xi(n)\sqrt{n}$. Since $\xi(n) = o(n^{-1/2})$, δ can be chosen arbitrarily small, and so $\Pr^*\left(\operatorname{diam}\mathcal{M}_S^{\xi(n)} > C\right) \to 0$. ■

4. Stability of almost-ERM

The main result of this section, Corollary 15, shows L_2-*stability* of almost-ERM on Donsker classes. It implies that, in probability, the L_2 (and thus L_1) distance between almost-minimizers on similar training sets (with $o(\sqrt{n})$ changes) goes to zero when n tends to infinity.

This result provides a partial answer to the questions raised in the machine learning literature by Kutin and Niyogi (2002); Mukherjee et al. (2006): is it true that when one point is added to the training set, the ERM algorithm is less and less likely to jump to a far (in the L_1 sense) hypothesis? In fact, since binary-valued function classes are uniform Donsker if and only if the VC dimension is finite, Corollary 15 proves that almost-ERM over binary VC classes possesses L_1-stability. For the

real-valued classes, uniform Glivenko-Cantelli property is weaker than uniform Donsker property, and therefore it remains unclear if almost-ERM over uGC but not uniform Donsker classes is stable in the L_1 sense.

Use of L_1-stability goes back to Devroye and Wagner (1979), who showed that this stability is sufficient to bound the difference between the leave-one-out error and the expected error of a learning algorithm. In particular, Devroye and Wagner show that nearest-neighbor rules possess L_1-stability (see also Devroye et al., 1996). Our Corollary 15 implies L_1-stability of ERM (or almost-ERM) algorithms on Donsker classes.

In the following $[n]$ denotes the set $\{1,2,\dots,n\}$ and $A \triangle B$ is the symmetric difference of sets A and B.

Corollary 15 *Assume $\mathcal{F}$ is P-Donsker and uniformly bounded with envelope $F \equiv 1$. For $I \subset \mathbb{N}$, define $S(I) = (Z_i)_{i\in I}$. Let $I_n \subset \mathbb{N}$ such that $M_n := |I_n \triangle [n]| = o(n^{1/2})$. Suppose $f_n \in \mathcal{M}_{S([n])}^{\xi(n)}$ and $f_n' \in \mathcal{M}_{S(I_n)}^{\xi'(n)}$ for some $\xi(n) = o(n^{-1/2})$ and $\xi'(n) = o(n^{-1/2})$. Then*

$$\left\| f_n - f_n' \right\| \xrightarrow{P^*} 0.$$

The norm $\|\cdot\|$ can be replaced by $L_2(P)$ or $L_1(P)$ norm.

Proof It is enough to show that $f_n' \in \mathcal{M}_{S([n])}^{\xi''(n)}$ for some $\xi''(n) = o(n^{-1/2})$ and result follows from Theorem 10.

$$\begin{aligned}
\frac{1}{n}\sum_{i\in[n]} f_n'(Z_i) &\le \frac{M_n}{n} + \frac{1}{n}\sum_{i\in I_n} f_n'(Z_i) \\
&\le \frac{M_n}{n} + \frac{|I_n|}{n}\left(\xi'(n) + \inf_{g\in\mathcal{F}} \frac{1}{|I_n|}\sum_{i\in I_n} g(Z_i)\right) \\
&\le \frac{M_n}{n} + \frac{|I_n|}{n}\xi'(n) + \frac{1}{n}\sum_{i\in I_n} f_n(Z_i) \\
&\le 2\frac{M_n}{n} + \frac{|I_n|}{n}\xi'(n) + \frac{1}{n}\sum_{i\in[n]} f_n(Z_i) \\
&\le 2\frac{M_n}{n} + \frac{|I_n|}{n}\xi'(n) + \xi(n) + \inf_{g\in\mathcal{F}} \frac{1}{n}\sum_{i\in[n]} g(Z_i).
\end{aligned}$$

Define

$$\xi''(n) := 2\frac{M_n}{n} + \frac{|I_n|}{n}\xi'(n) + \xi(n).$$

Because $M_n = o(n^{\frac{1}{2}})$, it follows that $\xi''(n) = o(n^{-1/2})$. Corollary 11 implies convergence in $L_2(P)$, and, therefore, in $L_1(P)$ norm. ■

5. Rates of Decay of $\operatorname{diam}\mathcal{M}_S^{\xi(n)}$

The statement of Lemma 14 reveals that the rate of the decay of the diameter $\operatorname{diam}\mathcal{M}_S^{\xi(n)}$ is related to the rate at which $\Pr^*\left(\sup_{\mathcal{F}} |\nu - \nu_n| \ge \delta\right) \to 0$ for a fixed δ. A number of papers studied this

rate of convergence, and here we refer to the notion of *Komlos-Major-Tusnady class* (KMT class), as defined by Koltchinskii (1994). Let $\nu'_n : \mathcal{Z}^n \mapsto \ell^\infty(\mathcal{F})$ be the empirical process defined on the probability space $(\mathcal{Z}', \mathcal{A}', P')$.

Definition 16 *$\mathcal{F}$ is called a Komlos-Major-Tusnady class with respect to P and with the rate of convergence τ_n ($\mathcal{F} \in KMT(P;\tau_n)$) if $\mathcal{F}$ is P-pregaussian and for each $n \geq 1$ there is a version $\nu^{(n)}$ of a P-Brownian bridge defined on $(\mathcal{Z}', \mathcal{A}', P')$ such that for all $t > 0$,*

$$\Pr^* \left(\sup_{\mathcal{F}} |\nu^{(n)} - \nu'_n| \geq \tau_n(t + K \log n) \right) \leq \Lambda e^{-\theta t}$$

where $K > 0$, $\Lambda > 0$ and $\theta > 0$ are constants, depending only on $\mathcal{F}$.

Sufficient conditions for a class to be $KMT(P; n^{-\alpha})$ have been investigated in the literature; some results of this type can be found in Koltchinskii (1994); Rio (1993) and Dudley (2002), Section 9.5(B).

The following theorem shows that for KMT classes fulfilling a suitable entropy condition, it is possible to give explicit rates of decay for the diameter of ERM almost-minimizers.

Theorem 17 *Assume $\mathcal{F}$ is P-Donsker and $\mathcal{F} \in KMT(P; n^{-\alpha})$ for some $\alpha > 0$. Assume $\mathcal{N}(\varepsilon, \mathcal{F}, \|\cdot\|) \leq \left(\frac{A}{\varepsilon}\right)^V$ for some constants $A, V > 0$. Let $\xi(n)\sqrt{n} = o(n^{-\eta})$, $\eta > 0$. Then*

$$n^\gamma \mathrm{diam} \mathcal{M}_S^{\xi(n)} \xrightarrow{P^*} 0$$

for any $\gamma < \frac{1}{3(2V+1)} \min(\alpha, \eta)$.

Proof The result of Lemma 14 is stated for a fixed n. We now choose C, ξ, and δ depending on n as follows. Let $C(n) = Bn^{-\gamma}$, where $\gamma < \frac{1}{3(2V+1)} \min(\alpha, \eta)$ and $B > 0$ is an arbitrary constant. Let $\xi = \xi(n)$. Let $\delta(n) = n^{-\beta}$, where $\beta = \frac{1}{2}(\min(\alpha, \eta) + 3(2V+1)\gamma)$. When β is defined this way, we have

$$\min(\alpha, \gamma) > \beta > 3(2V+1)\gamma$$

because $\gamma < \frac{1}{3(2V+1)} \min(\alpha, \eta)$ by assumption. In particular, $\beta < \eta$ and, hence, eventually $\delta(n) > \xi(n)\sqrt{(n)} = o(n^{-\eta})$.

Since $C(n)$ decays to zero and $\varepsilon(n) = \min(C(n)^3/128, C(n)/4)$, eventually $\varepsilon(n) = C(n)^3/128 = n^{-3\gamma} B^3/128$.

Since $\mathcal{F} \in KMT(P; n^{-\alpha})$,

$$\Pr^* \left(\sup_{\mathcal{F}} |\nu^{(n)} - \nu_n| \geq n^{-\alpha}(t + K \log n) \right) \leq \Lambda e^{-\theta t}$$

for any $t > 0$, choosing $t = n^\alpha \delta(n)/2 - K \log n$ we obtain

$$\Pr^* \left(\sup_{\mathcal{F}} |\nu^{(n)} - \nu_n| \geq \delta(n)/2 \right) \leq \Lambda e^{-\theta(n^{\alpha-\beta}/2 - K \log n)}.$$

Lemma 14 then implies

$$\begin{aligned}
&\Pr^*\left(\operatorname{diam}\mathcal{M}_S^{\xi(n)} > C(n)\right) \le \mathcal{N}(\varepsilon,\mathcal{F},\|\cdot\|)^2\left(\frac{128\delta}{C(n)^3} + \Pr^*\left(\sup_{\mathcal{F}}\left|\nu_n' - \nu'\right| \ge \delta/2\right)\right)\\
&\le \left(\frac{128A}{B^3}n^{3\gamma}\right)^{2V}\frac{128}{B^3}n^{-\beta}n^{3\gamma} + \left(\frac{128A}{B^3}n^{3\gamma}\right)^{2V}\Lambda e^{-\theta(n^{\alpha-\beta}/2 - K\log n)}\\
&= \left(\frac{128A}{B^3}\right)^{2V}\frac{128}{B^3}n^{3\gamma(2V+1)-\beta} + \Lambda\left(\frac{128A}{B^3}\right)^{2V} n^{k\theta+6\gamma V}e^{-\frac{\theta}{2}n^{\alpha-\beta}}.
\end{aligned}$$

Since $\alpha > \beta > 3\gamma(2V+1)$, both terms above go to zero, that is,

$$\Pr^*\left(n^{\gamma}\operatorname{diam}\mathcal{M}_S^{\xi(n)} > B\right) \to 0 \text{ for any } B > 0.$$

■

The entropy condition in Theorem 17 is clearly verified by VC-subgraph classes of dimension V. In fact, since L_2 norm dominates $\|\cdot\|$ seminorm, upper bounds on L_2 covering numbers of VC-subgraph classes induce analogous bounds on $\|\cdot\|$ covering numbers. Corollary 18 is a an application of Theorem 17 to this important family of classes. It follows in a straight-forward way from the remark above.

Corollary 18 *Assume $\mathcal{F}$ is a VC-subgraph class with VC-dimension V, and for some $\alpha > 0$ $\mathcal{F} \in KMT(P, n^{-\alpha})$. Let $\xi(n)\sqrt{n} = o(n^{-\eta})$, $\eta > 0$. Then*

$$n^{\gamma}\operatorname{diam}\mathcal{M}_S^{\xi(n)} \xrightarrow{P^*} 0$$

for any $\gamma < \frac{1}{3(2V+1)}\min(\alpha,\eta)$.

6. Expected Error Stability of almost-ERM

In the previous section, we proved bounds on the rate of decay of the diameter of almost-minimizers. In this section, we show that given such a bound, as well as some additional conditions on the class, the differences between *expected errors* of almost-minimizers decay faster than $n^{-1/2}$. This implies a form of *strong expected error stability* for ERM.

The proof of Theorem 20 relies on the following ratio inequality of Pollard (1995).

Proposition 19 *Let $\mathcal{G}$ be a uniformly bounded function class with the envelope function $G \equiv 2$. Assume $\mathcal{N}(\gamma,\mathcal{G}) = \sup_Q \mathcal{N}(2\gamma,\mathcal{G},L_1(Q)) < \infty$ for $0 < \gamma \le 1$ and Q ranging over all discrete probability measures. Then*

$$\Pr^*\left(\sup_{f\in\mathcal{G}}\frac{|P_n f - Pf|}{\varepsilon(P_n|f| + P|f|) + 5\gamma} > 26\right) \le 32\mathcal{N}(\gamma,\mathcal{G})\exp(-n\varepsilon\gamma).$$

The next theorem gives explicit rates for expected error stability of ERM over VC-subgraph classes fulfilling a KMT type condition.

Theorem 20 *If $\mathcal{F}$ is a VC-subgraph class with VC-dimension V, $\mathcal{F} \in KMT(P; n^{-\alpha})$ and $\sqrt{n}\xi(n) = o(n^{-\eta})$, then for any $\kappa < \min\left(\frac{1}{6(2V+1)}\min(\alpha,\eta), 1/2\right)$*

$$n^{1/2+\kappa} \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} |P(f-f')| \xrightarrow{P^*} 0.$$

7. Conclusions

We presented some new results establishing stability properties of ERM over certain classes of functions. This study was motivated by the question, raised by some recent papers, of L_1-stability of ERM under perturbations of a single sample (Mukherjee et al., 2006; Kutin and Niyogi, 2002; Rakhlin et al., 2005). We gave a partially positive answer to this question, proving that, in fact, ERM over Donsker classes fulfills L_2-stability (and hence also L_1-stability) under perturbations of $o(n^{\frac{1}{2}})$ among the n samples of the training set. This property follows directly from the main result of the paper which shows decay (in probability) of the diameter of the set of solutions of almost-ERM with tolerance function $\xi(n) = o(n^{-\frac{1}{2}})$. We stress that for classification problems (i.e., for binary-valued functions) no generality is lost in assuming the Donsker property, since for ERM to be a sound algorithm, the equivalent Glivenko-Cantelli property has to be assumed anyway. On the other hand, in the real-valued case many complexity-based characterizations of Donsker property are available in the literature.

In the perspective of possible algorithmic applications, we analyzed some additional assumptions implying uniform rates on the decay of the L_1 diameter of almost-minimizers. It turned out that an explicit rate of this type can be given for VC-subgraph classes satisfying a suitable Komlos-Major-Tusnady type condition. For this condition, many independent characterizations are known.

Finally, using a suitable ratio inequality we showed how L_1-stability results can induce strong forms of expected error stability, providing a further insight into the behavior of the Empirical Risk Minimization algorithm.

Results of this paper can be used to analyze stability of a class of clustering algorithms by casting them in the empirical risk minimization framework (see Rakhlin and Caponnetto, 2006).

Algorithmic implications of our results would require further investigation. For example, in the context of on-line learning, when a point is added to the training set, with high probability one would only have to search for empirical minimizers in a small L_1-ball around the current hypothesis, which might be a tractable problem. Moreover, L_1-stability might have consequences for computational complexity of ERM. While it has been shown that ERM is NP-hard even for simple function classes (see for example, Ben-David et al., 2003), our results could allow more optimistic average-case analysis.

Acknowledgments

We would like to thank S. Mukherjee, T. Poggio and S. Smale for useful discussions and suggestions.

This report describes research done at the Center for Biological & Computational Learning, which is in the McGovern Institute for Brain Research at MIT, as well as in the Dept. of Brain & Cognitive Sciences, and which is affiliated with the Computer Sciences & Artificial Intelligence

Laboratory (CSAIL), as well as in the Dipartimento di Informatica e Scienze dell'Informazione (DISI) at University of Genoa, Italy. This research was sponsored by grants from: Office of Naval Research (DARPA) Contract No. MDA972-04-1-0037, Office of Naval Research (DARPA) Contract No. N00014-02-1-0915, National Science Foundation (ITR/SYS) Contract No. IIS-0112991, National Science Foundation (ITR) Contract No. IIS-0209289, National Science Foundation-NIH (CRCNS) Contract No. EIA-0218693, National Science Foundation-NIH (CRCNS) Contract No. EIA-0218506, and National Institutes of Health (Conte) Contract No. 1 P20 MH66239-01A1. Additional support was provided by: Central Research Institute of Electric Power Industry (CRIEPI), Daimler-Chrysler AG, Compaq/Digital Equipment Corporation, Eastman Kodak Company, Honda R&D Co., Ltd., Industrial Technology Research Institute (ITRI), Komatsu Ltd., Eugene McDermott Foundation, Merrill-Lynch, NEC Fund, Oxygen, Siemens Corporate Research, Inc., Sony, Sumitomo Metal Industries, and Toyota Motor Corporation. This research has also been partially funded by the FIRB Project ASTAA and the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

Appendix A.

In this appendix we derive some results presented in Section 3. In particular, Lemma 14, which was used in the proof of Theorem 10, and Corollary 11. Let us start with some technical Lemmas.

Lemma 21 *Let* $f_0, f_1 \in \mathcal{F}$, $\|f_0 - f_1\| \geq C/2$, $\|f_1\| \leq \|f_0\|$. *Let* $h : \mathcal{F} \to \mathbb{R}$ *be defined as* $h(f') = \frac{\langle f', f_0 \rangle}{\|f_0\|^2}$. *Then for any* $\varepsilon \leq \frac{C^3}{128}$

$$\inf_{\mathcal{B}(f_0,\varepsilon)} h - \sup_{\mathcal{B}(f_1,\varepsilon)} h \geq \frac{C^2}{16}.$$

Proof

$$\begin{aligned}
\Delta &:= \inf_{\mathcal{B}(f_0,\varepsilon)} h - \sup_{\mathcal{B}(f_1,\varepsilon)} h \\
&= h(f_0) - h(f_1) + \inf\{h(f' - f_0) + h(f_1 - f'') \,|\, f' \in \mathcal{B}(f_0,\varepsilon), f'' \in \mathcal{B}(f_1,\varepsilon)\} \\
&\geq h(f_0) - h(f_1) - \frac{2\varepsilon}{\|f_0\|} \geq h(f_0) - h(f_1) - \frac{8\varepsilon}{C},
\end{aligned}$$

since $\|f_0\| \geq C/4$.

Finally

$$2\langle f_0 - f_1, f_0 \rangle = \|f_0 - f_1\|^2 - \|f_1\|^2 + \|f_0\|^2 \geq \|f_0 - f_1\|^2 \geq \frac{C^2}{4},$$

then

$$h(f_0) - h(f_1) \geq \frac{C^2}{8\|f_0\|^2} \geq \frac{C^2}{8},$$

which proves that

$$\Delta \geq \frac{C^2}{8} - \frac{8\varepsilon}{C} \geq \frac{C^2}{16}.$$

■

The following Lemma is an adaptation of Lemma 2.3 of Kim and Pollard (1990).

Lemma 22 *Let f_0, f_1, h be defined as in Lemma 21. Suppose $\varepsilon \leq \frac{C^3}{128}$. Let ν_μ be a Gaussian process on $\mathcal{F}$ with mean μ and covariance* $\mathrm{cov}(\nu_\mu(f), \nu_\mu(f')) = \langle f, f' \rangle$.

Then for all $\delta > 0$

$$\Pr^* \left(|\sup_{\mathcal{B}(f_0,\varepsilon)} \nu_\mu - \sup_{\mathcal{B}(f_1,\varepsilon)} \nu_\mu| \leq \delta \right) \leq \frac{64\delta}{C^3}.$$

Proof Define the Gaussian process $Y(\cdot) = \nu_\mu(\cdot) - h(\cdot)\nu_\mu(f_0)$. Since $\mathrm{cov}(Y(f'), \nu_\mu(f_0)) = \langle f', f_0 \rangle - h(f') \|f_0\|^2 = 0$, $\nu_\mu(f_0)$ and $Y(\cdot)$ are independent.

We now reason conditionally with respect to $Y(\cdot)$. Define

$$\Gamma_i(z) = \sup_{\mathcal{B}(f_i,\varepsilon)} \{Y(\cdot) + h(\cdot)z\} \qquad \text{with } i = 0, 1.$$

Notice that

$$\Pr^* \left(|\sup_{\mathcal{B}(f_0,\varepsilon)} \nu_\mu - \sup_{\mathcal{B}(f_1,\varepsilon)} \nu_\mu| \leq \delta | Y \right) = \Pr^* \left(|\Gamma_0(\nu_\mu(f_0)) - \Gamma_1(\nu_\mu(f_0))| \leq \delta \right).$$

Moreover Γ_0 and Γ_1 are convex and

$$\inf \partial_- \Gamma_0 - \sup \partial_+ \Gamma_1 \geq \inf_{\mathcal{B}(f_0,\varepsilon)} h - \sup_{\mathcal{B}(f_1,\varepsilon)} h \geq \frac{C^2}{16},$$

by Lemma 21. Then $\Gamma_0 = \Gamma_1$ in a single point z_0 and

$$\Pr^* \left(|\Gamma_0(\nu_\mu(f_0)) - \Gamma_1(\nu_\mu(f_0))| \leq \delta \right) \leq \Pr^* \left(\nu_\mu(f_0) \in [z_0 - \Delta, z_0 + \Delta] \right),$$

with $\Delta = 16\delta/C^2$.

Furthermore,

$$\Pr^* \left(\nu_\mu(f_0) \in [z_0 - \Delta, z_0 + \Delta] \right) \leq \frac{32\delta}{C^2 \sqrt{2\pi \mathrm{var}(\nu_\mu(f_0))}},$$

and $\mathrm{var}(\nu_\mu(f_0)) = \|f_0\|^2 \geq C^2/16$, which completes the proof. ■

The reasoning in the proof of the next lemma goes as follows. We consider a finite cover of $\mathcal{F}$. Pick any two almost-minimizers which are far apart. They belong to two covering balls with centers far apart. Because the two almost-minimizers belong to these balls, the infima of the empirical risks over these two balls are close. This is translated into the event that the suprema of the shifted empirical process over these two balls are close. By looking at the Gaussian limit process, we are able to exploit the covariance structure to show that the suprema of the Gaussian process over balls with centers far apart are unlikely to be close.

Proof [Lemma 14]

Consider the ε-covering $\{f_i | i = 1, \ldots, \mathcal{N}(\varepsilon, \mathcal{F}, \|\cdot\|)\}$. Such a covering exists because $\mathcal{F}$ is totally bounded in $\|\cdot\|$ norm (see page 89, van der Vaart and Wellner, 1996). For any $f, f' \in \mathcal{M}_S^\xi$ s.t. $\|f - f'\| > C$, there exist k and l such that $\|f - f_k\| \leq \varepsilon \leq C/4$, $\|f' - f_l\| \leq \varepsilon \leq C/4$. By triangle inequality it follows that $\|f_k - f_l\| \geq C/2$.

Moreover

$$\inf_{\mathcal{F}} P_n \leq \inf_{\mathcal{B}(f_k,\varepsilon)} P_n \leq P_n f \leq \inf_{\mathcal{F}} P_n + \xi$$

and

$$\inf_{\mathcal{F}} P_n \leq \inf_{\mathcal{B}(f_l,\varepsilon)} P_n \leq P_n f' \leq \inf_{\mathcal{F}} P_n + \xi.$$

Therefore,

$$\left| \inf_{\mathcal{B}(f_k,\varepsilon)} P_n - \inf_{\mathcal{B}(f_l,\varepsilon)} P_n \right| \leq \xi.$$

The last relation can be restated in terms of the empirical process ν_n:

$$\left| \sup_{\mathcal{B}(f_k,\varepsilon)} \{-\nu_n - \sqrt{n}P\} - \sup_{\mathcal{B}(f_l,\varepsilon)} \{-\nu_n - \sqrt{n}P\} \right| \leq \xi\sqrt{n} \leq \delta.$$

$$\Pr^*\left(\operatorname{diam}\mathcal{M}_S^\xi > C\right) = \Pr^*\left(\exists f, f' \in \mathcal{M}_S^\xi, \|f - f'\| > C\right) \leq$$
$$\Pr^*\left(\exists l, k \quad \text{s.t. } \|f_k - f_l\| \geq C/2, \left| \sup_{\mathcal{B}(f_k,\varepsilon)} \{-\nu_n - \sqrt{n}P\} - \sup_{\mathcal{B}(f_l,\varepsilon)} \{-\nu_n - \sqrt{n}P\} \right| \leq \delta\right).$$

By union bound

$$\Pr^*\left(\operatorname{diam}\mathcal{M}_S^\xi > C\right)$$
$$\leq \sum_{\substack{k,l=1 \\ \|f_k - f_l\| \geq C/2}}^{\mathcal{N}(\varepsilon,\mathcal{F},\|\cdot\|)} \Pr^*\left(\left| \sup_{\mathcal{B}(f_k,\varepsilon)} \{-\nu_n - \sqrt{n}P\} - \sup_{\mathcal{B}(f_l,\varepsilon)} \{-\nu_n - \sqrt{n}P\} \right| \leq \delta\right).$$

We now want to bound the terms in the sum above. Assuming without loss of generality that $\|f_k\| \geq \|f_l\|$, we obtain

$$\begin{aligned}
&\Pr^*\left(\left|\sup_{\mathcal{B}(f_k,\varepsilon)}\{-\nu_n-\sqrt{n}P\}-\sup_{\mathcal{B}(f_l,\varepsilon)}\{-\nu_n-\sqrt{n}P\}\right|\leq\delta\right)\\
&=\Pr^*\left(\left|\sup_{\mathcal{B}(f_k,\varepsilon)}\{-\nu'_n-\sqrt{n}P\}-\sup_{\mathcal{B}(f_l,\varepsilon)}\{-\nu'_n-\sqrt{n}P\}\right|\leq\delta\right)\\
&=\Pr^*\left(\left|\sup_{\mathcal{B}(f_k,\varepsilon)}\{-\nu'-\sqrt{n}P+\nu'-\nu'_n\}-\sup_{\mathcal{B}(f_l,\varepsilon)}\{-\nu'-\sqrt{n}P+\nu'-\nu'_n\}\right|\leq\delta\right)\\
&\leq\Pr^*\left(\left|\left|\sup_{\mathcal{B}(f_k,\varepsilon)}\{-\nu'-\sqrt{n}P\}-\sup_{\mathcal{B}(f_l,\varepsilon)}\{-\nu'-\sqrt{n}P\}\right|-2\sup_{\mathcal{F}}\left|\nu'_n-\nu'\right|\right|\leq\delta\right)\\
&\leq\Pr^*\left(2\sup_{\mathcal{F}}\left|\nu'_n-\nu'\right|\geq\delta\ \vee\ \left|\sup_{\mathcal{B}(f_k,\varepsilon)}\{-\nu'-\sqrt{n}P\}-\sup_{\mathcal{B}(f_l,\varepsilon)}\{-\nu'-\sqrt{n}P\}\right|\leq 2\delta\right)\\
&\leq\Pr^*\left(\left|\sup_{\mathcal{B}(f_k,\varepsilon)}\{-\nu'-\sqrt{n}P\}-\sup_{\mathcal{B}(f_l,\varepsilon)}\{-\nu'-\sqrt{n}P\}\right|\leq 2\delta\right)+\Pr^*\left(\sup_{\mathcal{F}}\left|\nu'_n-\nu'\right|\geq\delta/2\right)\\
&\leq\frac{128\delta}{C^3}+\Pr^*\left(\sup_{\mathcal{F}}\left|\nu'_n-\nu'\right|\geq\delta/2\right),
\end{aligned}$$

where the first inequality results from a union bound argument while the second one results from Lemma 22 noticing that $-\nu'-\sqrt{n}P$ is a Gaussian process with covariance $\langle f,f'\rangle$ and mean $-\sqrt{n}P$, and since by construction $\varepsilon\leq C^3/128$.

Finally, the claimed result follows from the two last relations. ■

We now prove, Corollary 11, the extension of Theorem 10 to L_2 diameters. The proof relies on the observation that a P-Donsker class is also Glivenko-Cantelli.

Proof [Corollary 11] Note that

$$\left\|f-f'\right\|_{L_2}^2=\left\|f-f'\right\|^2+\left(P(f-f')\right)^2.$$

The expected errors of almost-minimizers over a Glivenko-Cantelli (and therefore over Donsker) class are close because empirical averages uniformly converge to the expectations.

$$\begin{aligned}
&\Pr^*\left(\exists f,f'\in\mathcal{M}_S^{\xi(n)}\quad\text{s.t. }\left\|f-f'\right\|_{L_2}>C\right)\\
&\leq\Pr^*\left(\exists f,f'\in\mathcal{M}_S^{\xi(n)}\quad\text{s.t. }\left|Pf-Pf'\right|>C/\sqrt{2}\right)+\Pr^*\left(\operatorname{diam}\mathcal{M}_S^{\xi(n)}>C/\sqrt{2}\right).
\end{aligned}$$

The first term can be bounded as

$$\begin{aligned}
&\Pr^*\left(\exists f,f'\in\mathcal{M}_S^{\xi(n)}\quad\text{s.t. }\left|Pf-Pf'\right|>C/\sqrt{2}\right)\\
&\leq\Pr^*\left(\exists f,f'\in\mathcal{F},\left|P_nf-P_nf'\right|\leq\xi(n),\left|Pf-Pf'\right|>C/\sqrt{2}\right)\\
&\leq\Pr^*\left(\sup_{f,f'\in\mathcal{F}}\left|(P_n-P)(f-f')\right|>\left|C/\sqrt{2}-\xi(n)\right|\right)
\end{aligned}$$

which goes to 0 because the class $\{f - f' | f, f' \in \mathcal{F}\}$ is Glivenko-Cantelli. The second term goes to 0 by Theorem 10. ■

Appendix B.

In this appendix we report the proof of Theorem 20 stated in Section 6. We first need to derive a preliminary lemma.

Lemma 23 *Let $\mathcal{F}$ be P-Donsker class with envelope function $G \equiv 1$. Assume $\mathcal{N}(\gamma, \mathcal{F}) = \sup_Q \mathcal{N}(\gamma, \mathcal{F}, L_1(Q)) < \infty$ for $0 < \gamma \leq 1$ and Q ranging over all discrete probability measures. Let $\mathcal{M}_S^{\xi(n)}$ be defined as above with $\xi(n) = o(n^{-1/2})$ and assume that for some sequence of positive numbers $\lambda(n) = o(n^{1/2})$*

$$\lambda(n) \sup_{f, f' \in \mathcal{M}_S^{\xi(n)}} P|f - f'| \xrightarrow{P^*} 0. \tag{1}$$

Suppose further that for some $1/2 < \rho < 1$

$$\lambda(n)^{2\rho-1} - \log \mathcal{N}(\frac{1}{2} n^{-1/2} \lambda(n)^{\rho-1}, \mathcal{F}) \to +\infty. \tag{2}$$

Then

$$\Pr^* \left(\sqrt{n} \sup_{f, f' \in \mathcal{M}_S^{\xi(n)}} |P(f - f')| \leq \sqrt{n}\xi(n) + 131\lambda(n)^{\rho-1} \right) \to 0.$$

Proof Define $\mathcal{G} = \{f - f' : f, f' \in \mathcal{F}\}$ and $\mathcal{G}' = \{|f - f'| : f, f' \in \mathcal{F}\}$. By Example 2.10.7 of van der Vaart and Wellner (1996), $\mathcal{G} = (\mathcal{F}) + (-\mathcal{F})$ and $\mathcal{G}' = |\mathcal{G}| \subseteq (\mathcal{G} \wedge 0) \vee (-\mathcal{G} \wedge 0)$ are Donsker as well. Moreover, $\mathcal{N}(2\gamma, \mathcal{G}) \leq \mathcal{N}(\gamma, \mathcal{F})^2$ and the envelope of $\mathcal{G}$ is $G \equiv 2$. Applying Proposition 19 to the class $\mathcal{G}$, we obtain

$$\Pr^* \left(\sup_{f, f' \in \mathcal{F}} \frac{|P_n(f - f') - P(f - f')|}{\varepsilon(P_n|f - f'| + P|f - f'|) + 5\gamma} > 26 \right) \leq 32\mathcal{N}(\gamma/2, \mathcal{F})^2 \exp(-n\varepsilon\gamma).$$

The inequality therefore holds if the sup is taken over a smaller (random) subclass $\mathcal{M}_S^{\xi(n)}$.

$$\Pr^* \left(\sup_{f, f' \in \mathcal{M}_S^{\xi(n)}} \frac{|P(f - f')| - \xi(n)}{\varepsilon(P_n|f - f'| + P|f - f'|) + 5\gamma} > 26 \right) \leq 32\mathcal{N}(\gamma/2, \mathcal{F})^2 \exp(-n\varepsilon\gamma).$$

Since $\sup_x \frac{A(x)}{B(x)} \geq \sup_x \frac{A(x)}{\sup_x B(x)} = \frac{\sup_x A(x)}{\sup_x B(x)}$,

$$\Pr^* \left(\sup_{f, f' \in \mathcal{M}_S^{\xi(n)}} \big(|P(f - f')| - \xi(n)\big) > 26 \sup_{f, f' \in \mathcal{M}_S^{\xi(n)}} \big(\varepsilon(P_n|f - f'| + P|f - f'|) + 5\gamma\big) \right) \leq 32\mathcal{N}(\gamma/2, \mathcal{F})^2 \exp(-n\varepsilon\gamma). \tag{3}$$

By assumption,

$$\lambda(n) \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} P|f-f'| \xrightarrow{P^*} 0.$$

Because $\mathcal{G}'$ is Donsker and $\lambda(n) = o(n^{1/2})$,

$$\lambda(n) \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} \left| P_n|f-f'| - P|f-f'| \right| \xrightarrow{P^*} 0.$$

Thus,

$$\lambda(n) \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} P_n|f-f'| + P|f-f'| \xrightarrow{P^*} 0.$$

Letting $\varepsilon = \varepsilon(n) := n^{-1/2}\lambda(n)^\rho$, this implies that for any $\delta > 0$, there exist N_δ such that for all $n > N_\delta$,

$$\Pr^* \left(\sqrt{n} \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} 26\varepsilon(n) \left(P_n|f-f'| + P|f-f'| \right) > \lambda(n)^{\rho-1} \right) < \delta.$$

Now, choose $\gamma = \gamma(n) := n^{-1/2}\lambda(n)^{\rho-1}$ (note that since $\rho < 1$, eventually $0 < \gamma(n) < 1$), the last inequality can be rewritten in the following form

$$\Pr^* \left(\sqrt{n} \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} 26 \left(\varepsilon(n) \left(P_n|f-f'| + P|f-f'| \right) + 5\gamma(n) \right) > 131\lambda(n)^{\rho-1} \right) < \delta.$$

Combining the relation above with Equation 3,

$$\begin{aligned} &\Pr^* \left(\sqrt{n} \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} |P(f-f')| \le \sqrt{n}\xi(n) + 131\lambda(n)^{\rho-1} \right) \\ &\ge 1 - 32\mathcal{N}\left(\frac{1}{2} n^{-1/2}\lambda(n)^{\rho-1}, \mathcal{F} \right)^2 \exp(-\lambda(n)^{2\rho-1}) - \delta. \end{aligned}$$

The result follows by the assumption on the entropy and by arbitrariness of δ.

■

We are now ready to prove Theorem 20.

Proof [Theorem 20] By Corollary 18,

$$n^\gamma \mathrm{diam} \mathcal{M}_S^{\xi(n)} \xrightarrow{P^*} 0$$

for any $\gamma < \min\left(\frac{1}{3(2V+1)} \min(\alpha, \eta), 1/2 \right)$. Let $\lambda(n) = n^\gamma$ and note that $\lambda(n) = o(\sqrt{n})$, which is a condition in Lemma 23. First, we show that a power decay of the $\|\cdot\|$ diameter implies the same rate

of decay of the L_1 diameter, hence verifying condition (1) in Lemma 23. Proof of this fact is very similar to the proof of Corollary 11, except that C is replaced by $C\lambda(n)^{-1}$.

$$\begin{aligned}
&\Pr^*\left(\exists f, f' \in \mathcal{M}_S^{\xi(n)} \quad \text{s.t. } \left\|f - f'\right\|_{L_2} > C\lambda(n)^{-1}\right) \\
&\leq \Pr^*\left(\exists f, f' \in \mathcal{M}_S^{\xi(n)} \quad \text{s.t. } \left|Pf - Pf'\right| > C\lambda(n)^{-1}/\sqrt{2}\right) \\
&+ \Pr^*\left(\operatorname{diam}\mathcal{M}_S^{\xi(n)} > C\lambda(n)^{-1}/\sqrt{2}\right).
\end{aligned}$$

The second term goes to zero since $\lambda(n)\operatorname{diam}\mathcal{M}_S^{\xi(n)} \xrightarrow{P^*} 0$. Moreover, since $\lambda(n) = o(\sqrt{n})$ and $\mathcal{G}$ is Donsker, the first term can be bounded as

$$\begin{aligned}
&\Pr^*\left(\exists f, f' \in \mathcal{M}_S^{\xi(n)} \quad \text{s.t. } \left|Pf - Pf'\right| > C\lambda(n)^{-1}/\sqrt{2}\right) \\
&\leq \Pr^*\left(\exists f, f' \in \mathcal{F}, \left|P_n f - P_n f'\right| \leq \xi(n), \left|Pf - Pf'\right| > C\lambda(n)^{-1}/\sqrt{2}\right) \\
&\leq \Pr^*\left(\sup_{f,f' \in \mathcal{F}} |P(f - f') - P_n(f - f')| > \left|\frac{C}{\sqrt{2}}\lambda(n)^{-1} - \xi(n)\right|\right) \\
&= \Pr^*\left(\lambda(n)\sup_{g \in \mathcal{G}} |Pg - P_n g| > \left|\frac{C}{\sqrt{2}} - \xi(n)\lambda(n)\right|\right) \to 0,
\end{aligned}$$

proving condition (1) in Lemma 23.

We now verify condition (2) in Lemma 23. Since $\mathcal{F}$ is a VC-subgraph class of dimension V, its entropy numbers $\log \mathcal{N}(\varepsilon, \mathcal{F})$ behave like $V \log \frac{A}{\varepsilon}$ (A is a constant), that is

$$\log \mathcal{N}\left(\frac{1}{2}n^{-1/2}\lambda(n)^{\rho-1}, \mathcal{F}\right) \leq const + \frac{1}{2}V \log n + (1-\rho)V \log \lambda(n).$$

Condition (2) of Lemma 23 will therefore hold whenever $\lambda(n)$ grows faster than $(\log n)^{\frac{1}{2\rho-1}}$, for any $1 > \rho > \frac{1}{2}$. In our problem, $\lambda(n)$ grows polynomially, so condition (2) is satisfied for any fixed $1 > \rho > 1/2$.

Hence, by Lemma 23

$$\Pr^*\left(\sqrt{n} \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} |P(f - f')| \leq \sqrt{n}\xi(n) + 131 n^{\gamma(\rho-1)}\right) \to 0.$$

Choose any $0 < \kappa < \gamma/2$ and multiply both sides of the inequality by n^κ. We obtain

$$\Pr^*\left(n^\kappa\sqrt{n} \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} |P(f - f')| \leq \sqrt{n}\xi(n)n^\kappa + 131 n^{\gamma(\rho-1)+\kappa}\right) \to 0.$$

Now fix a ρ such that $1/2 < \rho < 1 - \kappa/\gamma$. Because $0 < \kappa < \gamma/2$, there is always such a choice of ρ. Furthermore, $1 > \rho > 1/2$ so that the above convergence holds. Our choice of ρ implies that $\gamma(\rho - 1) + \kappa < 0$ and so $n^{\gamma(\rho-1)+\kappa} \to 0$. Since $\kappa < \gamma/2 < \eta$, $\sqrt{n}\xi(n)n^\kappa \to 0$. Hence,

$$n^{1/2+\kappa} \sup_{f,f' \in \mathcal{M}_S^{\xi(n)}} |P(f - f')| \xrightarrow{P^*} 0$$

for any $\kappa < \min\left(\frac{1}{6(2V+1)}\min(\alpha,\eta), 1/2\right)$. ■

References

P. L. Bartlett and S. Mendelson. Empirical minimization. *Probability Theory and Related Fields*, 135(3):311–334, 2006.

P. L. Bartlett, S. Mendelson, and P. Philips. Local complexities for empirical risk minimization. In J. Shawe-Taylor and Y. Singer, editors, *Proceedings of the 17th Annual Conference on Learning Theory*, pages 270–284. Springer, 2004.

S. Ben-David, N. Eiron, and P. M. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of mathematics. Springer, New York, 1996.

L. P. Devroye and T. J. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

R. M. Dudley. *Uniform Central Limit Theorems*. Cambridge University Press, 1999.

R. M. Dudley. *Real Analysis and Probability*. Cambrige Studies in Advaced Mathematics. Cambrigde University Press, 2002.

E. Giné and J. Zinn. Gaussian characterization of uniform Donsker classes of functions. *The Annals of Probability*, 19:758–782, 1991.

J. Kim and D. Pollard. Cube root asymptotics. *Annals of Statistics*, 18:191–219, 1990.

V. Koltchinskii. Local Rademacher complexities and oracle inequalities in risk minimization. *Annals of Statistics*, 2006. To appear.

V. Koltchinskii. Komlós-Major-Tusnády approximation for the general empirical process and Haar expansion of classes of functions. *Journal of Theoretical Probability*, 7:73–118, 1994.

S. Kutin and P. Niyogi. Almost-everywhere algorithmic stability and generalization error. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 275–28, San Francisco, CA, 2002. Morgan Kaufmann.

W. S. Lee, P. L. Bartlett, and R. C. Williamson. The importance of convexity in learning with squared loss. *IEEE Transactions on Information Theory*, 44(5):1974–1980, 1998.

S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Statistical learning: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25:161–193, 2006.

T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, 2004.

D. Pollard. Uniform ratio limit theorems for empirical processes. *Scandinavian Journal of Statistics*, 22(3):271–278, 1995.

A. Rakhlin and A. Caponnetto. Stability of *k*-means clustering. In *Proceedings of Neural Information Processing Systems Conference*, 2006. To appear.

A. Rakhlin, S. Mukherjee, and T. Poggio. Stability results in learning theory. *Analysis and Applications*, 3(4):397–417, 2005.

E. Rio. Strong approximation for set-indexed partial sum processes via KMT constructions I. *The Annals of Probability*, 21(2):759–790, 1993.

M. Rudelson and R. Vershynin. Combinatorics of random processes and sections of convex bodies. *Annals of Mathematics*. To appear.

S. van de Geer. *Empirical Processes in M-Estimation*. Cambridge University Press, 2000.

A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes with Applications to Statistics*. Springer-Verlag, New York, 1996.

V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequences of events to their probabilities. *Th. Prob. and its Applications*, 17(2):264–280, 1971.

V .N. Vapnik and A. Ya. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

Journal of Machine Learning Research 7 (2006) 2585-2602 Submitted 3/06; Revised 8/06; Published 12/06

Linear State-Space Models for Blind Source Separation

Rasmus Kongsgaard Olsson RKO@IMM.DTU.DK
Lars Kai Hansen LKH@IMM.DTU.DK
Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Lyngby, Denmark

Editor: Aapo Hyvärinen

Abstract

We apply a type of generative modelling to the problem of blind source separation in which prior knowledge about the latent source signals, such as time-varying auto-correlation and quasi-periodicity, are incorporated into a linear state-space model. In simulations, we show that in terms of signal-to-error ratio, the sources are inferred more accurately as a result of the inclusion of strong prior knowledge. We explore different schemes of maximum-likelihood optimization for the purpose of learning the model parameters. The Expectation Maximization algorithm, which is often considered the standard optimization method in this context, results in slow convergence when the noise variance is small. In such scenarios, quasi-Newton optimization yields substantial improvements in a range of signal to noise ratios. We analyze the performance of the methods on convolutive mixtures of speech signals.

Keywords: blind source separation, state-space model, independent component analysis, convolutive model, EM, speech modelling

1. Introduction

We are interested in blind source separation (BSS) in which unknown source signals are estimated from noisy mixtures. Real world application of BSS techniques are found in as diverse fields as audio (Yellin and Weinstein, 1996; Parra and Spence, 2000; Anemüller and Kollmeier, 2000), brain imaging and analysis (McKeown et al., 2003), and astrophysics (Cardoso et al., 2002). While most prior work is focused on mixtures that can be characterized as instantaneous, we will here investigate causal convolutive mixtures. The mathematical definitions of these classes of mixtures are given later in this introductory section. Convolutive BSS is relevant in many signal processing applications, where the instantaneous mixture model cannot possibly capture the latent causes of the observations due to different time delays between the sources and sensors. The main problem is the lack of general models and estimation schemes; most current work is highly application specific with the majority focused on applications in separation of speech signals. In this work we will also be concerned with speech signals, however, we will formulate a generative model that may be generalizable to several other application domains.

One of the most successful approaches to convolutive BSS is based on the following assumptions: 1) The mixing process is linear and causal, 2) the source signals are statistically independent, 3) the sources can be fully characterized by their *time variant* second order statistics (Weinstein et al., 1993; Parra and Spence, 2000). The last assumption is defining for this approach. Keeping to second order statistics we simplify computations but have to pay the price of working with time-

variant statistics. It is well-known that stationary second order statistics, that is, covariances and correlation functions, are not informative enough in the convolutive mixing case.

Our research concerns statistical analysis and generalizations of this approach. We formulate a generative model based on the same statistics as the Parra-Spence model. The benefit of this generative approach is that it allows for estimation of additional noise parameters and injection of well-defined a priori information in a Bayesian sense (Olsson and Hansen, 2005). Furthermore, we propose several algorithms to learn the parameters of the proposed models.

The linear mixing model reads

$$\mathbf{x}_t = \sum_{k=0}^{L-1} \mathbf{A}_k \mathbf{s}_{t-k} + \mathbf{w}_t. \tag{1}$$

At discrete time t, the observation vector, $\mathbf{x}_t$, results from the convolution sum of the L time-lagged mixing matrices $\mathbf{A}_k$ and the source vector $\mathbf{s}_t$. The individual sources, that is, the elements of $\mathbf{s}_t$, are assumed to be statistically independent. The observations are corrupted by additive i.i.d. Gaussian noise, $\mathbf{w}_t$. BSS is concerned with estimating $\mathbf{s}_t$ from $\mathbf{x}_t$, while $\mathbf{A}_k$ is unknown. It is apparent from (1) that only filtered versions of the elements of $\mathbf{s}_t$ can be retrieved, since the inverse filtering can be applied to the unknown $\mathbf{A}_k$. As a special case of the filtering ambiguity, the *scale* and the ordering of the sources is unidentifiable. The latter is evident from the fact that various permutation applied simultaneously to the elements of $\mathbf{s}_t$ and the columns of $\mathbf{A}_t$ produce identical mixtures, $\mathbf{x}_t$.

Equation (1) collapses to an *instantaneous* mixture in the case of $L = 1$ for which a variety of Independent Component Analysis (ICA) methods are available (e.g., Comon, 1994; Bell and Sejnowski, 1995; Hyvarinen et al., 2001). As already mentioned, however, we will treat the class of convolutive mixtures, that is $L > 1$.

Convolutive Independent Component Analysis (C-ICA) is a class of BSS methods for (1) where the source estimates are produced by computing the 'unmixing' transformation that restores statistical independence. Often, an inverse linear filter (e.g., FIR) is applied to the observed mixtures. Simplistically, the separation filter is estimated by minimizing the mutual information, or 'cross' moments, of the 'separated' signals. In many cases non-Gaussian models/higher-order statistics are required, which require a relatively long data series for reliable estimation. This can be executed in the time domain (Lee et al., 1997; Dyrholm and Hansen, 2004), or in the frequency domain (e.g., Parra and Spence, 2000). The transformation to the Fourier domain reduces the matrix convolution of (1) to a matrix product. In effect, the more difficult convolutive problem is decomposed into a number of manageable instantaneous ICA problems that can be solved independently using the mentioned methods. However, frequency domain decomposition suffers from *permutation over frequency* which is a consequence of the potential different orderings of sources at different frequencies. Many authors have explored solutions to the permutation-over-frequency problem that are based on measures of spectral structure (e.g., Anemüller and Kollmeier, 2000), where amplitude correlation across frequency bands is assumed and incorporated in the algorithm.

The work presented here forges research lines that treat instantaneous ICA as a density estimation problem (Pearlmutter and Parra, 1997; Højen-Sørensen et al., 2002), with richer source priors that incorporate time-correlation, non-stationarity, periodicity and the convolutive mixture model to arrive at an C-ICA algorithm. The presented algorithm, which operates entirely in the time-domain, relies on a linear state-space model, for which estimation and exact source inference are available. The states directly represent the sources, and the transition structure can be interpreted as describing the internal time-correlation of the sources. To further increase the audio realism of the model,

Olsson and Hansen (2005) added a harmonic excitation component in the source speech model (Brandstein, 1998); this idea is further elaborated and tested here.

Algorithms for the optimization of the likelihood of the linear state-space model are devised and compared, among them the basic EM algorithm, which is used extensively in latent variable models (Moulines et al., 1997). In line with Bermond and Cardoso (1999), the EM-algorithm is shown to exhibit slow convergence in good signal to noise ratios.

It is interesting that the two 'unconventional' aspects of our generative model: the non-stationarity of the source signals and their harmonic excitation, do not change the basic quality of the state-space model, namely that exact inference of the sources and exact calculation of the log-likelihood and its gradient are still possible.

The paper is organized as follows: First we introduce the state-space representation of the convolutive mixing problem and the source models in Section 2, in Section 3 we briefly recapitulate the steps towards exact inference for the source signals, while Section 4 is devoted to a discussion of parameter learning. Sections 5 and 6 present a number of experimental illustrations of the approach on simulated and speech data respectively.

2. Model

The convolutive blind source separation problem is cast as a density estimation task in a latent variable model as was suggested in Pearlmutter and Parra (1997) for the instantaneous ICA problem

$$\mathrm{p}(\mathbf{X}|\theta) = \int \mathrm{p}(\mathbf{X}|\mathbf{S},\theta_1)\mathrm{p}(\mathbf{S}|\theta_2)d\mathbf{S}.$$

Here, the matrices $\mathbf{X}$ and $\mathbf{S}$ are constructed as the column sets of $\mathbf{x}_t$ and $\mathbf{s}_t$ for all t. The functional forms of the conditional likelihood, $\mathrm{p}(\mathbf{X}|\mathbf{S},\theta_1)$, and the joint source prior, $\mathrm{p}(\mathbf{S}|\theta_2)$, should ideally be selected to fit the realities of the separation task at hand. The distributions depend on a set of tunable parameters, $\theta \equiv \{\theta_1,\theta_2\}$, which in a blind separation setup is to be learned from the data. In the present work, $\mathrm{p}(\mathbf{X}|\mathbf{S},\theta_1)$ and $\mathrm{p}(\mathbf{S}|\theta_2)$ have been restricted to fit into a class of linear state-space models, for which effective estimation schemes exist (Roweis and Ghahramani, 1999)

$$\mathbf{s}_t = \mathbf{F}^n\mathbf{s}_{t-1} + \mathbf{C}^n\mathbf{u}_t + \mathbf{v}_t, \quad (2)$$

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t + \mathbf{w}_t. \quad (3)$$

Equations (2) and (3) describe the *state/source* and *observation* spaces, respectively. The parameters of the former are time-varying, indexed by the block index n, while the latter noisy mixing process is stationary. The randomness of the model is enabled by i.i.d. zero mean Gaussian variables, $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0},\mathbf{Q}^n)$, and $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0},\mathbf{R})$ The 'input' or 'control' signal $\mathbf{u}_t \equiv \mathbf{u}_t(\psi^n)$ deterministically shifts the mean of $\mathbf{s}_t$ depending on parameters ψ^n. Various structures can be imposed on the model parameters, $\theta_1 = \{\mathbf{A},\mathbf{R}\}$ and $\theta_2 = \{\mathbf{F}^n,\mathbf{C}^n,\mathbf{Q}^n,\psi^n\}$, in order to create the desired effects. For equations (2) and (3) to pose as a generative model for the instantaneous mixture of first-order autoregressive, AR(1), sources it need only be assumed that $\mathbf{F}^n$ and $\mathbf{Q}^n$ are diagonal matrices and that $\mathbf{C}^n = \mathbf{0}$. In this case, $\mathbf{A}$ functions as the mixing matrix. In Section 2.1, we generalize to AR(p) and convolutive mixing.

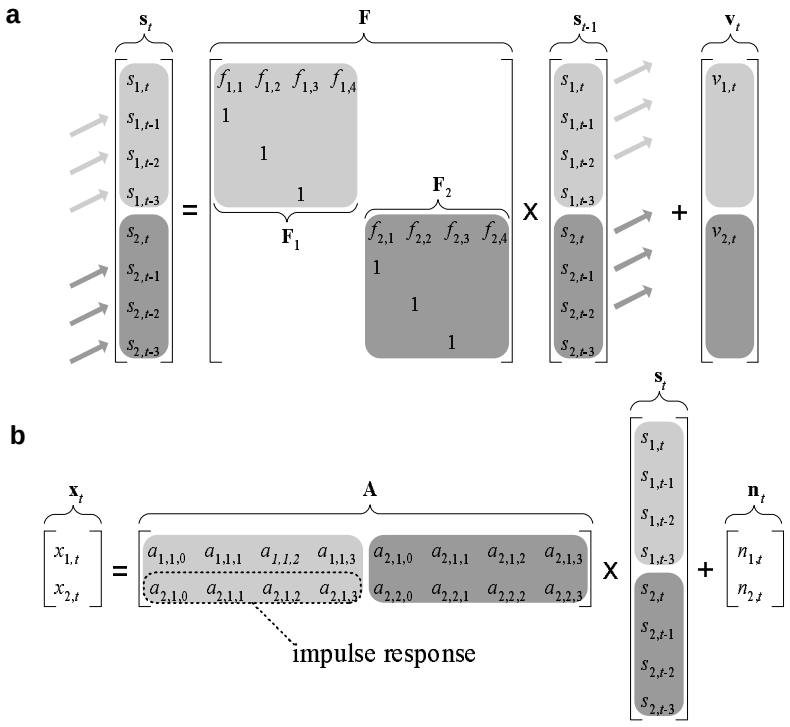


Figure 1: The dynamics of the linear state space model when it has been constrained to describe a noisy convolutive mixture of $P = 2$ autoregressive (AR) sources. This is achieved by augmenting the source vector to contain time-lagged signals. In **a** is shown the corresponding source update, when the order of the AR process is $p = 4$. In **b**, the sources are mixed through filters ($L = 4$) into $Q = 2$ noisy mixtures. Blanks signify zeros.

2.1 Auto-Regressive Source Prior

The AR(p) source prior for source i in frame n is defined as follows,

$$s_{i,t} = \sum_{k=1}^{p} f_{i,k}^{n} s_{i,t-k} + v_{i,t}$$

where $t \in \{1,2,..,T\}$, $n \in \{1,2,..,N\}$ and $i \in \{1,2,..,P\}$. The excitation noise is i.i.d. zero mean Gaussian: $v_{i,t} \sim \mathcal{N}(0, q_i^n)$. It is an important point that the convolutive mixture of AR(p) sources can be contained in the linear state-space model (2) and (3), this is illustrated in Figure 1. The enabling trick, which is standard in time series analysis, is to augment the source vector to include a time history so that it contains L time-lagged samples of all P sources

$$\mathbf{s}_t = \left[\begin{array}{cccc} (\mathbf{s}_{1,t})^\top & (\mathbf{s}_{2,t})^\top & \ldots & (\mathbf{s}_{P,t})^\top \end{array} \right]^\top$$

where the i'th source is represented as

$$\mathbf{s}_{i,t} = \left[\begin{array}{cccc} s_{i,t} & s_{i,t-1} & \ldots & s_{i,t-L+1} \end{array} \right]^\top .$$

Furthermore, constraints are enforced on the matrices of θ

$$\mathbf{F}^n = \begin{bmatrix} \mathbf{F}_1^n & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_2^n & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_P^n \end{bmatrix}, \qquad \mathbf{Q}^n = \begin{bmatrix} \mathbf{Q}_1^n & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2^n & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q}_P^n \end{bmatrix},$$

$$\mathbf{F}_i^n = \begin{bmatrix} f_{i,1}^n & f_{i,2}^n & \cdots & f_{i,p-1}^n & f_{i,p}^n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \qquad (\mathbf{Q}_i^n)_{jj'} = \begin{cases} (q_i^2)^n & j = j' = 1 \\ 0 & j \neq 1 \bigvee j' \neq 1 \end{cases},$$

$$\mathbf{C}^n = \mathbf{0},$$

where $\mathbf{F}_i^n$ was defined for $p = L$. In the interest of the simplicity of the presentation, it is assumed that $\mathbf{F}_i^n$ has L row and columns. We furthermore assume that $p \leq L$; in the case of $p < L$, zeros replace the affected (rightmost) coefficients. Hence, the dimensionality of $\mathbf{A}$ is $Q \times (p \times P)$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{11}^\top & \mathbf{a}_{12}^\top & .. & \mathbf{a}_{1P}^\top \\ \mathbf{a}_{21}^\top & \mathbf{a}_{22}^\top & .. & \mathbf{a}_{2P}^\top \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_{Q1}^\top & \mathbf{a}_{Q2}^\top & .. & \mathbf{a}_{QP}^\top \end{bmatrix}$$

where $\mathbf{a}_{ij} = [a_{ij,1}, a_{ij,2}, .., a_{ij,L}]^\top$ can be interpreted as the impulse response of the channel filter between source i and sensor j. Overall, the model can described can be described as the generative, time-domain equivalent of Parra and Spence (2000).

2.2 Harmonic Source Prior

Many classes of audio signals, such as voiced speech and musical instruments, are approximately piece-wise periodic. By the Fourier theorem, such sequences can be represented well by a harmonic series. In order to account for colored noise residuals and noisy signals in general, a harmonic and noise (HN) model is suggested (McAulay and Quateri, 1986). The below formulation is used

$$s_{i,t} = \sum_{t'=1}^{p} f_{i,t'}^n s_{i,t-t'} + \sum_{k=1}^{K} \left[c_{i,2k-1}^n \sin(\omega_{0,i}^n t) + c_{i,2k}^n \cos(\omega_{0,i}^n t) \right] + v_{i,t}$$

where $\omega_{0,i}^n$' is the fundamental frequency of source i in frame n and the Fourier coefficients are contained in $c_{i,2k-1}^n$ and $c_{i,2k}^n$. The harmonic model is represented in the state space model (2) & (3)

through the definitions

$$\begin{aligned}
\mathbf{C}^n &= \begin{bmatrix} (\mathbf{c}_1^n)^\top & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & (\mathbf{c}_2^n)^\top & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & (\mathbf{c}_P^n)^\top \end{bmatrix}, \\
\mathbf{c}_i^n &= \begin{bmatrix} c_{i,1}^n & c_{i,2}^n & \cdots & c_{i,2K}^n \end{bmatrix}^\top, \\
\mathbf{u}_t^n &= \begin{bmatrix} (\mathbf{u}_{1,t}^n)^\top & (\mathbf{u}_{2,t}^n)^\top & \ldots & (\mathbf{u}_{P,t}^n)^\top \end{bmatrix}^\top,
\end{aligned}$$

where the k'th harmonics of source i in frame n are defined as $(\mathbf{u}_{i,t}^n)_{2k-1} = \sin(k\omega_{0,i}^n t)$ and $(\mathbf{u}_{i,t}^n)_{2k} = \cos(k\omega_{0,i}^n t)$, implying the following parameter set for the source mean: $\psi^n = \begin{bmatrix} \omega_{0,1}^n & \omega_{0,2}^n & \ldots & \omega_{0,P}^n \end{bmatrix}$. Other parametric mean functions could, of course, be used, for example, a more advanced speech model.

3. Source Inference

In a maximum a posteriori sense, the sources, $\mathbf{s}_t$, can be optimally reconstructed using the Kalman filter/smoother (Kalman and Bucy, 1960; Rauch et al., 1965). This is based on the assumption that the parameters θ are known, either a priori or have been estimated as described in Section 4. While the filter computes the time-marginal moments of the source posterior conditioned on past and present samples, that is, $\langle \mathbf{s}_t \rangle_{p(\mathbf{S}|\mathbf{x}_{1:t},\theta)}$ and $\langle \mathbf{s}_t \mathbf{s}_t^\top \rangle_{p(\mathbf{S}|\mathbf{x}_{1:t},\theta)}$, the smoother conditions on samples from the entire block: $\langle \mathbf{s}_t \rangle_{p(\mathbf{S}|\mathbf{x}_{1:T},\theta)}$ and $\langle \mathbf{s}_t \mathbf{s}_t^\top \rangle_{p(\mathbf{S}|\mathbf{x}_{1:T},\theta)}$. For the Kalman filter/smoother to compute MAP estimates, it is a precondition due that the model is linear and Gaussian. The computational complexity is $O(TL^3)$ due to a matrix inversion occurring in the recursive update. Note that the forward recursion also yields the exact log-likelihood of the parameters given the observations, $\mathcal{L}(\theta)$. A thorough review of linear state-space modelling, estimation and inference from a machine learning point of view can be found in Roweis and Ghahramani (1999).

4. Learning

The task of learning the parameters of the state-space model from data is approached by maximum-likelihood estimation, that is, the log-likelihood function, $\mathcal{L}(\theta)$, is optimized with respect to the parameters, θ. The log-likelihood is defined as a marginalization over the hidden sources

$$\mathcal{L}(\theta) = \log \mathrm{p}(\mathbf{X}|\theta) = \log \int \mathrm{p}(\mathbf{X}, \mathbf{S}|\theta) d\mathbf{S}.$$

A closed-form solution, $\theta = \arg\max_{\theta'} \mathcal{L}(\theta')$, is not available, hence iterative algorithms that optimize $\mathcal{L}(\theta)$ are employed. In the following sections three such algorithms are presented.

4.1 Expectation Maximization Algorithm

Expectation Maximization (EM) (Dempster et al., 1977), has been applied to latent variable models in, for example, Shumway and Stoffer (1982) and Roweis and Ghahramani (1999). In essence, EM

is iterative optimization of a lower bound decomposition of the log-likelihood

$$\mathcal{L}(\theta) \quad \geq \quad \mathcal{F}(\theta, \hat{p}) = \mathcal{J}(\theta, \hat{p}) - \mathcal{R}(\hat{p}) \tag{4}$$

where $\hat{p}(\mathbf{S})$ is any normalized distribution and the following definitions apply

$$\begin{aligned} \mathcal{J}(\theta, \hat{p}) &= \int \hat{p}(\mathbf{S}) \log p(\mathbf{X}, \mathbf{S}|\theta) d\mathbf{S}, \\ \mathcal{R}(\hat{p}) &= \int \hat{p}(\mathbf{S}) \log \hat{p}(\mathbf{S}) d\mathbf{S}. \end{aligned}$$

Jensen's inequality leads directly to (4). The algorithm alternates between performing Expectation (E) and Maximization (M) steps, guaranteeing that $\mathcal{L}(\theta)$ does not decrease following an update. On the E-step, the Kalman smoother is used to compute the marginal moments from the source posterior, $\hat{p} = p(\mathbf{S}|\mathbf{X}, \theta)$, see Section 3. The M-step amounts to optimization of $\mathcal{J}(\theta, \hat{p})$ with respect to θ (since this is the only $\mathcal{F}(\theta, \hat{p})$ term which depends on θ. Due to the choice of a linear Gaussian model, closed-form estimators are available for the M-step (see appendix A for derivations).

In order to improve on the convergence speed of the basic EM algorithm, the search vector devised by the M-step update is premultiplied by an adaptive step-size η. A simple exponentially increase of η from 1 was used until a decrease in $\mathcal{L}(\theta)$ was observed at which point η was reset to 1. This speed-up scheme was applied successfully in Salakhutdinov and Roweis (2003). Below follow the M-step estimators for the AR and HN models. All expectations $\langle \cdot \rangle$ are over the source posterior, $p(\mathbf{S}|\mathbf{X}, \theta)$:

4.1.1 AUTOREGRESSIVE MODEL

For source i in block n:

$$\begin{aligned} \mathbf{f}_{i,\mathtt{new}}^n &= \Big[\sum_{t=2+t_0(n)}^{T+t_0(n)} \langle \mathbf{s}_{i,t-1} \mathbf{s}_{i,t-1}^\top \rangle \Big]^{-\top} \Big[\sum_{t=2+t_0(n)}^{T+t_0(n)} \langle s_{i,t} \mathbf{s}_{i,t-1} \rangle \Big], \\ q_{i,\mathtt{new}}^n &= \frac{1}{T-1} \sum_{t=2+t_0(n)}^{T+t_0(n)} \Big[\langle s_{i,t}^2 \rangle - \big(\mathbf{f}_{i,\mathtt{new}}^n \big)^\top \langle s_{i,t} \mathbf{s}_{i,t-1} \rangle \Big], \end{aligned}$$

where $t_0(n) = (n-1)T$. Furthermore:

$$\begin{aligned} \mathbf{A}_{\mathtt{new}} &= \Big[\sum_{t=1}^{NT} \mathbf{x}_t \langle \mathbf{s}_t \rangle^\top \Big] \Big[\sum_{t=1}^{NT} \langle \mathbf{s}_t (\mathbf{s}_t)^\top \rangle \Big]^{-1}, \\ \mathbf{R}_{\mathtt{new}} &= \frac{1}{NT} \sum_{t=1}^{NT} \mathrm{diag}[\mathbf{x}_t (\mathbf{x}_t)^\top - \mathbf{A}_{\mathtt{new}} \langle \mathbf{s}_t \rangle (\mathbf{x}_t)^\top], \end{aligned}$$

where the diag$[\cdot]$ operator extracts the diagonal elements of the matrix. Following an M-step, the solution corresponding to $||\mathbf{A}_i|| = 1 \ \forall i$ is chosen, where $||\cdot||$ is the Frobenius norm and $\mathbf{A}_i = \begin{bmatrix} \mathbf{a}_{i1} & \mathbf{a}_{i2} & \cdots & \mathbf{a}_{iQ} \end{bmatrix}^\top$, meaning that $\mathbf{A}$ and $\mathbf{Q}^n$ are scaled accordingly.

4.1.2 HARMONIC AND NOISE MODEL

The linear source parameters and signals are grouped as

$$\mathbf{d}_i^n \equiv \begin{bmatrix} (\mathbf{f}_i^n)^\top & (\mathbf{c}_i^n)^\top \end{bmatrix}^\top \quad , \quad \mathbf{z}_i \equiv \begin{bmatrix} (\mathbf{s}_{i,t-1})^\top & (\mathbf{u}_{i,t})^\top \end{bmatrix}^\top,$$

where

$$\mathbf{f}_i^n \equiv \left[\begin{array}{cccc} f_{i,1}^n & f_{i,2}^n & \cdots & f_{i,p}^n \end{array} \right]^\top \quad , \quad \mathbf{c}_i^n \equiv \left[\begin{array}{cccc} c_{i,1} & c_{i,2}^n & \cdots & c_{i,p}^n \end{array} \right]^\top .$$

It is in general not trivial to maximize $\mathcal{J}(\theta, \hat{p})$ with respect to $\omega_{i,0}^n$, since several local maxima exist, for example, at multiples of $\omega_{i,0}^n$ (McAulay and Quateri, 1986). However, simple grid search in a region provided satisfactory results. For each point in the grid we optimize $\mathcal{J}(\theta, \hat{p})$ with respect to $\mathbf{d}_i^n$:

$$\mathbf{d}_{i,\mathbf{new}}^n = \left[\sum_{t=2}^{NT} \left\langle \mathbf{z}_{i,t}(\mathbf{z}_{i,t})^\top \right\rangle \right]^{-1} \sum_{t=2}^{NT} \left\langle \mathbf{z}_{i,t}(s_{i,t})^\top \right\rangle .$$

The estimators of $\mathbf{A}$, $\mathbf{R}$ and q_i^n are similar to those in the AR model.

4.2 Gradient-based Learning

The derivative of the log-likelihood, $\frac{d\mathcal{L}(\theta)}{d\theta}$, can be computed and used in a quasi-Newton (QN) optimizer as is demonstrated in Olsson et al. (2006). The computation reuse the analysis of the M-step. This can be realized by rewriting $\mathcal{L}(\theta)$ as in Salakhutdinov et al. (2003):

$$\frac{d\mathcal{L}(\theta)}{d\theta} = \int p(\mathbf{S}|\mathbf{X}, \theta) \frac{d \log p(\mathbf{X}, \mathbf{S}|\theta)}{d\theta} d\mathbf{S} = \frac{d\mathcal{J}(\theta, \hat{p})}{d\theta}. \tag{5}$$

Due to the definition of $\mathcal{J}(\theta, \hat{p})$, the desired gradient in (5) can be computed following an E-step at relatively little effort. Furthermore, the analytic expressions are available from the derivation of the EM algorithm, see appendix A for details. A minor reformulation of the problem is necessary in order to maintain non-negative variances. Hence, the reformulations $\Omega^2 = \mathbf{R}$ and $(\phi_i^n)^2 = q_i^n$ are introduced. Updates are devised for Ω and ϕ_i^n. The derivatives are

$$\begin{aligned}
\frac{d\mathcal{L}(\theta)}{d\mathbf{A}} &= -\mathbf{R}^{-1}\mathbf{A} \sum_{t=1}^{NT} \left\langle \mathbf{s}_t \mathbf{s}_t^\top \right\rangle + \mathbf{R}^{-1} \sum_{t=1}^{N} \mathbf{x}_t \left\langle \mathbf{s}_t^\top \right\rangle , \\
\frac{d\mathcal{L}(\theta)}{d\Omega} &= \Omega^{-3} \sum_{t=1}^{NT} \left[\mathbf{x}_t \mathbf{x}_t^\top + \mathbf{A} \left\langle \mathbf{s}_t \mathbf{s}_t^\top \right\rangle \mathbf{A}^\top - 2\mathbf{x}_t \left\langle \mathbf{s}_t^\top \right\rangle \mathbf{A}^\top \right] , \\
\frac{d\mathcal{L}(\theta)}{d\mathbf{f}_i^n} &= \sum_{t=2+t_0(n)}^{T+t_0(n)} \left[\langle s_{i,t} \mathbf{s}_{i,t-1} \rangle - \left\langle \mathbf{s}_{i,t-1} \mathbf{s}_{i,t-1}^\top \right\rangle \mathbf{f}_i^n / q_i^n \right] , \\
\frac{d\mathcal{L}(\theta)}{d\phi_i^n} &= (1-T)/\phi_i^n + \\
&\quad \phi_i^{-3} \sum_{t=2+t_0(n)}^{T-1+t_0(n)} \left[\left\langle s_{i,t} s_{i,t}^\top \right\rangle + (\mathbf{f}_i^n)^\top \left\langle \mathbf{s}_{i,t-1} \mathbf{s}_{i,t-1}^\top \right\rangle \mathbf{f}_i^n - 2(\mathbf{f}_i^n)^\top \left\langle s_{i,t} \mathbf{s}_{i,t-1}^\top \right\rangle \right] .
\end{aligned}$$

In order to enforce the unit L2 norm on $\mathbf{A}_i$, a Lagrange multiplier is added to the derivative of $\mathbf{A}$. In this work, the QN optimizer of choice is the BFGS optimizer of Nielsen (2000).

4.3 Stochastic Gradient Learning

Although quasi-Newton algorithms often converge rapidly with a high accuracy, they do not scale well with the number of blocks, N. This is due to the fact that the number of parameters is asymptotically proportional to N, and therefore the internal inverse Hessian approximation becomes increasingly inaccurate. In order to be able to efficiently learn θ_2 ($\mathbf{A}$ and $\mathbf{R}$) for large N, a stochastic gradient approach (SGA), (Robbins and Monro, 1951), is employed.

It is adapted here to estimation in block-based state-space models, considering only a single randomly and uniformly sampled block, n, at any given time. The likelihood term corresponding to block n is $\mathcal{L}(\theta_1^n, \theta_2)$, where $\theta_1^n = \{\mathbf{F}^n, \mathbf{C}^n, \mathbf{Q}^n, \psi^n\}$. The stochastic gradient update to be applied is computed at the current optimum with respect to θ_1^n,

$$\begin{aligned} \Delta\theta_2 &= \eta \frac{d\mathcal{L}(\hat{\theta}_1^n, \theta_2)}{d\theta_2}, \\ \hat{\theta}_1^n &= \arg\max_{\theta_1^n} \mathcal{L}(\theta_1^n, \theta_2). \end{aligned}$$

where $\hat{\theta}_1^n$ is estimated using the EM algorithm. Employing an appropriate 'cooling' of the learning rate, η, is mandatory in order to ensure convergence: one such, devised by Robbins and Monro (1951), is choosing η proportional to $\frac{1}{k}$ where k is the iteration number. In our simulations, the SGA seemed more robust to the initial parameter values than the QN and the EM algorithms.

5. Learning from Synthetic Data

In order to investigate the convergence of the algorithms, AR(2) processes with time-varying pole placement were generated and mixed through randomly generated filters. For each signal frame, $T = 200$, the poles of the AR processes were constructed so that the amplification, r, was fixed while the center frequency was drawn uniformly from $\mathcal{U}(\pi/10, 9\pi/10)$. The filter length was $L = 8$ and the coefficients of the mixing filters, that is, the $\mathbf{a}_{ij}$ of $\mathbf{A}$, were generated from i.i.d. Gaussians weighted by an exponentially decaying function. Quadratic mixtures with $Q = P = 2$ were used: the first 2 elements of $\mathbf{a}_{12}$ and $\mathbf{a}_{21}$ were set to zero to simulate a situation with different channel delays. All channel filters were normalized to $||\mathbf{a}_{ij}||_2 = 1$. Gaussian i.i.d. noise was added in each channel, constructing the desired signal to noise ratio.

For evaluation purposes, the signal-to-error ratio (SER) was computed for the inferred sources. The true and estimated sources were mapped to the output by filtering through the direct channel so that the true source at the output is $\tilde{s}_{i,t} = \mathbf{a}_{ii} * s_{i,t}$. Similarly defined, the estimated source at the sensor is $\hat{s}_{i,t}$. Permutation ambiguities were resolved prior to evaluating the SER,

$$\mathrm{SER}_i = \frac{\sum_t \tilde{s}_{i,t}^2}{\sum_t (\tilde{s}_{i,t} - \hat{s}_{i,t})^2}.$$

The EM and QN optimizers were applied to learn the parameters from $N = 10$ frames of samples with SNR $= 20$dB, $r = 0.8$. The algorithms were restarted 5 times with random initializations, $\mathbf{A}_{ij} \in \mathcal{N}(0,1)$, the one that yielded the maximal likelihood was selected. Figure 2 shows the results of the EM run: the close match between the true and learned models confirms that the parameters can indeed be learned from the data using maximum-likelihood optimization. In Table 1, the generative approach is contrasted with a stationary finite impulse response (FIR) filter separator that

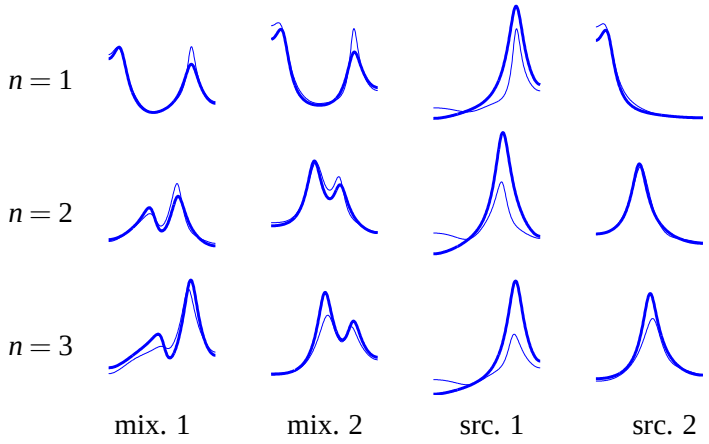


Figure 2: The true (bold) and estimated models for the first 3 frames of the synthetic data based on the autoregressive model. The amplitude frequency responses of the combined source and channel filters are shown: for source i, this amounts to the frequency response of the filter, with the scaling and poles of $\theta_{1,i}$ and zeros of the direct channel $\mathbf{a}_{ii}$. For the mixtures, the responses across channels were summed. The EM algorithm provided the estimates.

	Estimated	Generative	MSE FIR
AR	9.1 ± 0.4	9.7 ± 0.4	7.5 ± 0.2
HN	11.8 ± 0.7	13.2 ± 0.4	7.9 ± 0.5

Table 1: The signal-to-error ratio (SER) performance on synthetic data based on the autoregressive (AR) and harmonic-and-noise (HN) source models. Mean and standard deviation of the mean are shown for 1) the EM algorithm applied to the mixtures, 2) inferences from data and the true model, and, 3) the optimal FIR filter separator. The mean SER and the standard deviation of the mean were calculated from $N = 10$ signal frames, SNR $= 20$dB.

in a supervised fashion was optimized to minimize the squared error between the estimated and true sources, $L_{\text{FIR}} = 25$. Depending on the signal properties, the generative approach, which results in a time-varying filter, results in a clear advantage over the time-invariant FIR filter, which has to compromise across the signal's changing dynamics. As a result, the desired signals are only sub-optimally inferred by methods that apply a constant filter to the mixtures. The performance of the learned model is upper-bounded by that of the generative model, since the maximum likelihood estimator is only unbiased in the limit.

The convergence speed of the EM scheme is highly sensitive to the signal-to-noise ratio of the data, as was documented in Olsson et al. (2006), whereas the QN algorithm is more robust to this condition. In Bermond and Cardoso (1999), it was shown that the magnitude of the update of $\mathbf{A}$ scales inversely with the SNR. By varying the SNR in the synthetic data and applying the EM algorithm, it was confirmed that the predicted convergence slowdown occurs at high SNR. In contrast, the QN algorithm was found to be much more robust to the noise conditions of the data. Figure 3 shows the SER performance of the two algorithms as computed following a fixed number

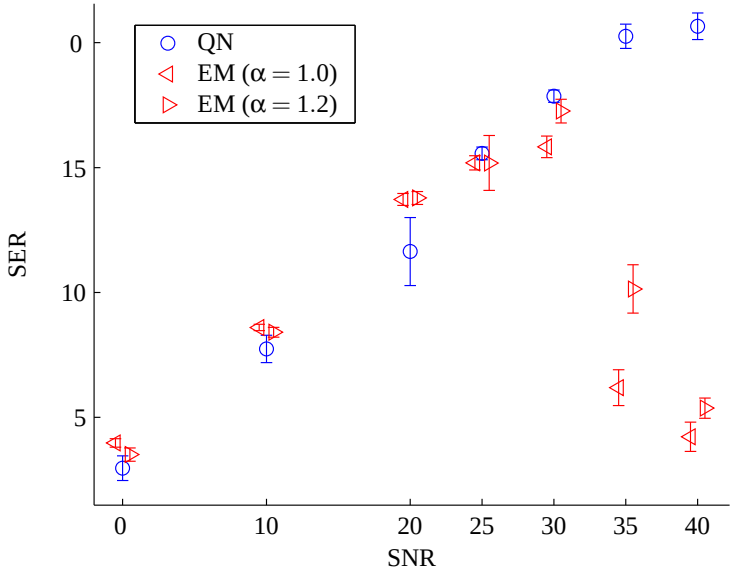


Figure 3: Convergence properties of the EM and QN algorithms as measured on the synthetic data (autoregressive sources). The signal-to-error ratio (SER) was computed in a range of SNR following 300 iterations. As the SNR increases, more accurate estimates are provided by all algorithms, but the number of iterations required increases more dramatically for the EM algorithm. Results are shown for the basic EM algorithm as well as for the step-size adjusted version.

of iterations ($i_{\max} = 300$). It should be noted that the time consumption per iteration is similar for the two algorithms, since a similar number of E-step computations is used (and E-steps all but dominate the cost).

For the purpose of analyzing the HN model, a synthetic data set was generated. The fundamental frequency of the harmonic component was sampled uniformly in a range, see Figure 4, amplitudes and phases, $K = 4$, were drawn from a Gaussian distribution and subsequently normalized such that $||\mathbf{c}_i|| = 1$. The parameters of the model were estimated using the EM algorithm on data, which was constructed as SNR $= 20$dB, HNR $= 20$dB. The fundamental frequency search grid was defined by 101 evenly spaced points in the generative range. In Figure 4, true and learned parameters are displayed. A close match between the true and estimated harmonics is observed.

In cases when the sources are truly harmonic and noisy, it is expected that the AR model performs worse than the HN model. This is due to the fact that a harmonic mean structure is required for the model to be unbiased. The AR model will compensate by estimating a larger variance, q_i, leading to suboptimal inference. In Figure 5, the bias is quantified by measuring the performance gap between the HN and AR models for varying HNR. The source parameters were estimated by the EM algorithm, whereas the mixing matrix, $\mathbf{A}$, was assumed known.

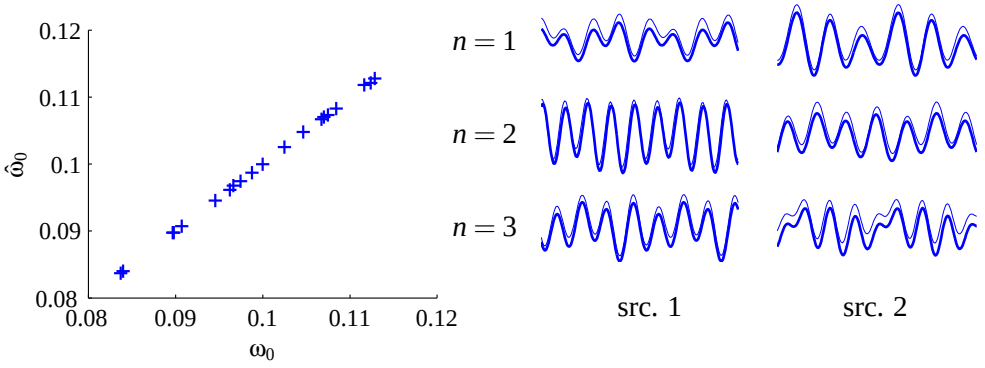


Figure 4: The true and estimated parameters from synthetic mixtures of harmonic-and-noisy sources as obtained by the EM algorithm. Left: fundamental frequencies in all frames. Right: the waveforms of the true (bold) and estimated harmonic components. For visualization purposes, the estimated waveform was shifted by a small offset.

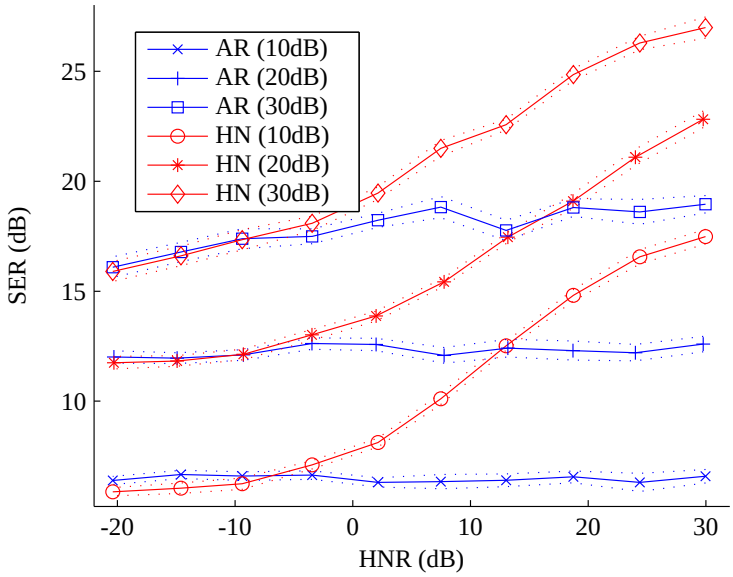


Figure 5: The signal-to-error ratio (SER) performance of the autoregressive (AR) and harmonic-and-noisy (HN) models for the synthetic data set ($N = 100$) in which the harmonic-to-noise ratio (HNR) was varied. Results are reported for SNR $= 10, 20, 30$dB. The results indicate that the relative advantage of using the correct model (HN) can be significant. The error-bars represent the standard deviation of the mean.

6. Speech Mixtures

The separation of multiple speech sources from room mixtures has potential applications in hearing aids and speech recognition software (see, for example, Parra and Spence, 2000). For this purpose,

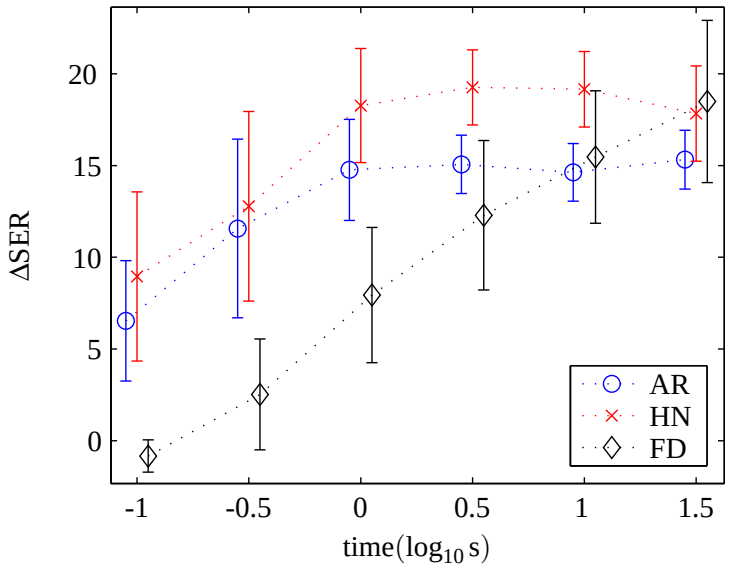


Figure 6: The separation performance (SER) on test mixtures as a function of the training data duration for the autoregressive (AR) and harmonic-and-noisy (HN) priors. Using the stochastic gradient (SG) algorithm, the parameters were estimated from the training data. Subsequently, the learned filters, $\mathbf{A}$, were applied to the test data, reestimating the source model parameters. The noise was constructed at 40dB and assumed known. For reference, a frequency domain (FD) blind source separation algorithm was applied to the data.

we investigate the models based on the autoregressive (AR) and harmonic-and-noisy source (HN) priors and compare with a standard frequency domain method (FD). More specifically, a learning curve was computed in order to illustrate that the inclusion of prior knowledge of speech benefits the separation of the speech sources. In Figure 6 is shown the relationship between the separation performance on test mixtures and the duration of the training data, confirming the hypothesis that the AR and HN models converge faster than the FD method. Furthermore it is seen that the HN model can obtain a larger SER than the AR model.

The mixtures were constructed by filtering speech signals (sampled at 8Hz) through a set of simulated room impulse responses, that is, $\mathbf{a}_{ij}$, and subsequently adding the filtered signals. The room impulse responses were constructed by simulating $Q = 2$ speakers and $P = 2$ microphones in an (ideal) anechoic room, the cartesian coordinates in the horizontal plane given (in m) by $\{(1,3),(3,3)\}$ and $\{(1.75,1),(2.25,1)\}$ for the speakers and microphones, respectively.[1]. This corresponds to a maximum distance of 1.25m between the speakers and the microphones, and a set of room impulse responses that are essentially Kronecker delta functions well represented using a filter length of $L = 8$.

1. A Matlab function, rir.m, implementing the image method (Allen and Berkley, 1979) is available at http://2pi.us/rir.html.

The SG algorithm was used to fit the model to the mixtures and subsequently infer the source signals. The speech data, divided into blocks of length $T = 200$, was preprocessed with a standard pre-emphasis filter, $H(z) = 1 - 0.95z^{-1}$, and inversely filtered prior to the SER calculations. From initial conditions ($q_i^n = 1$, $f_{i,j}^n = 0$, $c_{i,j}^n = 0$ and $a_{i,j,k}$ normally distributed, variance 0.01, for all i, j, n, k except $a_{1,1,1} = 1$, $a_{2,2,1} = 1$; $\omega_{0,i}^n$ was drawn from a uniform distribution corresponding to the interval $50 - 200$Hz), the algorithm was allowed $i_{\max} = 500$ iterations to converge and restarts were not necessary. The source model order was set to $p = 1$ (autoregression order) and in the case of the harmonic-and-noise model, the number of harmonics was set to $K = 6$. The complex JADE algorithm was employed in the frequency domain as the reference method (Cardoso and Souloumiac, 1993). In order to correct the permutations across the 101 frequencies, amplitude correlation between the bands was maximized (see, for example, Olsson and Hansen, 2006).

In order to qualitatively assess the effect of the two priors, a mixture of speech signals was constructed using $P = 2$ speech signals (a female and a male, shown in Figure 7a and b). They were mixed through artificial channels, $\mathbf{A}$, which were generated as in Section 5. Noise was added up to a level of 20dB. The EM algorithm was used to fit the source models to the mixtures. It is clear from Figure 7 c-f that the estimated harmonic model to a large extent explains the voiced parts of the speech signals, and the unvoiced parts to a lesser extent. In regions of rapid fundamental frequency variation, the harmonic part cannot be fitted as well (the frames are too long here). In Figure 7 g and h, the separation performances of the AR and HN models are contrasted. Most often, the HN performs better than the AR model. A notable exception occurs in the case when either speaker is silent, in which case the misfit of the HN model is more severe, suggesting that the performance can be improved by model control.

7. Conclusion

It is demonstrated that careful generative modelling is a viable approach to convolutive source separation and can yield improved results. Noisy observations, non-stationarity of the sources and small data volumes are examples of scenarios which benefit from the higher level of modelling detail.

The performance of the model was shown to depend on the choice of optimization scheme when the signal-to-noise ratio is high. In this case, the EM algorithm, which is often preferable for its conceptual and analytical simplicity, experiences a substantial slowdown, and alternatives must be employed. Such an alternative is a gradient-based quasi-Newton algorithm, which is shown to be particularly useful in low-noise settings. Furthermore, the required gradients are obtained in the process of deriving the EM algorithm.

The harmonic-and-noise model was investigated as a means to estimating more accurately a number of speech source signals from the their mixtures. Although a substantial improvement is shown to result when the sources are truly harmonic, the overall model is vulnerable to overfitting when the energy of one or more sources is locally near-zero. An improvement of the existing framework would be a model control scheme, such as variational Bayes, which could potentially cancel the negative impact of speaker silence. This is a topic for future research.

Acknowledgments

The authors thank the Oticon Fonden for providing financial support.

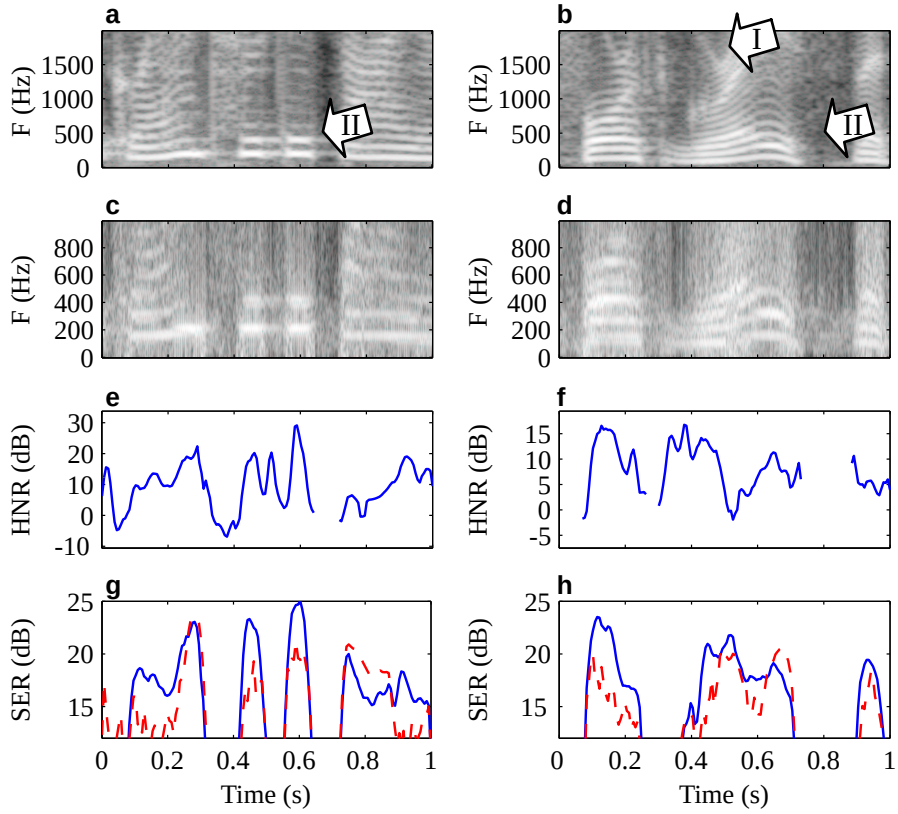


Figure 7: The source parameters of the autoregressive (AR) and harmonic-and-noisy (HN) models were estimated from $Q = 2$ convolutive mixtures using the EM algorithm. Spectrograms show the low-frequent parts of the original female (a) and male (b) speech sources. The appropriateness of the HN model can be assessed in c and d, which displays the re-synthesization of the two sources from the parameters ($K = 6$), as well as e and f, where the estimated ratio of harmonics to noise (HNR) is displayed. Overall the fit seem good, except at rapid variations of the fundamental frequency, for example, at (I), where the analysis frames are too long. The relative separation performance of the AR and HN models, which is shown in g and h for the two sources, confirms that the HN model is superior in most cases, with a notable exception in regions such as (II), where one of the speakers is silent. This implies a model complexity mismatch which is more severe for the more complex HN model.

Appendix A.

Below, an example of an M-step update derivation is shown for $\mathbf{F}^n$. As a by-product of the analysis, the derivative for the gradient-based optimizers appears. Care must be devised in obtaining the derivatives, since $\mathbf{F}^n$ is a structured matrix, for example, certain elements are one and zero. Therefore, the cost-function is expressed in terms of $\mathbf{f}_i^n$ rather than $\mathbf{F}^n$. Since all variables, which are here

indexed by the block identifier, n, are Gaussian, we have that:

$$\begin{aligned}\mathcal{J}(\theta) &= -\frac{1}{2}\sum_{n=1}^{N}\Big[\sum_{i=1}^{P}\Big\{\log|\Sigma_i^n| + \Big\langle\left(\mathbf{s}_{i,1}^n-\mu_i^n\right)^T\left(\Sigma_i^n\right)^{-1}\left(\mathbf{s}_{i,1}^n-\mu_i^n\right)\Big\rangle\Big\} \\ &\quad +(T-1)\sum_{i=1}^{P}\log q_i^n + \frac{1}{q_i^n}\sum_{t=2}^{T}\sum_{i=1}^{P}\Big\langle\left(s_{i,t}^n-\left(\mathbf{f}_i^n\right)^T\mathbf{s}_{i,t-1}^n\right)^2\Big\rangle \\ &\quad +T\log\det\mathbf{R}+\sum_{t=1}^{T}\Big\langle\left(\mathbf{x}_t^n-\mathbf{A}\mathbf{s}_t^n\right)^T\mathbf{R}^{-1}\left(\mathbf{x}_t^n-\mathbf{A}\mathbf{s}_t^n\right)\Big\rangle\Big].\end{aligned}$$

The vector derivative of $\mathcal{J}(\theta)$ with respect to $\mathbf{f}_i^n$ is:

$$\frac{d\mathcal{J}(\theta)}{d\mathbf{f}_i^n} = \frac{1}{q_i^n}\left[\sum_{t=2}^{T}\Big\langle\mathbf{s}_{i,t-1}^n\left(\mathbf{s}_{i,t-1}^n\right)^\top\Big\rangle\mathbf{f}_i^n-\sum_{t=2}^{T}\left\langle\mathbf{s}_{i,t-1}^n s_{i,t}^n\right\rangle\right].$$

This was the desired gradient, which is directly applicable in a gradient-based algorithm. By equating to zero and solving, the M-step update is derived:

$$\mathbf{f}_{i,\mathtt{new}}^n = \left[\sum_{t=2}^{T}\Big\langle\mathbf{s}_{i,t-1}^n\left(\mathbf{s}_{i,t-1}^n\right)^\top\Big\rangle\right]^{-1}\sum_{t=2}^{T}\left\langle\mathbf{s}_{i,t-1}^n s_{i,t}^n\right\rangle.$$

References

J. Allen and D. A. Berkley. Image method for efficiently simulating small-room acoustics. *Journal of the Acoustical Society of America*, 65:943–950, 1979.

J. Anemüller and B. Kollmeier. Amplitude modulation decorrelation for convolutive blind source separation. In *Proc. ICA 2000*, pages 215–220, 2000.

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

O. Bermond and J.-F. Cardoso. Approximate likelihood for noisy mixtures. In *Proc. ICA*, pages 325–330, 1999.

M. Brandstein. On the use of explicit speech modeling in microphone array applications. In *Proc. ICASSP*, pages 3613–3616, 1998.

J.-F. Cardoso, H. Snoussi, J. Delabrouille, and G. Patanchon. Blind separation of noisy Gaussian stationary sources. Application to cosmic microwave background imaging. In *Proc. EUSIPCO*, pages 561–564, 2002.

J. F. Cardoso and A. Souloumiac. Blind beamforming for non-gaussian signals. *IEE Proceedings F*, 140(6):362–370, 1993.

P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistics Society, Series B*, 39:1–38, 1977.

M. Dyrholm and L. K. Hansen. CICAAR: Convolutive ICA with an auto-regressive inverse model. In *Proc. ICA 2004*, pages 594–601, 2004.

P. A. Højen-Sørensen, Ole Winther, and Lars Kai Hansen. Mean-field approaches to independent component analysis. *Neural Computation*, 14(4):889–918, 2002.

A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, Inc, 2001.

R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering ASME Transactions*, 83:95–107, 1960.

T.W. Lee, A. J. Bell, and R. H. Lambert. Blind separation of delayed and convolved sources. In *Advances of Neural Information Processing Systems*, volume 9, page 758, 1997.

R.J. McAulay and T.F. Quateri. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 34(4):744–754, 1986.

M. McKeown, L.K. Hansen, and T.J. Sejnowski. Independent component analysis for fmri: What is signal and what is noise? *Current Opinion in Neurobiology*, 13(5):620–629, 2003.

E. Moulines, J. Cardoso, and E. Cassiat. Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. In *Proc. ICASSP*, volume 5, pages 3617–3620, 1997.

H. B. Nielsen. UCMINF - an algorithm for unconstrained nonlinear optimization. Technical Report IMM-Rep-2000-19, Technical University of Denmark, 2000.

R. K. Olsson and L. K. Hansen. A harmonic excitation state-space approach to blind separation of speech. In *Advances in Neural Information Processing Systems*, volume 17, pages 993–1000, 2005.

R. K. Olsson and L. K. Hansen. Blind separation of more sources than sensors in convolutive mixtures. In *International Conference on Acoustics, Speech and Signal Processing*, 2006.

R. K. Olsson, K. B. Petersen, and T. Lehn-Schiøler. State-space models - from the EM algorithm to a gradient approach. *Neural Computation - to appear*, 2006.

L. Parra and C. Spence. Convolutive blind separation of non-stationary sources. *IEEE Transactions, Speech and Audio Processing*, pages 320–7, 5 2000.

B. A. Pearlmutter and L. C. Parra. A context-sensitive generalization of ICA. In *In Advances in Neural Information Processing Systems 9*, pages 613–619, 1997.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

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

R. Salakhutdinov and S. Roweis. Adaptive overrelaxed bound optimization methods. In *International Conference on Machine Learning*, pages 664–671, 2003.

R. Salakhutdinov, S. T. Roweis, and Z. Ghahramani. Optimization with EM and Expectation-Conjugate-Gradient. In *International Conference on Machine Learning*, volume 20, pages 672–679, 2003.

R. Shumway and D. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Anal.*, 3(4):253–264, 1982.

E. Weinstein, M. Feder, and A. V. Oppenheim. Multi-channel signal separation by decorrelation. *IEEE Transactions on Speech and Audio Processing*, 1(4), 1993.

D. Yellin and E. Weinstein. Multichannel signal separation: Methods and analysis. *IEEE Transactions on Signal Processing*, 44(1):106–118, 1996.

Journal of Machine Learning Research 7 (2006) 2603-2620 Submitted 9/05; Revised 10/06; Published 12/06

On Representing and Generating Kernels by Fuzzy Equivalence Relations

Bernhard Moser BERNHARD.MOSER@SCCH.AT
Knowledge-Based Technology
Software Competence Center Hagenberg
A-4232 Hagenberg, Austria

Editor: Ralf Herbrich

Abstract

Kernels are two-placed functions that can be interpreted as inner products in some Hilbert space. It is this property which makes kernels predestinated to carry linear models of learning, optimization or classification strategies over to non-linear variants. Following this idea, various kernel-based methods like support vector machines or kernel principal component analysis have been conceived which prove to be successful for machine learning, data mining and computer vision applications. When applying a kernel-based method a central question is the choice and the design of the kernel function. This paper provides a novel view on kernels based on fuzzy-logical concepts which allows to incorporate prior knowledge in the design process. It is demonstrated that kernels mapping to the unit interval with constant one in its diagonal can be represented by a commonly used fuzzy-logical formula for representing fuzzy rule bases. This means that a great class of kernels can be represented by fuzzy-logical concepts. Apart from this result, which only guarantees the existence of such a representation, constructive examples are presented and the relation to unlabeled learning is pointed out.

Keywords: kernel, triangular norm, T-transitivity, fuzzy relation, residuum

1. Motivation

Positive-definiteness plays a prominent role especially in optimization and machine learning due to the fact that two-place functions with this property, so-called kernels, can be represented as inner products in some Hilbert space. Thereby, optimization techniques conceived on the basis of linear models can be extended to non-linear algorithms. For a survey of applications see, for example, Jolliffe (1986), Schölkopf and Smola (2002) and Schölkopf et al. (1998).

Recently in Moser (2006) it was shown that kernels with values from the unit interval can be interpreted as fuzzy equivalence relations motivated by the idea that kernels express a kind of similarity. This means that the concept of fuzzy equivalence relations, or synonymously fuzzy similarity relations, is more general than that of kernels, provided only values in the unit interval are considered. Fuzzy equivalence relations distinguish from Boolean equivalence relations by a many-valued extension of transitivity which can be interpreted as many-valued logical model of the statement "*IF x is similar to y AND y is similar to z THEN x is similar to z*". In contrast to the Boolean case, in many-valued logics the set of truth values is extended such that also assertions, for example, whether two elements x and y are similar, can be treated as a matter of degree. The standard model for the set of (quasi) truth values of fuzzy logic and other many-valued logical systems is the unit interval. If $E(x,y)$ represents the (quasi) truth value of the statement that x is

similar to y, then the many-valued version of transitivity is modeled by

$$T(E(x,y),E(y,z)) \leq E(x,z)$$

where T is a so-called triangular norm which is an extension of the Boolean conjunction. This many-valued concept for transitivity is called T-transitivity. For a survey on triangular norms see, for example, Dubois and Prade (1985), Gottwald (1986), Gottwald (1993) and Klement et al. (2000), and for fuzzy equivalence relations and T-transitivity see, for example, Bodenhofer (2003), Höhle (1993), Höhle (1999), Klement et al. (2000), and Zadeh (1971).

Based on the semantics of fuzzy logic, this approach allows to incorporate knowledge-based models for the design of kernels. From this perspective, the most interesting mathematical question is how positive-semidefinite fuzzy equivalence relations can be characterized or at least constructed under some circumstances. At least for some special cases, proofs are provided in Section 4, which motivate further research aiming at establishing a more general theory on the positive-definiteness of fuzzy equivalence relations. These cases are based on the most prominent representatives of triangular norms, that is the Minimum, the Product and the Łukasiewicz t-norm.

The paper is structured as follows. First of all, in Section 2, some basic prerequisites concerning kernels and fuzzy relations are outlined. In Section 3, a result about the T-transitivity of kernels from Moser (2006) is cited and interpreted as existence statement that guarantees a representation of kernels mapping to the unit interval with constant 1 in its diagonal by a certain, commonly used, fuzzy-logical construction of a fuzzy equivalence relation. Finally, in contrast to the pure existence theorem of Section 3, in Section 4 constructive examples of fuzzy equivalence relations are provided which are proven to be kernels. In a concluding remark, the relationship to the problem of labeled and unlabeled learning is pointed out.

2. Prerequisites

This section summarizes definitions and facts from the theory of kernels as well as from fuzzy set theory which are needed later on.

2.1 Kernels and Positive-Semidefiniteness Preserving Functions

There is an extensive literature concerning kernels and kernel-based methods like support vector machines or kernel principal component analysis especially in the machine learning, data mining and computer vision communities. For an overview and introduction, see, for example, Schölkopf and Smola (2002). Here we present only what is needed later on. For completeness let us recall the basic definition for kernels and positive-semidefiniteness.

Definition 1 *Let $\mathcal{X}$ be a non-empty set. A real-valued function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be a kernel iff it is symmetric, that is, $k(x,y) = k(y,x)$ for all $x,y \in \mathcal{X}$, and positive-semidefinite, that is, $\sum_{i,j=1}^{n} c_i c_j k(x_i,x_j) \geq 0$ for any $n \in \mathbb{N}$, any choice of $x_1,\ldots,x_n \in \mathcal{X}$ and any choice of $c_1,\ldots,c_n \in \mathbb{R}$.*

One way to generate new kernels from known kernels is to apply operations which preserve the positive-semidefiniteness property. A characterization of such operations is provided by C. H. FitzGerald (1995).

Theorem 2 ***(Closeness Properties of Kernels)*** *Let $f : \mathbb{R}^n \to \mathbb{R}$, $n \in \mathbb{N}$, then $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ given by*

$$k(x,y) := f(k_1(x,y),\ldots,k_n(x,y))$$

is a kernel for any choice of kernels $k_1,\dots,k_n$ on $\mathcal{X}\times\mathcal{X}$ iff f is the real restriction of an entire function on $\mathbb{C}^n$ of the form

$$f(x_1,\dots,x_n)=\sum_{r_1\geq 0,\dots,r_n\geq 0} c_{r_1,\dots,r_n}x_1^{r_1}\cdots x_n^{r_n} \tag{1}$$

where $c_{r_1,\dots,r_n}\geq 0$ for all nonnegative indices $r_1,\dots,r_n$.

2.2 Triangular Norms

Triangular norms have been originally studied within the framework of probabilistic metric spaces, see Schweizer and Sklar (1961) and Schweizer and Sklar (1983). In this context, t-norms proved to be an appropriate concept when dealing with triangle inequalities. Later on, t-norms and their dual version, t-conorms, have been used to model conjunction and disjunction for many-valued logic, see Dubois and Prade (1985), Gottwald (1986), Gottwald (1993) and Klement et al. (2000).

Definition 3 *A function $T:[0,1]^2\to[0,1]$ is called* t-norm *(triangular norm), if it satisfies the following conditions:*

(i)	$\forall x,y\in[0,1]:$	$T(x,y)=T(y,x)$	(*commutativity*)
(ii)	$\forall x,y,z\in[0,1]:$	$T(x,T(y,z))=T(T(x,y),z)$	(*associativity*)
(iii)	$\forall x,y,z\in[0,1]:$	$y\leq z\Longrightarrow T(x,y)\leq T(x,z)$	(*monotonicity*)
(iv)	$\forall x,y\in[0,1]:$	$T(x,1)=x\wedge T(1,y)=y$	(*boundary condition*)

Further, a t-norm is called Archimedean *if it is continuous and satisfies*

$$x\in(0,1)\Rightarrow T(x,x)<x.$$

Due to its associativity, many-placed extensions $T_n:[0,1]^n\to[0,1]$, $n\in\mathbb{N}$, of a t-norm T are uniquely determined by

$$T_n(x_1,\dots,x_n)=T(x_1,T_{n-1}(x_2,\dots,x_n)).$$

Archimedean t-norms are characterized by the following representation theorem due to Ling (1965):

Theorem 4 *Let $T:[0,1]^2\to[0,1]$ be a t-norm. Then T is Archimedean if, and only if, there is a continuous, strictly decreasing function $f:[0,1]\to[0,\infty]$ with $f(1)=0$ such that for $x,y\in[0,1]$,*

$$T(x,y)=f^{-1}(\min(f(x)+f(y),f(0))).$$

By setting $g(x)=\exp(-f(x))$, Ling's characterization yields an alternative representation with a multiplicative generator function

$$T(x,y)=g^{-1}(\max(g(x)\,g(y),g(0))).$$

For $g(x)=x$ we get the product $T_P(x,y)=xy$. The setting $f(x)=1-x$ yields the so-called Łukasiewcz t-norm $T_L(x,y)=\max(x+y-1,0)$. Due to Ling's theorem 4 an Archimedean t-norm T is isomorphic either to T_L or T_P, depending on whether the additive generator takes a finite value at 0 or not. In the former case, the Archimedean t-norm is called *non-strict*, in the latter it is called *strict*.

A many-valued model of an implication is provided by the so-called residuum given by

$$\overrightarrow{T}(a,b) = \sup\{c \in [0,1] | T(a,c) \leq b\} \tag{2}$$

where T is a left-continuous t-norm. Equation (2) is uniquely determined by the so-called adjunction property

$$\forall a,b,c \in [0,1] : T(a,b) \leq c \Leftrightarrow a \leq \overrightarrow{T}(b,c). \tag{3}$$

Consequently, the operator

$$\overleftrightarrow{T}(a,b) = \min\left\{\overrightarrow{T}(a,b), \overrightarrow{T}(b,a)\right\} \tag{4}$$

models a biimplication. For details, for example, see Gottwald (1986) and Klement et al. (2000). Tables 1 and 2 list examples of t-norms with their induced residuum $\overrightarrow{T}$. For further examples see, for example, Klement et al. (2000).

$T_{\cos}(a,b)$	=	$\max(ab - \sqrt{1-a^2}\sqrt{1-b^2}, 0)$
$T_L(a,b)$	=	$\max(a+b-1,0)$
$T_P(a,b)$	=	ab
$T_M(a,b)$	=	$\min(a,b)$

Table 1: Examples of t-norms

$\overrightarrow{T}_{\cos}(a,b)$	=	$\begin{cases} \cos(\arccos(b) - \arccos(a)) & \text{if } a > b, \\ 1 & \text{else} \end{cases}$
$\overrightarrow{T}_L(a,b)$	=	$\min(b-a+1,1)$
$\overrightarrow{T}_P(a,b)$	=	$\begin{cases} \frac{b}{a} & \text{if } a > b, \\ 1 & \text{else} \end{cases}$
$\overrightarrow{T}_M(a,b)$	=	$\begin{cases} b & \text{if } a > b, \\ 1 & \text{else} \end{cases}$

Table 2: Examples of residuums

2.3 T-Equivalences

If we want to classify based on a notion of similarity or indistinguishability, we face the problem of transitivity. For instance, let us consider two real numbers to be indistinguishable if and only if they differ by at most a certain bound $\varepsilon > 0$, this is modeled by the relation $\sim_\varepsilon$ given by $x \sim_\varepsilon y :\Leftrightarrow |x-y| < \varepsilon$, $\varepsilon > 0$, $x,y \in \mathbb{R}$. Note that the relation $\sim_\varepsilon$ is not transitive and, therefore, not an equivalence relation. The transitivity requirement turns out to be too strong for this example. The problem of identification and transitivity in the context of similarity of physical objects was early pointed out and discussed philosophically by Poincaré (1902) and Poincaré (1904). In the framework of fuzzy logic, the way to overcome this problem is to model similarity by fuzzy relations based on a many-valued concept of transitivity, see Bodenhofer (2003), Höhle (1993), Höhle (1999), Klement et al. (2000) and Zadeh (1971).

Definition 5 *A function $E : X^2 \longrightarrow [0,1]$ is called a* fuzzy equivalence relation*, or synonymously,* T-equivalence *with respect to the t-norm T if it satisfies the following conditions:*

$$\begin{array}{lll} (i) & \forall x \in X: & E(x,x) = 1 \qquad (\textit{reflexivity}) \\ (ii) & \forall x,y \in X: & E(x,y) = E(y,x) \qquad (\textit{symmetry}) \\ (iii) & \forall x,y,z \in X: & T(E(x,y),E(y,z)) \leq E(x,z) \qquad (\textit{T-transitivity}). \end{array}$$

The value $E(x,y)$ can be also looked at as the (quasi) truth value of the statement "*x is equal to y*". Following this semantics, *T-transitivity* can be seen as a many-valued model of the proposition, "*If x is equal to y and y is equal to z, then x is equal to z*". T-equivalences for Archimedean t-norms are closely related to metrics and pseudo-metrics as shown by Klement et al. (2000) and Moser (1995).

Theorem 6 *Let T be an Archimedean t-norm given by*

$$\forall a,b \in [0,1] : T(a,b) = f^{-1}(\min(f(a)+f(b), f(0))),$$

where $f : [0,1] \to [0,\infty]$ is a strictly decreasing, continuous function with $f(1) = 0$.
(i) If $d : X^2 \to [0,\infty[$ is a pseudo-metric, then the function $E_d : X^2 \to [0,1]$ defined by

$$E_d(x,y) = f^{-1}(\min(d(x,y), f(0)))$$

is a T-equivalence with respect to the t-norm T.
(ii) If $E : X^2 \to [0,1]$ is a T-equivalence relation, then the function $d_E : X^2 \to [0,\infty]$ defined by

$$d_E(x,y) = f(E(x,y))$$

is a pseudo-metric.

Another way to construct T-equivalences is to employ $\overrightarrow{T}$-operators. The proof of the following assertion can be found in Trillas and Valverde (1984), Kruse et al. (1993) and Kruse et al. (1994).

Theorem 7 *Let T be a left-continuous t-norm, $\overleftrightarrow{T}$ its induced biimplication, $\mu_i : \mathcal{X} \to [0,1]$, $i \in I$, I non-empty; then $E : \mathcal{X} \times \mathcal{X} \to [0,1]$ given by*

$$E(x,y) = \inf_{i \in I} \overleftrightarrow{T}(\mu_i(x), \mu_i(y)) \tag{5}$$

is a T-equivalence relation.

For further details on T-equivalences see also Boixader and Jacas (1999), Höppner et al. (2002), Jacas (1988), Trillas et al. (1999) and Valverde (1985).

3. Representing Kernels by T-Equivalences

It is interesting that the concept of kernels, which is motivated by geometric reasoning in terms of inner products and mappings to Hilbert spaces and which is inherently formulated by algebraic terms, is closely related to the concept of fuzzy equivalence relations as demonstrated and discussed in more detail in Moser (2006). In this section, we start with the result that any kernel $k : \mathcal{X} \times \mathcal{X} \to [0,1]$ with $k(x,x) = 1$ for all $x \in \mathcal{X}$ is T-transitive and, therefore, a fuzzy equivalence relation. The proof can be found in Moser (2006), see also Appendix A.1.

Theorem 8 *Any kernel $k : \mathcal{X} \times \mathcal{X} \to [0,1]$ with $k(x,x) = 1$ is (at least) $T_{\cos}$-transitive, where*

$$T_{\cos}(a,b) = \max\{a\,b - \sqrt{1-a^2}\sqrt{1-b^2}, 0\}. \tag{6}$$

The nomenclature is motivated by the fact that the triangular norm defined by Equation (6) is an Archimedean t-norm which is generated by the arcosine function as its additive generator. From this result, the following existence theorem can be derived, which guarantees that any kernel under consideration can be represented by the fuzzy-logical formula given by (5). In fuzzy systems, this formula is commonly used for modeling rule bases (see, for example, Kruse et al., 1993, 1994).

Theorem 9 *Let $\mathcal{X}$ be a non-empty universe of discourse, $k : \mathcal{X} \times \mathcal{X} \to [0,1]$ a kernel in the sense of Definition 1 and $k(x,x) = 1$ for all $x \in \mathcal{X}$; then there is a family of membership functions $\mu_i : \mathcal{X} \to [0,1]$, $i \in I$, I non-empty and a t-norm T, such that*

$$\forall x,y \in \mathcal{X} : k(x,y) = \inf_{i \in I} \overleftrightarrow{T}(\mu_i(x), \mu_i(y)). \tag{7}$$

Proof. Let us set $I := \mathcal{X}$, $\mu_{x_0}(x) = k(x,x_0)$ and let us choose $T_{\cos}$ as t-norm. For convenience let us denote

$$h(x,y) = \inf_{x_0 \in \mathcal{X}} \overleftrightarrow{T}_{\cos}(\mu_{x_0}(x), \mu_{x_0}(y)),$$

which is equivalent to

$$h(x,y) = \inf_{x_0 \in \mathcal{X}} \overleftrightarrow{T}_{\cos}(k(x_0,x), k(x_0,y)).$$

According to Theorem 8, k is $T_{\cos}$-transitive, that is,

$$\forall x_0,x,y \in \mathcal{X} : \overleftrightarrow{T}_{\cos}(k(x_0,x), k(x_0,y)) \leq k(x,y).$$

This implies that $h(x,y) \leq k(x,y)$ for all $x,y \in \mathcal{X}$. Now let us consider the other inequality. Due to the adjunction property (3), we obtain

$$T_{\cos}(k(x,y), k(x_0,y)) \leq k(x,x_0) \Leftrightarrow k(x,y) \leq \overrightarrow{T}_{\cos}(k(x_0,y), k(x,x_0))$$

and

$$T_{\cos}(k(x,y), k(x_0,x)) \leq k(y,x_0) \Leftrightarrow k(x,y) \leq \overrightarrow{T}_{\cos}(k(x_0,x), k(y,x_0)),$$

from which it follows that

$$\forall x,y,x_0 \in \mathcal{X} : k(x,y) \leq \min\{\overrightarrow{T}_{\cos}(k(x_0,y), k(x,x_0)), \overrightarrow{T}_{\cos}(k(x_0,x), k(y,x_0))\}.$$

Hence by Definition 4,

$$\forall x,y \in \mathcal{X} : k(x,y) \leq h(x,y)$$

which ends the proof. ■

For an arbitrary choice of fuzzy membership functions, there is no necessity that the resulting relation (7) implies positive-semidefiniteness and, therefore, a kernel. For an example of a $T_{\cos}$-equivalence which is not a kernel see Appendix A.4. Theorem 9 guarantees only the existence of a representation of the form (5) but it does not tell us how to construct the membership functions μ_i. In the following section, we provide examples of fuzzy equivalence relations which yield kernels for any choice of membership functions.

4. Constructing Kernels by Fuzzy Equivalence Relations

In the Boolean case, positive-definiteness and equivalence are synonymous, that is, a Boolean relation $R : \mathcal{X} \times \mathcal{X} \to \{0,1\}$ is positive-definite if and only if R is the indicator function of an equivalence relation $\cong$, that is, $R(x,y) = 1$ if $x \cong y$ and $R(x,y) = 0$ if $x \ncong y$. For a proof, see Appendix A.2. This relationship can be used to obtain an extension to fuzzy relations as given by the next theorem whose proof can be found in the Appendix A.3.

Theorem 10 *Let $\mathcal{X}$ be a non-empty universe of discourse, $\mu_i : \mathcal{X} \to [0,1]$, $i \in I$, I non-empty; then the fuzzy equivalence relation $E_M : \mathcal{X} \times \mathcal{X} \to [0,1]$ given by*

$$E_M(x,y) = \inf_{i \in I} \overleftrightarrow{T}_M(\mu_i(x), \mu_i(y))$$

is positive-semidefinite.

In the following, the most prominent representatives of Archimedean t-norms, the Product T_P and the Łukasiewicz t-norm T_L, are used to construct positive-semidefinite fuzzy similarity relations. Though the first part can also be derived from a result due to Yaglom (1957) that characterizes isotropic stationary kernels by its spectral representation, here we prefer to present a direct, elementary proof. Compare also Bochner (1955) and Genton (2001).

Theorem 11 *Let $\mathcal{X}$ be a non-empty universe of discourse, $\nu : \mathcal{X} \to [0,1]$ and let $h : [0,1] \to [0,1]$ be an isomorphism of the unit interval that can be expanded in the manner of Equation (1), that is $h(x) = \sum_k c_k x^k$ with $c_k \geq 0$; then the fuzzy equivalence relations $E_{L,h}, E_{P,h} : \mathcal{X} \times \mathcal{X} \to [0,1]$ given by*

$$E_{L,h}(x,y) = h\left(\overleftrightarrow{T}_L\left(h^{-1}(\nu(x)), h^{-1}(\nu(y))\right)\right) \tag{8}$$

and

$$E_{P,h}(x,y) = h\left(\overleftrightarrow{T}_P\left(h^{-1}(\nu(x)), h^{-1}(\nu(y))\right)\right) \tag{9}$$

are positive-semidefinite.

Proof. To prove the positive-definiteness of the two-placed functions $E_{L,h}$ and $E_{P,h}$ given by equations (8) and (9) respectively, we have to show that

$$\sum_{i,j=1}^{n} E_{L,h}(x_i,x_i)\, c_i c_j \geq 0, \sum_{i,j=1}^{n} E_{P,h}(x_i,x_j)\, c_i c_j \geq 0$$

for any $n \in \mathbb{N}$ and any choice of $x_1, \ldots, x_n \in \mathcal{X}$, respectively. According to an elementary result from Linear Algebra this is equivalent to the assertion that the determinants ($1 \leq m \leq n$)

$$D_m = det\left[\left(E(x_i,x_j)\right)_{i,j \in \{1,\ldots,m\}}\right]$$

of the minors of the matrix $\left(E(x_i,x_j)\right)_{i,j}$ satisfy

$$\forall m \in \{1,\ldots,n\} : D_m \geq 0,$$

where E denotes either $E_{L,h}$ or $E_{P,h}$. Recall that the determinant of a matrix is invariant with respect to renaming the indices, that is, if $\sigma : \{1,\ldots,n\} \to \{1,\ldots,n\}$ is a permutation then

$$det\left[(a_{ij})_{i,j}\right] = det\left[(a_{\sigma(i)\sigma(j)})_{i,j}\right].$$

For convenience, let denote $\mu_i = h^{-1}(\nu(x_i))$. Then, without loss of generality, we may assume that the values μ_i are ordered monotonically decreasing, that is,

$$\mu_i \geq \mu_j \text{ for } i < j. \tag{10}$$

Case T_L: Note that $\overleftrightarrow{T}_L(a,b) = \min\{\overrightarrow{T}_L(a,b), \overrightarrow{T}_L(b,a)\} = 1 - |a-b|$. Then we have to show that for all dimensions $n \in \mathbb{N}$, the determinant of

$$E^{(n)} = (1 - |\mu_i - \mu_j|)_{i,j \in \{1,\ldots,n\}}$$

is non-negative, that is

$$det[E^{(n)}] \geq 0.$$

Due to the assumption (10), we have

$$1 - |\mu_i - \mu_j| = \begin{cases} 1 - (\mu_i - \mu_j) & \text{if } i \leq j, \\ 1 - (\mu_j - \mu_i) & \text{else} \end{cases}$$

which yields

$$E^{(n)} = \begin{pmatrix} 1 & 1-(\mu_1-\mu_2) & \ldots & 1-(\mu_1-\mu_{n-1}) & 1-(\mu_1-\mu_n) \\ 1-(\mu_1-\mu_2) & 1 & \ldots & 1-(\mu_2-\mu_{n-1}) & 1-(\mu_2-\mu_n) \\ 1-(\mu_1-\mu_3) & 1-(\mu_2-\mu_3) & \ldots & 1-(\mu_3-\mu_{n-1}) & 1-(\mu_3-\mu_n) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1-(\mu_1-\mu_{n-1}) & 1-(\mu_2-\mu_{n-1}) & \ldots & 1 & 1-(\mu_{n-1}-\mu_n) \\ 1-(\mu_1-\mu_n) & 1-(\mu_2-\mu_n) & \ldots & 1-(\mu_{n-1}-\mu_n) & 1 \end{pmatrix}.$$

Now let us apply determinant-invariant elementary column operations to simplify this matrix by subtracting the column with index $i-1$ from the column with index i, $i \geq 2$. This yields

$$\tilde{E}^{(n)} = \begin{pmatrix} 1 & \mu_2-\mu_1 & \ldots & \mu_{n-1}-\mu_{n-2} & \mu_n-\mu_{n-1} \\ 1-(\mu_1-\mu_2) & -(\mu_2-\mu_1) & \ldots & \mu_{n-1}-\mu_{n-2} & \mu_n-\mu_{n-1} \\ 1-(\mu_1-\mu_3) & -(\mu_2-\mu_1) & \ldots & \mu_{n-1}-\mu_{n-2} & \mu_n-\mu_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1-(\mu_1-\mu_{n-1}) & -(\mu_2-\mu_1) & \ldots & -(\mu_{n-2}-\mu_{n-1}) & \mu_n-\mu_{n-1} \\ 1-(\mu_1-\mu_n) & -(\mu_2-\mu_1) & \ldots & -(\mu_{n-2}-\mu_{n-1}) & -(\mu_{n-1}-\mu_n) \end{pmatrix}.$$

Therefore,

$$\begin{aligned} \alpha &= \prod_{i=2}^{n} (\mu_{i-1} - \mu_i) \geq 0 \qquad (11) \\ det[E^{(n)}] &= det[\tilde{E}^{(n)}] = \alpha \, det[\hat{E}_n], \end{aligned}$$

where

$$\hat{E}^{(n)} = \begin{pmatrix} 1 & -1 & \ldots & -1 & -1 \\ 1-(\mu_1-\mu_2) & +1 & \ldots & -1 & -1 \\ 1-(\mu_1-\mu_3) & +1 & \ldots & -1 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1-(\mu_1-\mu_{n-1}) & +1 & \ldots & +1 & -1 \\ 1-(\mu_1-\mu_n) & +1 & \ldots & +1 & +1 \end{pmatrix}. \tag{12}$$

Let us apply Laplacian determinant expansion by minors to the first column of matrix (12), that is

$$det[A] = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} det[A_{ij}]$$

where $A = (a_{ij})$ is an $n \times n$-matrix, j arbitrarily chosen from $\{1,\ldots,n\}$ and A_{ij} is the matrix corresponding to the cofactor a_{ij} obtained by canceling out the i-th row and the j-th column from A (see, for example, Muir, 1960). For $n = 1$, we get the trivial case $det[\hat{E}^{(1)}] = 1$. Note that the first and the last rows of the matrices $\hat{E}^{(n)}_{i,1}$ for $1 < i < n$ only differ by their signum, consequently the minors $det[\hat{E}^{(n)}_{i,1}]$ for $1 < i < n$, $n \geq 2$, are vanishing, that is,

$$det[A_{i,1}] = 0, \text{ for } 1 < i < n.$$

Therefore, according to the Laplacian expansion, we get

$$det[\hat{E}^{(n)}] = 1 \cdot det[\hat{E}^{(n)}_{1,1}] + (-1)^n (1 - (\mu_1 - \mu_n)) \cdot det[\hat{E}^{(n)}_{1,n}]. \tag{13}$$

Observe that

$$\begin{aligned} det[\hat{E}^{(n)}_{1,1}] &= 2^{n-2} \\ det[\hat{E}^{(n)}_{1,n}] &= (-1)^{n-1} 2^{n-2}. \end{aligned}$$

Consequently, Equation (13) simplifies to

$$\begin{aligned} det[\hat{E}^{(n)}] &= 2^{n-2} \left(1 + (-1)^n (-1)^{n-1} 2^{n-2} (1 - (\mu_1 - \mu_n))\right) \\ &= 2^{n-2} (1 - (1 - (\mu_1 - \mu_n))) \\ &= 2^{n-2} (\mu_1 - \mu_n) \\ &\geq 0 \end{aligned}$$

which together with (11) proves the first case.

Case T_P: First of all, let us compute $\overleftrightarrow{T}_P(a,b) = \min\{\overrightarrow{T}_P(a,b), \overrightarrow{T}_L(b,a)\}$. Hence,

$$\overleftrightarrow{T}_P(a,b) = \begin{cases} \min\{\frac{b}{a}, \frac{a}{b}\} & \text{if } a,b > 0, \\ 0 & \text{if } a = 0 \text{ and } b > 0\ , \\ 0 & \text{if } b = 0 \text{ and } a > 0\ , \\ 1 & \text{if } a = 0 \text{ and } b = 0\ . \end{cases}$$

Again, without loss of generality, let us suppose that the values μ_i, $i \in \{1,\ldots,n\}$ are ordered monotonically decreasing, that is $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n$. Before checking the general case, let us consider the special case of vanishing μ-values. For this, let us assume for the moment that

$$\mu_i = \begin{cases} > 0 & \text{if } i < i_0\ , \\ 0 & \text{else} \end{cases}$$

which implies that $\overleftrightarrow{T}_P(\mu_i,\mu_j) = 0$ for $i < i_0$ and $j \geq i_0$ and $\overleftrightarrow{T}_P(\mu_i,\mu_j) = 1$ for $i \geq i_0$ and $j \geq i_0$. This leads to a decomposition of the matrix

$$E^{(n)} = \left(\overleftrightarrow{T}_P(\mu_i,\mu_j)\right)_{ij}$$

such that

$$det[E^{(n)}] = det[E^{(i_0-1)}] \cdot det[I_{n-i_0-1}]$$

where I_k denotes the $k \times k$-matrix with constant entries 1, hence $det[I_{n-i_0-1}] \in \{0,1\}$. Therefore, we may assume that

$$\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n > 0.$$

Then we have to show that for all dimensions $n \in \mathbb{N}$, the determinant of

$$E^{(n)} = \left(\min \left\{ \frac{\mu_i}{\mu_j}, \frac{\mu_j}{\mu_i} \right\} \right)_{i,j \in \{1,\ldots,n\}}$$

is non-negative, that is

$$det[E^{(n)}] \geq 0.$$

Consider

$$E^{(n)} = \begin{pmatrix} 1 & \frac{\mu_2}{\mu_1} & \ldots & \frac{\mu_{n-1}}{\mu_1} & \frac{\mu_n}{\mu_1} \\ \frac{\mu_2}{\mu_1} & 1 & \ldots & \frac{\mu_{n-1}}{\mu_2} & \frac{\mu_n}{\mu_2} \\ \frac{\mu_3}{\mu_1} & \frac{\mu_3}{\mu_2} & \ldots & \frac{\mu_{n-1}}{\mu_3} & \frac{\mu_n}{\mu_3} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\mu_{n-1}}{\mu_1} & \frac{\mu_{n-1}}{\mu_2} & \ldots & 1 & \frac{\mu_n}{\mu_{n-1}} \\ \frac{\mu_n}{\mu_1} & \frac{\mu_n}{\mu_2} & \ldots & \frac{\mu_n}{\mu_{n-1}} & 1 \end{pmatrix}. \tag{14}$$

Now, multiply the i-th column by $-\mu_{i+1}/\mu_i$ and add it to the $(i+1)$-th column of matrix (14), $1 \leq i < n$, then we get

$$\tilde{E}^{(n)} = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ * & 1 - \left(\frac{\mu_2}{\mu_1}\right)^2 & \ldots & 0 & 0 \\ * & * & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ * & * & \ldots & 1 - \left(\frac{\mu_{n-1}}{\mu_{n-2}}\right)^2 & 0 \\ * & * & \ldots & * & 1 - \left(\frac{\mu_n}{\mu_{n-1}}\right)^2 \end{pmatrix} \tag{15}$$

where $*$ is a placeholder for any real value. By this, the determinant of the matrix in Equation (15) readily turns out to be

$$det[E^{(n)}] = det[\tilde{E}^{(n)}] = \prod_{i=1}^{n-1} \left(1 - \left(\frac{\mu_{i+1}}{\mu_i} \right)^2 \right) \geq 0$$

which together with Theorem (2) ends the proof. ■

Note that relations (8) and (9) are T-transitive with respect to the corresponding isomorphic Archimedean t-norms,

$$T_{L,h}(x,y) = h(T_L(h^{-1}(x), h^{-1}(x))) \text{ and } T_{P,h}(x,y) = h(T_P(h^{-1}(x), h^{-1}(x))),$$

respectively.

Corollary 12 *Let $\mathcal{X}$ be a non-empty universe of discourse, $\mu_i : \mathcal{X} \to [0,1]$, $\lambda_i \in]0,1]$ with $\sum_i \lambda_i = 1$ where $i \in \{1,\ldots,n\}$, $n \in \mathbb{N}$, then the fuzzy equivalence relations $\tilde{E}_L, \tilde{E}_P : \mathcal{X} \times \mathcal{X} \to [0,1]$ given by*

$$\tilde{E}_L(x,y) = \sum_{i=1}^{n} \lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(x), \mu_i(y)) \tag{16}$$

and

$$\tilde{E}_P(x,y) = \prod_{i=1}^{n} \left(\overset{\leftrightarrow}{T}_P(\mu_i(x), \mu_i(y)) \right)^{\lambda_i} \tag{17}$$

are T_L- and T_P-equivalences, respectively, and kernels.

Proof. First of all, let us check the T_L-transitivity of formula (16). This can readily be shown by means of the definition of T_L and the T_L-transitivity of $\overset{\leftrightarrow}{T}_L$ due to the following inequalities:

$$\begin{aligned}
T_L\left(\sum_{i=1}^{n} \lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(x),\mu_i(y)), \sum_{i=1}^{n} \lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(y),\mu_i(yz) \right) &= \\
\max\left\{ \sum_{i=1}^{n} \lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(x),\mu_i(y)) + \sum_{i=1}^{n} \lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(y),\mu_i(z)) - 1, 0 \right\} &= \\
\max\left\{ \sum_{i=1}^{n} \lambda_i \left(\overset{\leftrightarrow}{T}_L(\mu_i(x),\mu_i(y)) + \sum_{i=1}^{n} \lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(y),\mu_i(z)) - 1 \right), 0 \right\} &\leq \\
\max\left\{ \sum_{i=1}^{n} \lambda_i T_L \left(\overset{\leftrightarrow}{T}_L(\mu_i(x),\mu_i(y)), \sum_{i=1}^{n} \lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(y),\mu_i(z)) \right), 0 \right\} &\leq \\
\max\left\{ \sum_{i=1}^{n} \lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(x),\mu_i(z)), 0 \right\} &= \\
\lambda_i \overset{\leftrightarrow}{T}_L(\mu_i(x),\mu_i(z)).
\end{aligned}$$

This, together with the T_P-transitivity of $\overset{\leftrightarrow}{T}_P$, proves that the formulas given by (16) and (17) are T_L- and T_P-equivalences, respectively.

Expanding the factors of formula (17) yields

$$\left(\overset{\leftrightarrow}{T}_P(\mu_i(x),\mu_i(y)) \right)^{\lambda_i} = \begin{cases} 1 & \text{if } \mu_i(x) = \mu_i(y) = 0, \\ \frac{\min(\mu_i^{\lambda_i}(x), \mu_i^{\lambda_i}(y))}{\max(\mu_i^{\lambda_i}(x), \mu_i^{\lambda_i}(y))} & \text{else} \end{cases} \tag{18}$$

which by comparing case T_P of the proof of Theorem 11 shows that the left-hand side of Equation (18) is positive-semidefinite.

As the convex combination and the product are special cases of positive-semidefiniteness preserving functions according to Theorem 1, the functions defined by equations (16) and (17) prove to be again positive-semidefinite and, therefore, kernels. ■

It is interesting to observe that both formulas (16) and (17) can be expressed in the form, $f(\|\tau(x) - \tau(y)\|_1)$, where $f : I \to [0,1]$, I some interval, is a strictly decreasing function, $\tau : \mathcal{X} \to I^n$, I some interval, $\tau(x) = (\tau_1(x),\ldots,\tau_n(x))$ and $\|\tau(x)\|_1 = \sum_{i=1}^{n} |\tau_i(x)|$. Indeed, for Equation (16) let us define

$$\begin{aligned}
f_L : [0,1] \to [0,1], \quad f_L(a) &= 1 - a \\
\tau_L : \mathcal{X} \to [0,1]^n, \quad \tau_L(x) &= (\lambda_1 \mu_1(x), \ldots, \lambda_n \mu_n(x))
\end{aligned}$$

and for Equation (17) and positive membership functions μ_i, $\mu_i(x) > 0$ for all $x \in \mathcal{X}$, let us define

$$\begin{aligned} f_P : [0,\infty[\to [0,1], \quad f_P(a) &= e^{-a} \\ \tau_P : \mathcal{X} \to]-\infty,1]^n, \quad \tau_P(x) &= (\lambda_1 \ln(\mu_1(x)), \ldots, \lambda_n \ln(\mu_n(x))) \end{aligned}$$

Therefore, we get

$$\tilde{E}_L(x,y) = 1 - \|\tau_L(x) - \tau_L(y)\|_1 \tag{19}$$

$$\tilde{E}_P(x,y) = e^{-\|\tau_P(x) - \tau_P(y)\|_1}. \tag{20}$$

While formulas (19) and (20) provide a geometrical interpretation by means of the norm $\|.\|_1$, the corresponding formulas (16) and (17) yield a semantical model of the assertion

"*IF* x is equal to y with respect to feature μ_1 *AND* ... *AND* x is equal to y with respect to feature μ_n *THEN* x is equal to y"

as aggregation of biimplications in terms of fuzzy logic. While in the former case, the aggregation has some compensatory effect, the latter is just a conjunction in terms of the Product triangular norm. For details on aggregation operators see, for example, Saminger et al. (2002) and Calvo et al. (2002).

The formulas (16) and (17) coincide for the following special case. If the membership functions μ_i are indicator functions of sets $A_i \subseteq \mathcal{X}$ which form a partition of $\mathcal{X}$, then the kernels (16) and (17) reduce to the indicator function characterizing the Boolean equivalence relation induced by this partition $\{A_1, \ldots, A_n\}$.

The formulas (16) and (17) for general membership functions therefore provide kernels which can be interpreted to be induced by a family of fuzzy sets and, in particular, by fuzzy partitions, that is, families of fuzzy sets fulfilling some criteria which extend the axioms for a Boolean partition in a many-valued logical sense. For definitions and further details on fuzzy partitions see, for example, De Baets and Mesiar (1998), Demirci (2003) and Höppner and Klawonn (2003).

It is a frequently used paradigm that the decision boundaries for a classification problem lie between clusters rather than intersecting them. Due to this cluster hypothesis, the problem of designing kernels based on fuzzy partitions is closely related to the problem of learning kernels from unlabeled data. For further details on semi-supervised learning see, for example, Seeger (2002), Chapelle et al. (2003) and T. M. Huang (2006). It is left to future research to explore this relationship to the problem of learning from labeled and unlabeled data and related concepts like covariance kernels.

5. Conclusion

In this paper, we have presented a novel view on kernels from a fuzzy logical point of view. Particularly, the similarity-measure aspect of a kernel is addressed and investigated by means of the so-called T-transitivity which is characteristic for fuzzy equivalence relations. As a consequence, we derived that a large class of kernels can be represented in a way that is commonly used for representing fuzzy rule bases. In addition to this proof for the existence of such a representation, constructive examples are presented. It is the idea of this research to look for a combination of knowledge-based strategies with kernel-based methods in order to facilitate a more flexible designing process of kernels which also allows to incorporate prior knowledge. Further research aims at

analyzing the behavior of kernels constructed in this way when applied in the various kernel methods like support vector machines, kernel principal components analysis and others. In particular, it is intended to focus on the problem of learning kernels from unlabeled data where the fuzzy partitions are induced by appropriate clustering principles.

Acknowledgments

Bernhard Moser gratefully acknowledges partial support by the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the K*plus* Competence Center Program. Furthermore special thanks go to the anonymous reviewers who gave helpful suggestions and to Felix Kossak for careful proof-reading.

Appendix A.

For sake of completeness the following sections provide proofs regarding Theorem 8, the characterization of kernels in the Boolean case and the construction of kernels by means of the minimum t-norm T_M. Furthermore, in Section A.4 an example of a non-positive-semidefinite $T_{\cos}$-equivalence is given.

A.1 Proof of Theorem 8

Let us start with the analysis of 3-dimensional matrices.

Lemma 13 *Let $M = (m_{ij})_{ij} \in [0,1]^{3\times 3}$ be a 3×3 symmetric matrix with $m_{ii} = 1$, $i = 1,2,3$; then M is positive-semidefinite iff for all $i,j,k \in \{1,2,3\}$ there holds*

$$m_{ij}m_{jk} - \sqrt{1-m_{ij}^2}\sqrt{1-m_{jk}^2} \leq m_{ik}$$

Proof. For simplicity, let $a = m_{1,2}$, $b = m_{1,3}$ and $c = m_{2,3}$. Then the determinant of M, $Det(M)$, is a function of the variables a,b,c given by

$$D(a,b,c) = 1 + 2abc - a^2 - b^2 - c^2.$$

For any choice of a,b, the quadratic equation $D(a,b,c) = 0$ can be solved for c, yielding two solutions $c_1 = c_1(a,b)$ and $c_2 = c_2(a,b)$ as functions of a and b,

$$\begin{aligned} c_1(a,b) &= ab - \sqrt{1-a^2}\sqrt{1-b^2} \\ c_2(a,b) &= ab + \sqrt{1-a^2}\sqrt{1-b^2}. \end{aligned}$$

Obviously, for all $|a| \leq 1$ and $|b| \leq 1$, the values $c_1(a,b)$ and $c_2(a,b)$ are real. By substituting $a = \cos\alpha$ and $b = \cos(\beta)$ with $\alpha,\beta \in [0,\frac{\pi}{2}]$, it becomes readily clear that

$$\begin{aligned} c_1(a,b) &= c_1(\cos(\alpha),\cos(\beta)) \\ &= \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta) \\ &= \cos(\alpha+\beta) \in [-1,1] \end{aligned}$$

and, analogously,

$$\begin{aligned} c_2(a,b) &= c_2(\cos(\alpha),\cos(\beta)) \\ &= \cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta) \\ &= \cos(\alpha-\beta)\in[-1,1]. \end{aligned}$$

As for all $a,b\in[-1,1]$ the determinant function $D_{a,b}(c) := D(a,b,c)$ is quadratic in c with negative coefficient for c^2, there is a uniquely determined maximum at $c_0(a,b) = ab$. Note that for all $a,b\in[-1,1]$, we have

$$c_1(a,b)\le c_0(a,b)\le c_2(a,b)$$

and

$$D(a,b,c_0(a,b)) = 1+2ab(ab)-a^2-b^2-(ab)^2 = (1-a^2)(1-b^2)\ge 0.$$

Therefore, $D(a,b,c)\ge 0$ if and only if $c\in[c_1(a,b),c_2(a,b)]$.

Recall from linear algebra that by renaming the indices, the determinant does not change. Therefore, without loss of generality, we may assume that

$$a\ge b\ge c.$$

For convenience, let $Q=\{(x,y,z)\in[0,1]^3 | x\ge y\ge z\}$. Then, obviously, for any choice of $a,b\in[0,1]$ there holds $(a,b,c_1(a,b))\in Q$. Elementary algebra shows that $(a,b,c_2(a,b))\in Q$ is only the case for $a=b=1$. As for $a=b=1$ the two solutions c_1, c_2 coincide, that is, $c_1(1,1)=c_2(1,1)=1$, it follows that for any choice of $(a,b,c)\in Q$, there holds $D(a,b,c)\ge 0$ if and only if

$$c_1(a,b) = ab-\sqrt{1-a^2}\sqrt{1-b^2}\le c. \tag{21}$$

If $(a,b,c)\notin Q$, then the inequality (21) is trivially satisfied which together with (21) proves the lemma ■

Now Theorem 8 immediately follows from Definition (1), Lemma (13) and the characterizing inequality (21).

A.2 Characterization of Kernels in the Boolean Case

The following lemma and proposition can also be found as an exercise in Schölkopf and Smola (2002).

Lemma 14 *Let $\sim$ be an equivalence relation on $\mathcal{X}$ and let $k:\mathcal{X}\times\mathcal{X}\to\{0,1\}$ be induced by $\sim$ via $k(x,y)=1$ if and only if $x\sim y$; then k is a kernel.*

Proof. By definition of positive-definiteness, let us consider an arbitrary sequence of elements $x_1,\ldots,x_n$. Then there are at most n equivalence classes $Q_1,\ldots,Q_m$ on the set of indices $\{1,\ldots,n\}$, $m\le n$, where $\bigcup_{i=1,\ldots,m} Q_i=\{1,\ldots,n\}$ and $Q_i\cap Q_j=\emptyset$ for $i\neq j$. Note that $k(x_i,x_j)=0$ if the indices

i, j belong to different equivalence classes. Then, for any choice of reals $c_1,\ldots,c_n$, we obtain

$$\begin{aligned}\sum_{i,j} c_i c_j k(x_i,x_j) &= \sum_{p=1}^{m}\sum_{i,j\in Q_p} c_i c_j k(x_i,x_j)\\ &= \sum_{p=1}^{m}\sum_{i,j\in Q_p} c_i c_j \cdot 1\\ &= \sum_{p=1}^{m}\left(\sum_{i\in Q_p} c_i\right)^2\\ &\geq 0\end{aligned}$$

■

Proposition 15 *$k:\mathcal{X}\times\mathcal{X}\to\{0,1\}$ with $k(x,x)=1$ for all $x\in\mathcal{X}$ is a kernel if and only if it is induced by an equivalence relation.*

Proof. It only remains to be shown that if k is a kernel, then it is the indicator function of an equivalence relation, that is, it is induced by an equivalence relation. If k is a kernel, according to Lemma 13, for all $x,y,z\in\mathcal{X}$, it has to satisfy $T_{\cos}(k(x,y),k(y,z))\leq k(x,z)$, which implies,

$$k(x,y)=1,\quad k(y,z)=1\Longrightarrow k(x,z)=1.$$

Obviously, we have $k(x,x)=1$ and $k(x,y)=k(y,x)$ due to the reflexivity and symmetry assumption of k, respectively. ■

A.3 Constructing Kernels by T_M

For convenience let us recall the basic notion of an α-cut from fuzzy set theory:

Definition 16 *Let $\mathcal{X}$ be a non-empty set and $\mu:\mathcal{X}\to[0,1]$; then*

$$[\mu]_\alpha=\{x\in\mathcal{X}\,|\,\mu(x)\geq\alpha\}$$

is called the α-cut of the membership function μ.

Lemma 17 *$k:\mathcal{X}\times\mathcal{X}\to[0,1]$ is a T_M-equivalence if and only if all α-cuts of k are Boolean equivalence relations.*

Proof.

(i) Let us assume that k is a T_M-equivalence. Let $\alpha\in[0,1]$, then by definition,

$$[k]_\alpha=\{(x,y)\in\mathcal{X}\times\mathcal{X}\,|\,k(x,y)\geq\alpha\}.$$

In order to show that $[k]_\alpha$ is a Boolean equivalence, the axioms for reflexivity, symmetry and transitivity have to be shown. Reflexivity and symmetry are trivially satisfied as for all $x,y\in\mathcal{X}$, there holds by assumption that $k(x,x)=1$ and $k(x,y)=k(y,x)$. In order to show transitivity, let us consider $(x,y),(y,z)\in[k]_\alpha$, that means $k(x,y)\geq\alpha$ and $k(y,z)\geq\alpha$; then by the T_M-transitivity assumption it follows that

$$\alpha\leq\min(k(x,y),k(y,z))\leq k(x,z),$$

hence $(x,z)\in[k]_\alpha$.

(ii) Suppose now that all α-cuts of k are Boolean equivalence relations. Then, in particular, $[k]_\alpha$ with $\alpha = 1$ is reflexive, hence $k(x,x) = 1$ for all $x \in \mathcal{X}$. The symmetry of k follows from the fact that for all $\alpha \in [0,1]$ and pairs $(x,y) \in [k]_\alpha$, by assumption, we have $(y,x) \in [k]_\alpha$. In order to show the T_M-transitivity property, let us consider arbitrarily chosen elements $x,y,z \in \mathcal{X}$. Let $\alpha = \min(k(x,y),k(y,z))$; then by the transitivity assumption of $[k]_\alpha$, it follows that $(x,z) \in [k]_\alpha$, consequently

$$k(x,z) \geq \alpha = \min(k(x,y),k(y,z)).$$

■

Proposition 18 *If $k : \mathcal{X} \times \mathcal{X} \to [0,1]$ is a T_M-equivalence then it is positive-semidefinite.*

Proof. Choose arbitrary elements $x_1,\ldots,x_n \in \mathcal{X}$ and consider the set of values which are taken by all combinations $k(x_i,x_j)$, $i,j \in \{1,\ldots,n\}$ and order them increasingly, that is

$$\left\{k(x_i,x_j) \,|\, i,j \in \{1,\ldots,n\}\right\} = \{\alpha_1,\ldots,\alpha_m\},$$

where $0 \leq \alpha_1 \leq \cdots \alpha_m \leq 1$. Observe that for all pairs (x_i,x_j), $i,j \in \{1,\ldots,n\}$ there holds

$$k(x_i,x_j) = \sum_{v=2}^{m} (\alpha_v - \alpha_{v-1}) \mathbf{1}_{[k]_{\alpha_v}}(x_i,x_j) + \alpha_1 \mathbf{1}_{[k]_{\alpha_1}}(x_i,x_j)$$

showing that on the set $\{x_1,\ldots,x_n\} \times \{x_1,\ldots,x_n\}$, the function k is a linear combination of indicator functions of Boolean equivalences (which are positive-semidefinite by Proposition 15) with non-negative coefficients and, consequently, it has to be positive semidefinite. ■

A.4 Example of a Non-Positive-Semidefinite $T_{\cos}$-Equivalence

For dimensions $n > 3$, the $T_{\cos}$-transitivity is no longer sufficient to guarantee positive semi-definiteness. Consider, for example $A_n = (a_{ij}^{(n)})_{ij}$ where

$$a_{ij}^{(n)} = \begin{cases} \lambda & \text{if } \min(i,j) = 1, \max(i,j) > 1\,, \\ 1 & \text{if } i = j, \\ 0 & \text{else}\,. \end{cases} \tag{22}$$

Choose $\lambda = 1/\sqrt{2}$, then $T_{\cos}(\lambda,\lambda) = 0$, hence we have $T_{\cos}(a_{ij}^{(n)}, a_{jk}^{(n)}) \leq a_{ik}^{(n)}$ for all indices $i,j,k \in 1,\ldots,n$. As $det(A_n) < 0$ for $n > 3$, the matrix A_n cannot be positive-semidefinite though the $T_{\cos}$-transitivity conditions are satisfied.

References

S. Bochner. *Harmonic Analysis and the Theory of Probability*. University of California Press, Los Angeles, California, 1955.

U. Bodenhofer. A note on approximate equality versus the Poincaré paradox. *Fuzzy Sets and Systems*, 133(2):155–160, 2003.

D. Boixader and J. Jacas. T-indistinguishability operators and approximate reasoning via CRI. In D. Dubois, E. P. Klement, and H. Prade, editors, *Fuzzy Sets, Logics and Reasoning about Knowledge*, volume 15 of *Applied Logic Series*, pages 255–268. Kluwer Academic Publishers, Dordrecht, 1999.

A. Pinkus C. H. FitzGerald, C.A. Micchelli. Functions that preserve families of positive semidefinite matrices. *Linear Alg. and Appl.*, 221:83–102, 1995.

T. Calvo, G. Mayor, and R. Mesiar, editors. *Aggregation Operators*, volume 97 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, Heidelberg, 2002.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. volume 15 of *NIPS*. 2003.

B. De Baets and R. Mesiar. T-partitions. *Fuzzy Sets and Systems*, 97:211–223, 1998.

M. Demirci. On many-valued partitions and many-valued equivalence relations. *Internat. J. Uncertain. Fuzziness Knowledge-Based Systems*, 11(2):235–253, 2003.

D. Dubois and H. Prade. A review of fuzzy set aggregation connectives. *Inform. Sci.*, 36:85–121, 1985.

M. G. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

S. Gottwald. Fuzzy set theory with t-norms and Φ-operators. In A. Di Nola and A. G. S. Ventre, editors, *The Mathematics of Fuzzy Systems*, volume 88 of *Interdisciplinary Systems Research*, pages 143–195. Verlag TÜV Rheinland, Köln, 1986.

S. Gottwald. *Fuzzy Sets and Fuzzy Logic*. Vieweg, Braunschweig, 1993.

U. Höhle. Fuzzy equalities and indistinguishability. In *Proc. 1st European Congress on Fuzzy and Intelligent Technologies*, volume 1, pages 358–363, Aachen, 1993.

U. Höhle. The Poincaré paradox and non-classical logics. In D. Dubois, E. P. Klement, and H. Prade, editors, *Fuzzy Sets, Logics and Reasoning about Knowledge*, volume 15 of *Applied Logic Series*, pages 7–16. Kluwer Academic Publishers, Dordrecht, 1999.

F. Höppner and F. Klawonn. Improved fuzzy partitions for fuzzy regression models. *Internat. J. Approx. Reason.*, 32:85–102, 2003.

F. Höppner, F. Klawonn, and P. Eklund. Learning indistinguishability from data. *Soft Computing*, 6 (1):6–13, 2002.

J. Jacas. On the generators of T-indistinguishability operators. *Stochastica*, 12:49–63, 1988.

I. T. Jolliffe. *Principal Component Analysis*. Springer Verlag, New York, 1986.

E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*, volume 8 of *Trends in Logic*. Kluwer Academic Publishers, Dordrecht, 2000.

R. Kruse, J. Gebhardt, and F. Klawonn. *Fuzzy-Systeme*. B. G. Teubner, Stuttgart, 1993.

R. Kruse, J. Gebhardt, and F. Klawonn. *Foundations of Fuzzy Systems*. John Wiley & Sons, New York, 1994.

C. H. Ling. Representation of associative functions. *Publ. Math. Debrecen*, 12:189–212, 1965.

B. Moser. On the t-transitivity of kernels. *Fuzzy Sets and Systems*, 157:1787–1796, 2006.

B. Moser. *A New Approach for Representing Control Surfaces by Fuzzy Rule Bases*. PhD thesis, Johannes Kepler Universität Linz, October 1995.

T. Muir. *A Treatise on the Theory of Determinants*. Dover, New York, 1960.

H. Poincaré. *La Science et l'Hypothése*. Flammarion, Paris, 1902.

H. Poincaré. *La Valeur de la Science*. Flammarion, Paris, 1904.

S. Saminger, R. Mesiar, and U. Bodenhofer. Domination of aggregation operators and preservation of transitivity. *Internat. J. Uncertain. Fuzziness Knowledge-Based Systems*, 10(Suppl.):11–35, 2002.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, 2002.

B. Schölkopf, A. J. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

B. Schweizer and A. Sklar. Associative functions and statistical triangle inequalities. *Publ. Math. Debrecen*, 8:169–186, 1961.

B. Schweizer and A. Sklar. *Probabilistic Metric Spaces*. North-Holland, Amsterdam, 1983.

M. Seeger. Covariance kernels from bayesian generative models. *Neural Information Processing Systems*, 14:905–912, 2002.

I. Kopriva T. M. Huang, V. Kecman. *Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi-supervised, and Unsupervised Learning*. Springer-Verlag, Berlin, 2006.

E. Trillas and L. Valverde. An inquiry into indistinguishability operators. In H. J. Skala, S. Termini, and E. Trillas, editors, *Aspects of Vagueness*, pages 231–256. Reidel, Dordrecht, 1984.

E. Trillas, S. Cubillo, and E. Castiñeira. Menger and Ovchinnikov on indistinguishabilities revisited. *Internat. J. Uncertain. Fuzziness Knowledge-Based Systems*, 7(3):213–218, 1999.

L. Valverde. On the structure of F-indistinguishability operators. *Fuzzy Sets and Systems*, 17(3): 313–328, 1985.

A. M. Yaglom. Some classes of random fields in n-dimensional space, related to stationary random processes. *Theory of Probability and its Applications*, 2:273–320, 1957.

L. A. Zadeh. Similarity relations and fuzzy orderings. *Inform. Sci.*, 3:177–200, 1971.

Journal of Machine Learning Research 7 (2006) 2621-2650 Submitted 4/06; Revised 10/06; Published 12/06

A Robust Procedure For Gaussian Graphical Model Search From Microarray Data With p Larger Than n

Robert Castelo ROBERT.CASTELO@UPF.EDU
Departament de Ciències Experimentals i de la Salut
Universitat Pompeu Fabra
Dr. Aiguader 88, E-08003 Barcelona, Spain

Alberto Roverato ALBERTO.ROVERATO@UNIBO.IT
Dipartimento di Scienze Statistiche
Università di Bologna
Via Belle Arti 41, I-40126 Bologna, Italy

Editor: Max Chickering

Abstract

Learning of large-scale networks of interactions from microarray data is an important and challenging problem in bioinformatics. A widely used approach is to assume that the available data constitute a random sample from a multivariate distribution belonging to a Gaussian graphical model. As a consequence, the prime objects of inference are *full-order partial correlations* which are partial correlations between two variables given the remaining ones. In the context of microarray data the number of variables exceed the sample size and this precludes the application of traditional structure learning procedures because a sampling version of full-order partial correlations does not exist. In this paper we consider *limited-order partial correlations*, these are partial correlations computed on marginal distributions of manageable size, and provide a set of rules that allow one to assess the usefulness of these quantities to derive the independence structure of the underlying Gaussian graphical model. Furthermore, we introduce a novel structure learning procedure based on a quantity, obtained from limited-order partial correlations, that we call the *non-rejection rate*. The applicability and usefulness of the procedure are demonstrated by both simulated and real data.

Keywords: Gaussian distribution, gene network, graphical model, microarray data, non-rejection rate, partial correlation, small-sample inference

1. Introduction

High-throughput experimental technologies developed within the field of molecular biology allow one to observe in real time the activity of thousands of biomolecules in the cell under tens of different experimental conditions. These technologies, known as *microarray* technologies, are able to put together in a solid substrate (a chip) of a few squared centimeters a bidimensional matrix (an array) formed by tens of thousands of probes. Each probe is specific to a nucleic acid sequence that recognizes (hybridises) marked samples (biomolecules) of complementary RNA (coming from the experimental conditions under study), quantifying the abundance of each recognized biomolecule. An open question within molecular biology research is to be able to describe the set of interactions, or biomolecular network, between the different functional elements in the genome that mediate the production of the biomolecules we observe through these high-throughput platforms. These data,

the so-called *microarray data*, can be seen as a random sample of a multivariate distribution defined by a set of random variables associated to the genome functional elements under study (e.g., genes). Each record corresponds to a vector of values describing the abundance of a particular kind of biomolecule (e.g., messenger RNA) produced by each genome functional element under a specific experimental condition (e.g., a specific tissue or cell line). Thus, a way to describe the interactions among the genome functional elements is by using conditional independencies and, more concretely, graphical models (see Pearl, 1988; Whittaker, 1990; Lauritzen, 1996) which have emerged as a powerful tool for the learning, description and manipulation of conditional independencies.

However, in a typical microarray data set the number of observations n (on the order of tens) is substantially smaller than the number of variables p (on the order of hundreds or even thousands) and this prevents us from applying directly most of the existing multivariate methods for structure learning of graphical models due to the difficulties in obtaining estimates of the joint probability distribution.

In this paper, we focus in Gaussian graphical models and investigate the role of marginal distributions in their structure learning. Firstly, we formally introduce the concept of *q-partial graph* that is a graph associated with the set of all marginal distributions of dimension $q+2$ and, furthermore, we provide a comprehensive description of the connection between a q-partial graph and the graph associated with the Gaussian graphical model of interest. Secondly, we propose a novel q-partial-correlations based procedure, *qp-procedure* hereafter, for structure learning of q-partial graphs based on a quantity that we call the *non-rejection rate*. The results of this paper can be applied also outside the biological context because they can be more generally useful whenever structure learning of a Gaussian graphical model is carried out in the special context in which (i) p is large compared to n, (ii) the underlying structure of the graphical model is sparse. Furthermore, the qp-procedure can also be regarded as a method to obtain shrinkage estimators of the covariance matrix. We remark that the theory of q-partial graphs is developed under the assumption of *faithfulness* of the probability distribution to its independence graph, however the qp-procedure is robust with respect to this assumption as we shall discuss at the end of the paper.

The paper is organized as follows. Sections 2 and 3 give the theory of Gaussian graphical models and their application to learning of biomolecular networks from microarray data, respectively. The theory of q-partial graphs is given in Section 4 whereas the required graph theory is provided in the Appendix. The qp-procedure is introduced in Section 5 where instances of its application to both simulated and real data are given and, finally, Section 6 contains a brief discussion.

2. Gaussian Graphical Models

In this section we review the Gaussian graphical model theory required for this paper. For a full account of graphical model theory we refer to Cox and Wermuth (1996), Lauritzen (1996) and Whittaker (1990) whereas, for the theory relating to structure learning of graphical models we refer to Cowell et al. (1999), Edwards (2000), Jones et al. (2005) and Whittaker (1990).

Let $X_V \equiv X$ be a random vector indexed by $V = \{1,\ldots,p\}$ with probability distribution P_V and let $G = (V,E)$ be an undirected graph; see Appendix A for the graph theory used here. For a subset $A \subseteq V$, we denote by X_A the subvector of X indexed by A, and by P_A the associated marginal distribution. For a triplet $I,J,U \subseteq V$ we write $X_I \perp\!\!\!\perp X_J | X_U$ to denote that X_I is conditionally independent of X_J given X_U; we allow U to be the empty set to denote the marginal independence of X_I and X_J.

We say that P_V is (undirected) *Markov* with respect to G if it holds that $X_I \perp\!\!\!\perp X_J | X_U$ whenever U separates I and J in G; in particular this implies that if $(i,j) \in \bar{E}$ then $X_i \perp\!\!\!\perp X_j | X_{V\setminus\{i,j\}}$. Here $\bar{E}$ denotes the set of missing edges of $G = (V,E)$ as formally defined in Appendix A. We say that P_V is *faithful* to G if all the conditional independence relationships in P_V can be read off the graph G through the Markov property. Consider a graph $G' = (V,E')$ larger than G, $G \subseteq G'$. It is straightforward to check that if P_V is Markov with respect to G then it is also Markov with respect to G'. However, if P_V is faithful to G then it is faithful to G' if and only if $G = G'$.

Throughout this paper X_V is assumed to have a multivariate normal distribution with mean vector μ_V and positive definite covariance matrix $\Sigma_{VV} \equiv \Sigma$. Furthermore, we assume that P_V is both Markov and faithful with respect to an undirected graph $G = (V,E)$. Hence, for a subset $Q \subset V$ with $i, j \notin Q$ it holds that $X_i \perp\!\!\!\perp X_j | X_Q$ if and only if the partial correlation coefficient

$$\rho_{ij.Q} = \frac{-\kappa^A_{ij}}{\sqrt{\kappa^A_{ii}\kappa^A_{jj}}}$$

is equal to zero, where $A = Q \cup \{i,j\}$ and $K^A = \{\kappa^A_{ij}\}$ is the *concentration matrix* of X_A, $K^A = (\Sigma_{AA})^{-1}$ (Lauritzen, 1996, p. 130). Of special interest is the case $A = V$ because the concentration matrix $K^V \equiv K = \{\kappa_{ij}\}$ is the inverse of Σ and the structure of $G = (V,E)$ can be derived from the zero pattern of K. More specifically, it holds that (Lauritzen, 1996, Proposition 5.2)

$$k_{ij} = 0 \quad \Leftrightarrow \quad \rho_{ij.V\setminus\{i,j\}} = 0 \quad \Leftrightarrow \quad (i,j) \in \bar{E}, \tag{1}$$

and for this reason G is called the *concentration graph* of X_V. For $|Q| = q$, the parameter $\rho_{ij.Q}$ is called a *q-order partial correlation* of X_i and X_j, and if $q = p-2$, that is, $Q = V\setminus\{i,j\}$, we say that $\rho_{ij.Q}$ is the *full-order partial correlation* of X_i and X_j.

A *Gaussian graphical model* (Dempster, 1972) is the family of p-variate normal distributions that are Markov with respect to a given undirected graph $G = (V,E)$. Let $X^{(n)} = (X^1, \ldots, X^n)$ be a random sample from P_V. For a Gaussian graphical model with graph G the sufficient statistics are given by the sample mean vector and by the sample covariance matrices S_{CC} for $C \in \mathcal{C}$ where $\mathcal{C}$ is the set of cliques of G (Lauritzen, 1996, p. 132). It follows that, when G is complete the sufficient statistics are the sample mean and the sample covariance matrix S. Here, we consider problems in which the sample size is small, and it is thus important to recall that, for $A \subseteq V$, the sample covariance matrix S_{AA} from $X_A^{(n)}$ has full rank, with probability one, if and only if $n > |A|$ (Dykstra, 1970) and that a necessary condition for the computation of several statistical quantities such as the maximum likelihood estimates of K and of the partial correlations in (1) is that S_{CC} has full rank for all $C \in \mathcal{C}$.

Structure learning aims at identifying the structure $G = (V,E)$ with the fewest number of edges on the basis of the available data such that the underlying distribution P_V is undirected Markov over G. In a frequentist approach to inference, a basic operation to be performed in structure learning procedures is a statistical test for the hypothesis that a given partial correlation is zero, $\rho_{ij.Q} = 0$, since for $Q = V\setminus\{i,j\}$ this is equivalent to the hypothesis that $(i,j) \in \bar{E}$. If, for $A = Q \cup \{i,j\}$, X_A has an (unrestricted) normal distribution then the generalized likelihood ratio test for the hypothesis that $\rho_{ij.Q} = 0$ has form $L = -n\log(1-\hat{\rho}^2_{ij.Q})$ where $\hat{\rho}_{ij.Q} = -\hat{\kappa}^A_{ij}/\sqrt{\hat{\kappa}^A_{ii}\hat{\kappa}^A_{ii}}$ and $\widehat{K^A} = (S_{AA})^{-1}$ is the maximum likelihood estimate of K^A (Whittaker, 1990, p. 175). Under the null hypothesis, the asymptotic distribution of L is χ^2_1, even though for a small sample size the exact distribution of the

statistical test may be preferred; see Schäfer and Strimmer (2005a). An alternative way to verify the above hypothesis is provided by the connection between partial correlations and regression coefficients. More specifically, in the regression of X_i on $X_{A\setminus\{i\}}$ the regression coefficient associated with X_j is zero if and only if $\rho_{ij.Q} = 0$ (see Cox and Wermuth, 1996, p. 69). In the structure learning procedure proposed in this paper, to verify the absence of an edge from the unrestricted model we will apply the usual t test for zero regression coefficients because it is optimal, in the sense that it is Uniformly Most Powerful Unbiased (UMPU) (see Lehmann, 1986, p. 397).

3. Gaussian Graphical Models For biomolecular Networks

Microarray data quantify the abundance of biomolecules, commonly known as expression level, by probing functional elements along the genome which, without loss of generality, we shall hereafter refer to as genes. A set of p genes being probed define a vector of random variables X_i, $i = 1, \ldots, p$, that take normalized values of the expression levels of the corresponding genes. For every variable X_i there is vector of n values coming from n different experimental conditions forming the so-called expression profile. The microarray data consist of the expression profiles of a set of genes and form a snapshot of the interactions between the genes in terms of statistical (in)dependencies which, in principle, could be inferred through structure learning of Gaussian graphical models and thus leading to a description of the underlying biomolecular network in these terms. Hence, the prime object of interest is the inverse of the covariance matrix, also known as concentration matrix, whose zero pattern defines the structure of the graphical model, known then as concentration graph.

However, in contrast with the usual data sets found in the literature, on which structure learning of Gaussian graphical models is applied, microarray data constitute a challenging problem because microarray experiments typically measure the expression level of a large number of genes across a small number of experimental conditions. As a consequence of the scarcity of the data, the maximum likelihood of the inverse covariance matrix does not exist because the sample covariance matrix has full rank, with probability one, if and only if $n > p$ (Dykstra, 1970). This paper tackles this specific circumstance under which we perform structure learning of Gaussian graphical models with *small n and large p*.

An important observation in this context is that a growing body of biological evidence suggests that biomolecular networks have a sparse structure. This feature, usually regarded as an advantage, has been exploited in a number of ways to enable learning of Gaussian graphical models from microarray data (see, among others, Wong et al., 2003; Dobra et al., 2004; Wille et al., 2004; Wille and Bühlmann, 2006; Shäfer and Strimmer, 2005a, 2005b, 2005c) among which some methods work by obtaining shrinkage estimators of the covariance matrix (Wong et al., 2003; Shäfer and Strimmer, 2005c) while some other have made an attempt to learn an approximate version of the biomolecular network by using marginal distributions of dimension smaller than n. We shall discuss this latter approach in more detail below.

Instead of trying to learn the concentration graph of a Gaussian graphical model from microarray data, a tool employed by the bioinformatics community to describe interactions between genes is the *relevance network*; see Butte et al. (2000) and Steuer et al. (2003a, 2003b). In relevance networks missing edges denote zero correlations between pairs of genes, that in the Gaussian case imply marginal independence. In these graphs, edges are typically represented by undirected lines; nevertheless in the graphical model literature these models are known as *covariance graphs* (Cox and Wermuth, 1993, 1996) and edges are represented by either bidirected arrows or dashed undi-

rected lines. A correlation coefficient is zero if and only if the corresponding covariance is zero and therefore the structure of a covariance graph is derived from the zero pattern of the covariance matrix Σ. Although structure learning of covariance graphs is not straightforward (Drton and Perlman, 2004; Drton and Richardson, 2004), a statistical test for the hypothesis that a single correlation coefficient is zero can be easily carried out for $n > 2$. This allows the implementation of naive learning procedures that consider separately every edge of the graph overcoming the *large p and small n* problem. In a similar vein to the relevance network approach see also the ARACNE algorithm by Margolin et al. (2006).

More recently, other families of graphical models have been used to describe biomolecular networks (see Friedman, 2004) and among these, an important role is played by Gaussian graphical models where missing edges correspond to zero partial correlations and, therefore, to conditional independence relationships. In these models an edge between two genes represents a direct association and, more generally, a path connecting two genes represents an undirect association mediated by other genes in the path (see Jones and West, 2005). The reason why concentration graphs are more adequate than covariance graphs to describe gene networks is that, even though two genes may present a non-zero correlation because they belong to a common biological pathway, they should not be joined by an edge when they influence each other only indirectly through other observed genes that act as confounders.

The Pearson correlation is a marginal measure of association between two genes, regardless of other genes in the network. On the other hand, partial correlation is a measure of association between two genes that keeps into account all the remaining observed genes. Consequently, partial correlations cannot be computed by only looking at bivariate marginal distributions but require the full joint distribution of genes, and this is problematic when n is small. More formally, the network structure is derived from the zero pattern of the concentration matrix $K = \Sigma^{-1}$ whose maximum likelihood estimate is $\widehat{K} = S^{-1}$ which requires that S has full rank and this holds, with probability one, if and only if $n > p$ (Dykstra, 1970). Furthermore, the statistical properties of procedures for fitting and testing partial correlations depend on $n - p$ and, as pointed out for instance by Yang and Berger (1994) and Dempster (1969), the estimators based on scalar multiples of S tend to distort the Eigenstructure of the true covariance matrix, unless $n \gg p$.

Several solutions have been proposed in the literature to carry out structure learning of biomolecular networks by means of concentration graphs; see Jones et al. (2005) and Shäfer and Strimmer (2005c) for a review. A popular approach is based on *limited-order partial correlations*, that is q-order partial correlations with $q < (n-2)$. Procedures based on limited-order partial correlations have been applied, among others, by de la Fuente et al. (2004), Magwene and Kim (2004), Wille et al. (2004), Wille and Bühlmann (2006) and are also implemented in the statistical software MIM (Edwards, 2000). The key point here is that if a set of $q+2$ genes such that $(q+2) < n$ is considered, then a test for the hypothesis of a zero q-order partial correlation can be carried out with standard techniques such as those described in Section 2. Consequently, it seems somehow sensible to replace full-order partial correlations with lower-order partial correlations so as to obtain a graph that can be regarded as an approximation of the entire concentration graph G. The procedures proposed in the literature for learning such an approximating graph are based on the application of the following rule to every distinct pair of vertices $i, j \in V$:

> Test the hypotheses $\rho_{ij.Q} = 0$ for every $Q \subseteq V \backslash \{i, j\}$ such that $|Q| = q$. Then, i and j are joined by an edge if and only if all of such hypotheses of zero q-order partial correlations are rejected.

In principle, q-order partial correlations can be computed for any $q < (n-2)$; however, in practice, testing $\binom{p-2}{q}$ partial correlations for each of the $p \times (p-1)/2$ pairs of genes is computationally intensive unless q is small and, to our knowledge, the above procedure has only be applied for $q \leq 3$. For instance, Wille and Bühlmann (2006) proposed a modified version of the above procedure that considers all q-order partial correlations for $q \leq 1$. We remark that this learning procedure presents two main drawbacks. Firstly, as shown in the next section, the usefulness of q-order partial correlations increases with q, so that a procedure that can be applied for larger values of q is called for. More seriously, however, an edge is added to the graph if all of $\binom{p-2}{q}$ null hypotheses are rejected. The statistical tests are performed separately so that the well-known problems deriving from the sequential application of several tests may occur. In particular, the probability that at least one hypothesis of zero q-order partial correlation is wrongly non-rejected increases with the number of performed tests and, consequently, if the value of $\binom{p-2}{q}$ is large then one should expect that most, or even all, of the edges are removed.

In the next section we provide a formal definition of the graph associated with q-order partial correlations that we call the *q-order partial correlation graph* of X_V, *q-partial graph* hereafter, denoted by $G^{(q)} = (V, E^{(q)})$, and derive some of its properties. In this way we generalize the results of Wille and Bühlmann (2006) given for $q = 1$ to an arbitrary value of q. In particular, it is easy to check that, under the assumption of faithfulness, it holds that $G \subseteq G^{(q)}$, and consequently that P_V is undirected Markov with respect to $G^{(q)}$. This means that every pair of vertices separated in $G^{(q)}$ corresponds to a conditional independence relation between the two corresponding variables and, more specifically, every missing edge corresponds to a pairwise conditional independence. In practice, however, the usefulness of $G^{(q)}$ depends on its closeness to G, that is, on the number of edges that are present in $G^{(q)}$ but are missing in G, and we will formally address this point.

Even though the q-partial graph $G^{(q)}$ of X_V may provide a good approximation to the concentration graph G, our standpoint is that the real object of interest is the concentration graph and that the q-partial graph is useful as an intermediate step of the analysis. In fact, if the dimension of the largest clique of $G^{(q)}$ is smaller than the sample size, then the corresponding graphical model, as well as all its submodels, can be fitted and, consequently, it is possible to apply traditional search procedures to learn the concentration graph by using the fitted q-partial graph as a starting point. In this perspective, in Section 5 we propose a novel procedure to learn q-partial graphs from data. This is based on limited-order partial correlations but can be used with larger values of q and, furthermore, it does not suffer of the problems deriving from multiple testing. Since the selected graph is the starting point for further investigation, our procedure is designed to be conservative, that is, it aims at keeping the number of wrongly removed edges small and, consequently, the probability of breaking the Markov condition of P_V low. It follows that the selected graph may still contain edges that should be removed. However, if the underlying concentration graph is sparse the procedure will remove a large number of edges leading to a great simplification of the learning problem. Furthermore, as shown by examples carried out on both simulated and real data, the resulting graph is manageable with standard techniques. We remark that our procedure neither imposes any constraints to induce a dimensionality reduction nor makes any assumption of sparseness of the graph. However, the usefulness of the proposed procedure does depend on the sparseness of G. It provides an indication whether the underlying concentration graph is sparse and, in this case, it will lead to a great simplification of the structure learning problem.

4. q-Partial Graphs

The use of limited-order partial correlations in structure learning is appealing when either $p > n$ or the available data are too scarce to produce reliable estimates of the concentration matrix. However, the object of interest is the concentration graph G of X_V and it is not clear which graph can be learnt by using q-order partial correlations, and what is the connection between such a graph and G. In this section we formally approach this question: firstly, we introduce the *q-partial graph* of X_V, that is a graph in which missing edges correspond to zero q-order partial correlations. Secondly, we characterize the class of graphs for which concentration graphs and q-partial graphs coincide and, in particular, we show how information on the concentration graph of X_V can be extracted from the q-partial graph of X_V. The theory here developed relies on the graph theory described in Appendix A and more specifically on the concepts of the outer connectivity of two vertices i and j, $d(i,j|G)$, the outer connectivity of the edges of G, $d(E|G)$, the outer connectivity of the missing edges of G, $d(\bar{E}|G)$, and finally, the outer connectivity of G, $d(G)$.

The concentration graph of X_V is associated with the probability distribution of X_V and we define the q-partial graph of X_V as a graph associated with the set of all marginal distributions of X_V of dimension $(q+2)$.

Definition 1 *For a random vector X_V and an integer $0 \leq q \leq (p-2)$ we define the* q-partial graph *of X_V, denoted by $G^{(q)} = (V, E^{(q)})$, as the undirected graph where $(i,j) \in \bar{E}^{(q)}$ if and only if there exists a set $U \subseteq V$ with $|U| \leq q$ and $i, j \notin U$ such that $X_i \perp\!\!\!\perp X_j | X_U$ holds in P_V.*

We first observe that $G^{(p-2)}$ and $G^{(0)}$ are the concentration graph and the covariance graph of X_V respectively, whereas $G^{(1)}$ is the 0-1 conditional independence graph introduced by Wille and Bühlmann (2006, Definition 3). It is also easy to show that that $G^{(q)}$ is larger than G, $G \subseteq G^{(q)}$, that is every edge in G is also an edge in $G^{(q)}$. This follows from the fact that if $(i,j) \in E$ then the faithfulness of X_V to G implies that there is no set $U \subseteq V$ with $i, j \notin U$ such that $X_i \perp\!\!\!\perp X_j | X_U$, and therefore it holds that $(i,j) \in E^{(q)}$; see also Wille and Bühlmann (2006).

The relation $G \subseteq G^{(q)}$ implies that X_V is Markov with respect to $G^{(q)}$. However, the usefulness of $G^{(q)}$ as a surrogate of G depends on the closeness of the two graphs. Every edge of G is present in $G^{(q)}$ and in the following proposition we characterize the missing edges of G that are also missing in $G^{(q)}$.

Proposition 1 *Let $G = (V,E)$ and $G^{(q)} = (V,E^{(q)})$ be the concentration and the q-partial graph of X_V respectively. If $(i,j) \in \bar{E}$ then $(i,j) \in \bar{E}^{(q)}$ if and only if $d(i,j|G) \leq q$.*

Proof Sufficiency. If $d(i,j|G) \leq q$ then there exists a nontrivial minimal $\{i,j\}$-separator $S \in \mathcal{S}_{(i,j|G)}$ such that $|S| \leq q$. By the Markov property, it holds that $X_i \perp\!\!\!\perp X_j | X_S$ so that $(i,j) \in \bar{E}^{(q)}$ by definition of q-partial graph. Necessity. If $(i,j) \in \bar{E}^{(q)}$ then there exists a set $U \subseteq V$ with $|U| \leq q$ and $i, j \notin U$ such that $X_i \perp\!\!\!\perp X_j | X_U$. By the faithfulness assumption, such a conditional independence relation can be also read off the graph G through the Markov property. In other worlds, U is a nontrivial $\{i,j\}$-separator in G so that there exists a subset $S \subseteq U$ such that $S \in \mathcal{S}_{(i,j|G)}$ and, consequently, $d(i,j|G) \leq |S| \leq q$. ■

The result stated in the above proposition is very intuitive. A missing edge in G is missing also in $G^{(q)}$ if and only if the outer connectivity of the corresponding vertices is smaller or equal to q or, that is, if and only if there exists a marginal distribution of X_V of dimension $(q+2)$ in which the

corresponding variables are conditionally independent. If this relation is satisfied for all the missing edges of G then the q-partial graph and the concentration graph are identical.

Proposition 2 *Let $G = (V,E)$ and $G^{(q)} = (V,E^{(q)})$ be the concentration and the q-partial graph of X_V respectively. Then $G = G^{(q)}$ if and only if $d(\bar{E}|G) \leq q$.*

Proof We have already shown that the inclusion relation $G \subseteq G^{(q)}$ is always satisfied. Consequently, we have only to show that $G \supseteq G^{(q)}$ if and only if $d(\bar{E}|G) \leq q$. The condition $G \supseteq G^{(q)}$ is satisfied if and only if $(i,j) \in \bar{E}$ implies $(i,j) \in \bar{E}^{(q)}$, and in the following we consider the latter formulation of the condition. Sufficiency. By Equation (9) in the Appendix, $d(\bar{E}|G) \leq q$ implies $d(i,j|G) \leq q$ for all $(i,j) \in \bar{E}$ and, by Proposition 1, this implies that $(i,j) \in \bar{E}^{(q)}$ for every $(i,j) \in \bar{E}$. Necessity. By Proposition 1, if $(i,j) \in \bar{E}^{(q)}$ for all $(i,j) \in \bar{E}$, then $d(i,j|G) \leq q$ for all $(i,j) \in \bar{E}$, and it follows from (9) that $d(\bar{E}|G) \leq q$. ■

The result of Proposition 2 clarifies that the concentration graph G and the q-partial graph $G^{(q)}$ of X_V coincide when $d(\bar{E}|G)$ is not greater than q so that a natural question concerns the connection between the sparseness of G and the value of $d(\bar{E}|G)$. This is discussed at the end of Appendix A where it is shown that there is no direct connection between the degree of sparseness of G and outer degree of missing edges. In particular it is possible to find examples in which the condition of Proposition 2 is satisfied for a graph G' but is not satisfied for a sparser graph $G \subset G'$. Note also that the condition of Proposition 2 is always satisfied when G is the complete graph. The point here is that sparseness is useful as long as it implies small separators for non-adjacent vertices, however it is not difficult to draw a very sparse graph in which two non-adjacent vertices have high value of outer connectivity.

It is somehow intuitive that larger values of q should be preferred and, in fact, an immediate consequence of Proposition 1 is the following relation of inclusion between partial graphs of different order.

Corollary 3 *Let $G^{(q)} = (V,E^{(q)})$ and $G^{(r)} = (V,E^{(r)})$ be the q-partial and the r-partial graph of X_V respectively. If $r \leq q$ then $G^{(q)} \subseteq G^{(r)}$.*

Proof We show that if $r \leq q$ and $(i,j) \in \bar{E}^{(r)}$ then $(i,j) \in \bar{E}^{(q)}$. From the definition of outer connectivity (see Appendix) $(i,j) \in \bar{E}^{(r)}$ implies $d(i,j|G^{(r)}) \leq r$. Since $r \leq q$, $d(i,j|G^{(r)}) \leq q$ and therefore by Proposition 1 $(i,j) \in \bar{E}^{(q)}$. ■

The results provided so far allow to understand in which cases q-partial graphs may be useful. They give a set of necessary and sufficient conditions, however such conditions are stated with respect to G, which is unknown, and therefore their usefulness is limited in practice to situations in which background knowledge on the problem under analysis may provide information on the structure of G. Also $G^{(q)}$ is typically unknown but it can be learnt from data and in the rest of this section we show how information on the structure of G can be extracted from $G^{(q)}$.

The fact that $G^{(q)}$ is larger than G implies that if an edge is missing in $G^{(q)}$ then it is also missing in G and the next theorem provides a sufficient condition to check whether an edge that is present in $G^{(q)}$ is also present in G.

Theorem 4 *Let $G = (V,E)$ and $G^{(q)} = (V,E^{(q)})$ be the concentration and the q-partial graph of X_V respectively. If $(i,j) \in E^{(q)}$ then a sufficient condition for the relation $(i,j) \in E$ to hold is $d(i,j|G^{(q)}) \leq q$.*

Proof Assume $(i,j) \in E^{(q)}$ and $d(i,j|G^{(q)}) \leq q$. As mentioned earlier in the paper, from the faithfulness of P_V it follows $G \subseteq G^{(q)}$ and thus by Equation (13) in Theorem 6 $d(i,j|G) \leq d(i,j|G^{(q)}) \leq q$. By Proposition 1, $d(i,j|G) \leq q$ implies that if $(i,j) \in \bar{E}$ then $(i,j) \in \bar{E}^{(q)}$ which would contradict the initial assumption and therefore $(i,j) \in E$. ■

Note that the condition of Theorem 4 can be checked on $G^{(q)}$, and an immediate consequence of Theorem 4 is the following corollary that provides a sufficient condition for checking the identity $G = G^{(q)}$ directly from $G^{(q)}$.

Corollary 5 *Let $G = (V,E)$ and $G^{(q)} = (V,E^{(q)})$ be the concentration and the q-partial graph of X_V respectively. A sufficient condition for the relation $G = G^{(q)}$ to hold is that $d(E^{(q)}|G^{(q)}) \leq q$.*

Assuming that $G^{(q)}$ is known, then Corollary 5 gives a condition to check the identity $G = G^{(q)}$. In the case one cannot conclude that G is equal to $G^{(q)}$ then Theorem 4 can be applied to decide which edges of $G^{(q)}$ belong also to G and which edges of $G^{(q)}$ may be spurious. Theorem 4 and Corollary 5 should be compared with Propositions 1 and 2. The former give weaker results but are of more practical use because if an estimate $\widehat{G}^{(q)} = (V,\widehat{E}^{(q)})$ of $G^{(q)}$ is available, then one can estimate $d(E^{(q)}|G^{(q)})$ with $d(\widehat{E}^{(q)}|\widehat{G}^{(q)})$ and $d(i,j|G^{(q)})$ with $d(i,j|\widehat{G}^{(q)})$.

The computation of the outer connectivity of two vertices is known to be a NP-hard problem. Nevertheless several algorithms are available to derive both upper and lower bounds to this number (Rosenberg and Heath, 2001) and, since all the results stated in this section involve inequalities, then such upper and lower bounds may be sufficient to check the required conditions. Note also that equations (7), (10), (11) and (12) in Appendix A are instances of easily computable upper bounds.

We close this section by noticing that the outer connectivity of edges and the outer connectivity of missing edges play a different role with respect to $G^{(q)}$. The quantities that determine the "closeness" of $G^{(q)}$ to G are $d(i,j|G)$ for $(i,j) \in \bar{E}$. Indeed, both the value of $d(E|G)$ and of $d(E^{(q)}|G^{(q)})$ are irrelevant here, and a concentration graph can coincide with a q-partial graph even if its edges have a very high maximal degree of outer connectivity; recall that $d(E|G) \leq d(E^{(q)}|G^{(q)})$ by (14). On the other hand, the values of $d(i,j|G^{(q)})$ for $(i,j) \in E^{(q)}$ are important for the practical usefulness of q-partial graphs: the larger the number of edges of $(i,j) \in E^{(q)}$ with $d(i,j|G^{(q)}) \leq q$ the larger is the amount of information that $G^{(q)}$ provides with respect to G. Note also that, unlike $d(\bar{E}|G)$, the value of $d(E|G)$ is related with the sparseness of G (see Theorem 6 in Appendix A).

5. The qp-Procedure

We now introduce a novel procedure to learn the q-partial graph $G^{(q)}$ of X_V, that we name the *qp-procedure*. This is based on limited-order partial correlations and, more specifically, on a quantity that we call the *non-rejection rate*. The latter is a probability associated with every pair of variables X_i and X_j, and turns out to be useful in discriminating between present and missing edges in $G^{(q)}$. The qp-procedure firstly estimates the value of all the $p \times (p-1)/2$ non-rejection rates and then a graph $\widehat{G}^{(q)}$ is constructed by removing from the complete graph all the edges corresponding to the pairs of variables whose fitted value of the non-rejection rate is above a given threshold. In Section 5.1 we formally introduce the non-rejection rate. In Section 5.2 we describe the procedure in more detail by means of two examples and, finally, in Section 5.3 we provide istances of the application of the procedure on both simulated and real data.

5.1 The Non-Rejection Rate

For a pair of vertices $i, j \in V$, with $i \neq j$, and an integer $q \leq (p-2)$ let $\mathcal{Q}_{ij}$ be the set made up of all the subsets Q of $V\backslash\{i,j\}$ such that $|Q| = q$; thus the cardinality of $\mathcal{Q}_{ij}$ is $m = \binom{p-2}{q}$. Furthermore, let T_{ij}^q be the random variable resulting of the two stage experiment in which firstly an element Q is sampled from $\mathcal{Q}_{ij}$ according to a (discrete) uniform distribution and then the data $X^{(n)}$ are used to test the null hypothesis $H_0 : \rho_{ij.Q} = 0$ against the alternative hypothesis $H_A : \rho_{ij.Q} \neq 0$. The random variable T_{ij}^q takes value 0 if the above null hypothesis is rejected and 1 otherwise. It follows that T_{ij}^q has a Bernoulli distribution and the non-rejection rate is defined as follows.

Definition 2 *For a random sample $X^{(n)}$ from X_V the* non-rejection rate *for the variables X_i and X_j with $i, j \in V$, $i \neq j$, is given by*

$$E\left[T_{ij}^q\right] = Pr(T_{ij}^q = 1).$$

In order for the non-rejection rate to be unambiguously defined, we have to specify the statistical test we use. In the following, we always take $q < (n-2)$ and apply the t test for zero regression coefficient as described at the end of Section 2.

If $Pr(T_{ij}^q = 1|Q)$ denotes the probability that H_0 is not rejected for a given set $Q \in \mathcal{Q}_{ij}$, then

$$Pr(T_{ij}^q = 1|Q) = \begin{cases} (1-\alpha) & \text{if } Q \text{ separates } i \text{ and } j \text{ in } G; \\ \beta_{ij.Q} & \text{otherwise;} \end{cases} \tag{2}$$

where α and $\beta_{ij.Q}$ are the probability of the first and the second type error of the test respectively. The value of α can be arbitrarily specified and we take it constant over all pairs of vertices and all elements of $\mathcal{Q}_{ij}$. The value of $\beta_{ij.Q}$ is usually unknown because it depends on the true value of the parameters. Nevertheless, the effectiveness of the *qp*-procedure depends on the statistical properties of the power function of the test, and for this reason we use a UMPU test; in particular, recall that $\beta_{ij.Q} \leq (1-\alpha)$.

The non-rejection rate for X_i and X_j can thus be computed by using the law of total probability as follows

$$\begin{aligned} Pr(T_{ij}^q = 1) &= \sum_{Q \in \mathcal{Q}_{ij}} Pr(T_{ij}^q = 1|Q) Pr(Q) \\ &= \frac{1}{m} \sum_{Q \in \mathcal{Q}_{ij}} Pr(T_{ij}^q = 1|Q). \end{aligned} \tag{3}$$

An element Q of $\mathcal{Q}_{ij}$ can either separate i and j in G or not separate them. We denote by $1_{ij}(Q)$ the indicator function that is 1 if $Q \in \mathcal{Q}_{ij}$ separates i and j in G and 0 otherwise. Furthermore, we denote by π_{ij} the proportion of elements of $\mathcal{Q}_{ij}$ which separate i and j in G so that

$$\pi_{ij} = \frac{1}{m} \sum_{Q \in \mathcal{Q}_{ij}} 1_{ij}(Q) \qquad \text{and} \qquad (1-\pi_{ij}) = \frac{1}{m} \sum_{Q \in \mathcal{Q}_{ij}} \{1 - 1_{ij}(Q)\}.$$

The second type error is defined only for the sets $Q \in \mathcal{Q}_{ij}$ such that $1_{ij}(Q) = 0$ and we define the average value of the second type error for the pair i and j over $\mathcal{Q}_{ij}$ as

$$\beta_{ij} := \frac{1}{m(1-\pi_{ij})} \sum_{Q \in \mathcal{Q}_{ij}} \beta_{ij.Q} \{1 - 1_{ij}(Q)\} \tag{4}$$

with $\beta_{ij} = 0$ if $\pi_{ij} = 1$.

We can now turn to the computation of the non-rejection rate in (3). By (2) it holds that

$$Pr(T_{ij}^q = 1) \quad = \quad \frac{1}{m} \sum_{Q \in \mathcal{Q}_{ij}} [\beta_{ij.Q}\{1 - 1_{ij}(Q)\} + (1-\alpha)1_{ij}(Q)]$$

and, by (4),

$$Pr(T_{ij}^q = 1) \quad = \quad \frac{1}{m} \left\{\beta_{ij}\, m\,(1-\pi_{ij}) + (1-\alpha)\, m\, \pi_{ij}\right\}$$

so that we obtain the final form

$$Pr(T_{ij}^q = 1) \quad = \quad \beta_{ij}\,(1-\pi_{ij}) + (1-\alpha)\,\pi_{ij}. \tag{5}$$

Equation (5) can be used to clarify the usefulness of the non-rejection rate in the statistical learning of $G^{(q)}$.

Consider first the situation in which the vertices i and j are joined by an edge in $G^{(q)} = (V, E^{(q)})$, that is, $(i, j) \in E^{(q)}$. In this case no element of $\mathcal{Q}_{ij}$ separates i and j in $G = (V, E)$ so that $\pi_{ij} = 0$ and $Pr(T_{ij}^q = 1) = \beta_{ij}$ where β_{ij} is the mean value of $\beta_{ij.Q}$ for $Q \in \mathcal{Q}_{ij}$. Since for every $Q \in \mathcal{Q}_{ij}$, $\beta_{ij.Q}$ belongs to the interval $(0, 1-\alpha)$ then also $0 \leq \beta_{ij} \leq (1-\alpha)$ but, more interestingly, β_{ij} is close to the boundary $(1-\alpha)$ only if the distribution of the $\beta_{ij.Q}$ for $Q \in \mathcal{Q}_{ij}$ is highly asymmetric on the interval $(0, 1-\alpha)$ with most of the values very close to the boundary $(1-\alpha)$; in other words, if the second type error $\beta_{ij.Q}$ is uniformly very high over $\mathcal{Q}_{ij}$. It follows that a value of $Pr(T_{ij}^q = 1)$ "close" to $1-\alpha$ means either that $(i, j) \in \bar{E}^{(q)}$ or that $(i, j) \in E^{(q)}$ but that such an edge is very difficult to identify on the basis of q-order partial correlations and of the available data. The qp-procedure aims at identifying some of, but not necessarily all the, missing edges of $G^{(q)}$ by keeping the number of wrongly removed edges low and thus trying to avoid breaking the Markov condition of the underlying probability distribution. In this perspective, it makes sense to remove the edges with $Pr(T_{ij}^q = 1)$ above a given threshold β^*. By keeping the value β^* very close to the boundary $(1-\alpha)$ the procedure will wrongly remove a present edge only when data strongly support its removal.

We now turn to the situation in which $(i, j) \in \bar{E}^{(q)}$. In this case $Pr(T_{ij}^q = 1)$ belongs to the interval $(\beta_{ij}, 1-\alpha)$ and, although it can take any value in such interval, it is important to notice that it will be closer to the boundary $(1-\alpha)$ for larger values of π_{ij}.

A missing edge is identified by the qp-procedure if its non-rejection rate is above β^*; however, the procedure does not aim at removing all missing edges and it is only important that the value of the non-rejection rate is above β^* for a large number of missing edges. A sufficient condition for this to happen is that (i) $G^{(q)}$ has a large number of missing edges and (ii) for a large number of such missing edges, the value of π_{ij} is high. Condition (i) can obviously be satisfied only if G is sparse but also the value of q plays a fundamental role because as shown in Corollary 3 a larger value of q increases the sparseness of the q-partial graph and, consequently, the values of the π_{ij}'s. On the other hand, a present edge is correctly identified by the procedure if the value of β_{ij} is below β^* and, in turn, this depends on the second type errors $\beta_{ij.Q}$ for $Q \in \mathcal{Q}_{ij}$. The statistical properties of inferential procedures involving q-order partial correlations depend on $n-q$. In the context we are considering, the sample size n cannot be easily increased but a way to make $n-q$ larger is to

decrease the value of q. We can conclude that a larger value of q allows us to identify a larger number of missing edges but also decreases the power of the statistical tests, making present edges more difficult to identify; see Section 5.3.

An interesting observation is that, in general, the effectiveness of inferential procedures in multivariate problems depends on the quantity $n-p$ being sufficiently large. The effectiveness of procedures based on the non-rejection rate also depends on $n-p$ but split such quantity into two parts:

$$(n-p) = (n-q) - (p-q) \tag{6}$$

the term $n-q$ has to be sufficiently large to guarantee the required power of statistical tests and $(p-q)$ has to be sufficiently small to guarantee the required sparseness of $G^{(q)}$, and there is a trade-off between these two requirements. However, for problems in which G is very sparse, the q-partial graph $G^{(q)}$ can be sufficiently sparse also for small values of q and, in turn, this leads to satisfactory values of $(n-q)$ even in the case $n-p$ is very small or even negative.

5.2 Description Of The Procedure

The qp-procedure is made up of five steps:

1. Specify a value $q < (n-2)$;
2. estimate the non-rejection rate $E[T_{ij}^q]$ for every pair of variables;
3. on the basis of the estimated non-rejection rates, decide whether to go
   - 3.1 on to step 4
   - 3.2 back to step 1 and modify the value of q (if possible);
4. specify a threshold β^*;
5. return a graph $\widehat{G}^{(q)}$ obtained by removing from the complete graph all the edges whose estimated non-rejection rate is greater than β^*.

We now describe every step in detail by means of an example. Figure 1 gives the image of a partial correlation matrix for 164 variables. It is made up of 20 diagonal blocks of size 12×12 and there is a 4×4 submatrix overlap between every two adjacent blocks. The associated concentration graph, that we denote by G, has 1206 edges corresponding to 9% of all possible edges. We used this matrix as a concentration matrix to generate $n = 40$ independent observations from a multivariate normal distribution with zero mean.

It is straightforward to check, by using the results of Section 4, that $G^{(20)} = G$ whereas $G^{(3)}$ is the complete graph and in this example we compare the qp-procedure for both $q = 3$ and $q = 20$.

We have thus set the value of q, and the second step of the procedure requires the estimation of the non-rejection rates. In principle, an unbiased estimate of the non-rejection rate for a pair of variables X_i and X_j can be easily obtained by first testing the hypothesis $\rho_{ij.Q} = 0$ for all $Q \in \mathcal{Q}_{ij}$, on the basis of the available data $X^{(n)}$, and then by computing the proportion of such tests in which the null hypothesis is not rejected. In practice, however, this requires the computation of $\binom{p-2}{q}$ statistical tests for every one of the $p \times (p-1)/2$ pairs of variables and may be computationally unfeasible. In order to overcome this difficulty we use a Monte Carlo method in which, for every pair X_i and

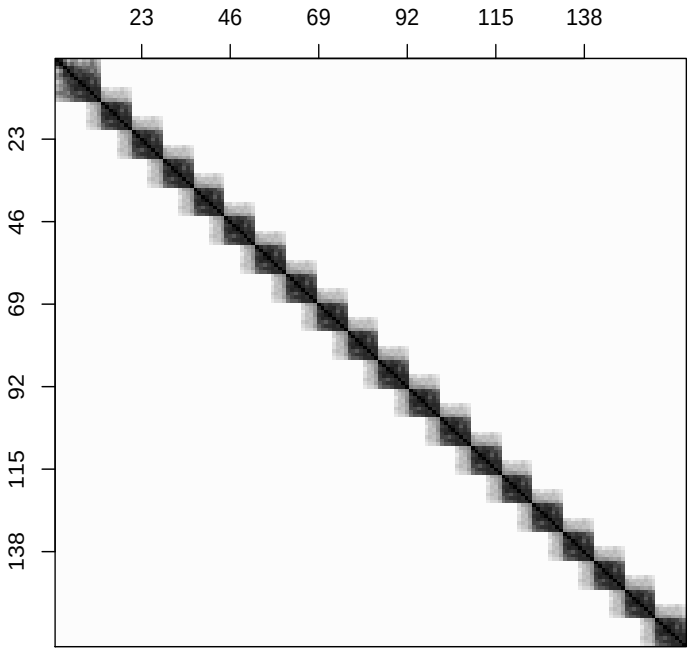


Figure 1: Image of a partial correlation matrix for 164 variables. Every entry of the matrix is represented as a gray-scaled point between zero (white points) and ± 1 (black points).

X_j, the required statistical tests are computed for a large number of sets randomly sampled from Q_{ij} according to a uniform distribution. In the example we are considering, the non-rejection rate is estimated by sampling 500 elements from Q_{ij}, for all of the 13366 pairs of variables. For the case $q = 20$, Figure 2 gives the boxplots of the estimates of the non-rejection rate for the present and missing edges of $G^{(20)}$. This picture provides a clear example of the different behavior of the non-rejection rate for present and missing edges and it is also worth recalling that that there is a large difference in the number of present and missing edges: 1206 versus 12160.

The third step involves a decision on the adequateness of the chosen value of q and possibly on the effectiveness of the non-rejection rate for the considered problem. The main tools used here are two plots that we call the *qp-hist plot* and the *qp-clique plot* respectively. The first is the histogram of estimated values of the $p \times (p-1)/2$ non-rejection rates, see Figure 3. The latter is more complex, see Figure 4, and provides information on the graphs potentially selected by specifying different values of the threshold β^*. More specifically, every circle in the plot corresponds to a graph and has three values associated with it: the threshold value used to construct the graph (horizontal axis); the number of vertices of the largest clique of the graph (vertical axis); the percentage of present edges in the graph (number inside the plot, beside the circle). Furthermore, adjacent circles are joined by a line and the dotted horizontal line corresponds to the sample size n. To understand the usefulness of this plot one has to recall that in Gaussian graphical models the real dimension of the problem is given by the size of the largest clique of the concentration graph. The *qp-clique* plot gives the dimension of the largest cliques of the graphs associated with different values of the threshold thus providing a way to assess the effectiveness of the non-rejection rate as a tool for

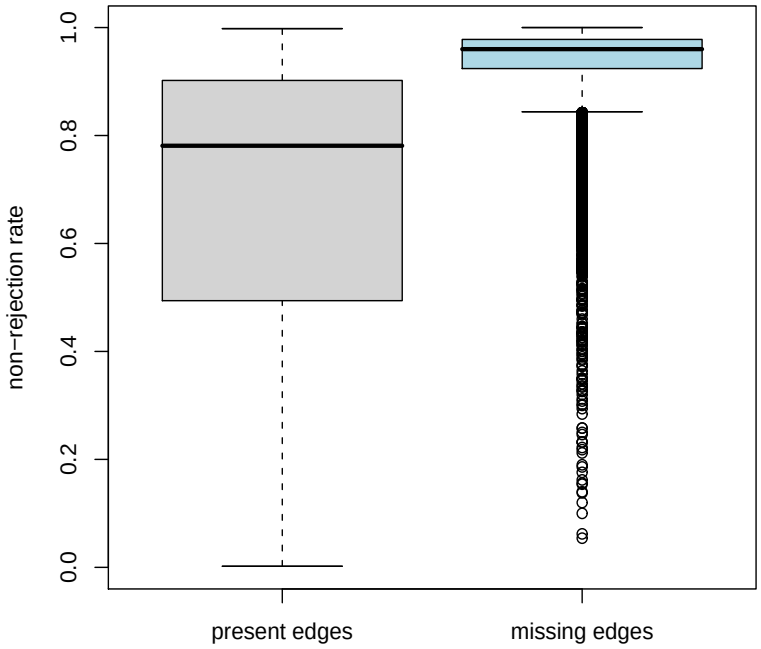


Figure 2: Boxplots of the estimated values of the non-rejection rate for the 1206 present edges and for the 12 160 missing edges of $G = G^{(20)}$.

dimensionality reduction. In particular, every circle below the dotted horizontal line corresponds to a model whose dimension is smaller than the sample size, and therefore that can be dealt with standard techniques.

We now analyze these two types of plots for the example considered. Both histograms in Figure 3 are asymmetric but the first histogram, for $q = 3$, is less asymmetric with a heavier left tail, and this is a first indication that for the case $q = 3$ the non-rejection rate may be of limited usefulness because we will not be able to remove many edges that are really missing without removing many others that should not be removed.

However, a more clear difference between the two cases can be derived from Figure 4. The dimension of models grows almost linearly for $q = 3$ whereas, for the case $q = 20$, it grows exponentially, increasing drastically only for threshold values larger than 0.975. For instance, for $q = 20$, a threshold equal to 0.9 would lead to the removal of 77% of edges, returning a graph with 23% of edges left. The same threshold for $q = 3$ would only lead to the removal of 43% of edges, returning a graph with 57% of edges left. Furthermore, the largest threshold that produces a graph for which the dimension of the largest clique is smaller than the sample size is 0.5 for $q = 3$ and 0.975 for $q = 20$. The qp-clique plot provides an indication of the sparseness of the q-marginal graph as well as of the usefulness of the non-rejection rate in statistical learning. As explained in Section 5.1, in the qp-procedure the threshold β^* has to be a value very close to one, and in the example for $q = 3$ any value close to one would lead to an insufficient dimensionality reduction. In this case, one should go back to the first step and, if possible, to increase the value of q. If the value of q cannot be increased, then one can conclude that the use of q-partial graphs is not appropriate for the problem

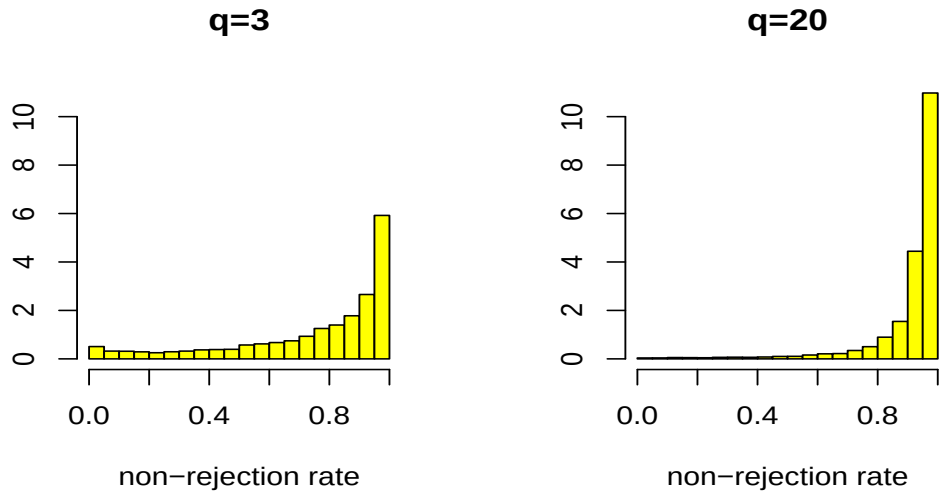


Figure 3: Histograms of the estimated values of the non-rejection rates.

under analysis. For the case $q = 20$ we can set $\beta^* = 0.975$ selecting in this way a graph $\widehat{G}^{(20)}$ with 9751 out of 13366 possible edges and whose largest clique has size 32. Figure 5 gives the adjacency matrix of $\widehat{G}^{(20)}$ and shows that, although this is clearly an overparameterized model, a substantial dimensionality reduction has been achieved while preserving the block diagonal structure of $G^{(20)}$. Indeed, only 34 of the 1206 present edges are wrongly removed corresponding to an error of 2.8%.

5.3 Experimental Results

In this section we use simulated data to describe the behavior of the non-rejection rate for different values of q, n and different degrees of sparsity of the concentration graph. Furthermore, we present the application of the procedure to a real data set.

For the simulations, we set $p = 150$ and constructed two graphs, $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ which have been randomly generated by imposing that every vertex has at most 5 and 20 adjacencies respectively. In this way, it follows from the results of Section 4 that for all $q \geq 5$ it holds that $G_1^{(q)} = G_1$ whereas for all $q \geq 20$ it holds that $G_2^{(q)} = G_2$. The graph G_1 has 375 edges whereas G_2 has 1499 edges that correspond to 3.36% and 13.4% of the 11175 possible edges respectively. Successively, an inverse covariance matrix with the zero pattern induced by G_1 has been randomly constructed (see Roverato, 2002) and then two samples, of size 20 and 150 respectively, have been randomly generated from a normal distribution with zero mean and the given covariance matrix. The same procedure was used to generate two random samples of size 20 and 50 for G_2.

We first consider G_1 and $n = 20$ and independently apply the qp-procedure with six different values of q, ranging from 1 to 17; recall that the latter is the maximum possible value of q when $n = 20$. Figure 6 shows the six qp-hist plots, which are displayed for increasing values of $(n-q)$, that is, for decreasing values of q, because the power of the statistical test we use increases with $(n-q)$. For $q = 17$ the tests have very low power and this results in a qp-hist plot where the

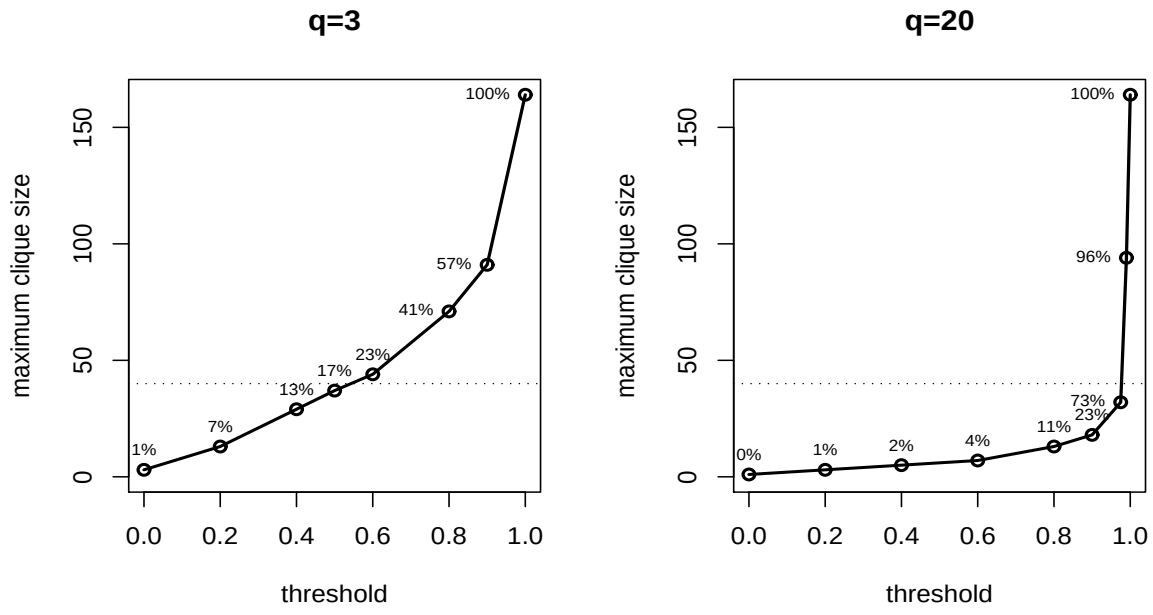


Figure 4: Plots giving the largest clique sizes of the graphs selected with different threshold values. For every graph the percentage of present edges is given and the dotted horizontal line is the sample size n.

non-rejection rate is very high for all pairs of variables. As the value of $(n-q)$ increases the qp-hist plots show heavier left tails while maintaining a strong negative asymmetric form. As Figure 7 clarifies, this happens because the distributions of the non-rejection rate for present and missing edges become more and more separated as $(n-q)$ increases. We remark that the present and missing edges in Figure 7 are relative to G_1 and not to $G_1^{(q)}$.

A numerical description of the results of these simulations is given in Tables 1 and 2. The first part of these tables gives the quantities used in the construction of the qp-clique plots: some threshold values (thr.) and, for every threshold, the size of the largest clique (l.c.) and the percentage of present edges (% pre.) of the corresponding graph. The remaining columns provide measures of goodness of the graph associated with each threshold. More specifically, "err." gives the number of wrongly removed edges, "% err." is the percentage of wrongly removed edges with respect to all the removed edges and, finally, "% imp." is the rate of improvement with respect to the random removal of edges: a learning procedure based on the random removal of edges would lead to a relative error whose expected value is the proportion of edges in the graph, that is 3.36% for G_1, and the improvement rate of a graph is the relative difference between "% err." and the proportion of present edges in the concentration graph. We remark that the last three columns of these tables are not available in real applications where the concentration graph is unknown.

Figures 6 and 7 seem to indicate that the value of q should be chosen as low as possible; nevertheless, as described in Section 5.1 the value of q should not be chosen too small in order to

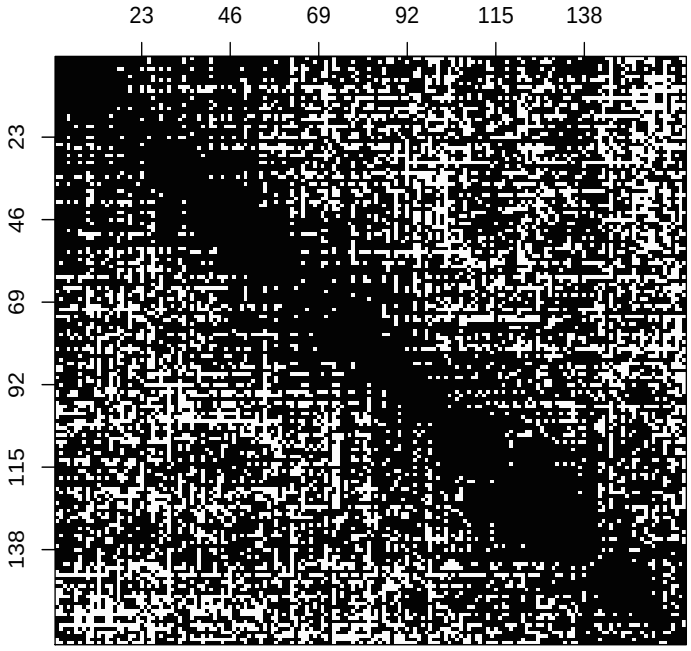


Figure 5: Adjacency matrix of the graph selected by the qp-procedure with $q = 20$ and $\beta^* = 0.975$. Black points are present edges (value 1 in the adjacency matrix) and white points missing edges (value 0 in the adjacency matrix).

guarantee an adequate sparseness of $G_1^{(q)}$. If in Tables 1 and 2 one takes, for the different values of q and $n = 20$, the largest threshold corresponding to a graph whose largest clique size is smaller than n, then the best solution is provided by $q = 10$ with a graph in which 6601 edges are missing, the largest clique has size 13 and the absolute error is 97 with a 56.21% improvement rate. However, also the case $q = 5$ provides a good solution with a graph in which 7194 edges are missing, the largest clique has size 19 and the absolute error is 103 with a 57.33% improvement rate. A value of q equal either to 5 or to 10 represents the most natural choice in the trade-off between $(n-q)$ and $(p-q)$ in (6), however we notice that, apart from $q = 17$ where the relative improvement is only 38.32%, all the other considered values of q provide satisfying solutions. This seems to suggest that the qp-procedure is not very sensitive to the choice of q. We can conclude that the qp-procedure is very effective despite the fact that we are considering an extremely challenging problem where the sample size is very small, $n = 20$, compared to the number of variables, $p = 150$. In order to show the behavior of the non-rejection rate as the sample size increases, in Figure 8 and Table 2 we provide an example in which the sample size is larger, $n = 150$, but still too low to permit the computation of sample full-order partial correlations. The boxplots in Figure 8 highlights the great effectiveness of the non-rejection rate in this case. Table 2 shows that one can either select the largest graph manageable with standard techniques, choosing in this way a graph with only 12 wrongly removed edges, or select a sparser graph; for instance, the threshold 0.60 gives a graph with 9365 out of 11 175 missing edges, absolute error 85 and a 72.94% improvement rate. It is also interesting to compare Figure 8 with the case $q = 17$ in Figures 6 and 7.

n	q	thr.	l.c.	% pre.	err.	% err.	% imp.
20	1						
		0.30	10	10.4	187	1.87	44.37
		0.60	13	14.2	177	1.85	45.00
		0.80	14	17.1	169	1.82	45.63
		0.85	14	18.5	166	1.82	45.68
		0.90	15	21.3	155	1.76	47.50
		0.95	17	27.2	136	1.67	50.18
		0.97	19	32.4	123	1.63	51.51
		0.98	19	36.9	111	1.58	53.05
		0.99	22	46.9	88	1.48	55.81
20	3						
		0.30	7	4.7	228	2.14	36.18
		0.60	9	10.1	191	1.90	43.35
		0.80	12	16.7	170	1.83	45.59
		0.85	14	19.8	156	1.74	48.15
		0.90	14	24.5	143	1.69	49.50
		0.95	17	34.2	120	1.63	51.36
		0.97	20	42.7	96	1.50	55.36
		0.98	22	50.4	79	1.43	57.49
		0.99	27	63.8	53	1.31	60.99
20	5						
		0.30	6	2.9	235	2.16	35.49
		0.60	8	6.9	195	1.87	44.13
		0.80	11	13.8	163	1.69	49.57
		0.85	12	17.3	152	1.65	50.98
		0.90	13	22.9	138	1.60	52.27
		0.95	19	35.6	103	1.43	57.33
		0.97	23	47.1	83	1.40	58.15
		0.98	28	57.0	65	1.35	59.70
		0.99	36	74.2	38	1.32	60.80

Table 1: Graph $G_1 = (V, E_1)$. Numerical description of the output of the qp-procedure applied for $n = 20$ and $q = 1, 3, 5$. The first part of the table gives the quantities used in the construction of the qp-clique plots: some threshold values (thr.) and, for every threshold, the size of the largest clique (l.c.) and the percentage of present edges (% pre.) of the corresponding graph. The last three columns give the number of wrongly removed edges (err.), the percentage of wrongly removed edges with respect to all the removed edges (% err.) and the rate of improvement with respect to the random removal of edges (% imp.).

n	q	thr.	l.c.	% pre.	err.	% err.	% imp.
20	10						
		0.30	4	0.7	313	2.82	15.94
		0.60	5	2.5	244	2.24	33.26
		0.80	7	7.6	199	1.93	42.59
		0.85	8	11.4	174	1.76	47.66
		0.90	9	19.0	149	1.65	50.93
		0.95	13	40.9	97	1.47	56.21
		0.97	25	67.2	58	1.58	52.83
		0.98	45	85.6	26	1.62	51.82
		0.99	99	98.1	6	2.82	16.06
20	15						
		0.30	2	0.1	371	3.32	1.03
		0.60	3	0.3	347	3.11	7.20
		0.80	5	1.0	303	2.74	18.36
		0.85	6	1.9	278	2.54	24.45
		0.90	6	5.5	233	2.21	34.28
		0.95	11	45.5	104	1.71	49.08
		0.97	50	94.2	10	1.53	54.29
		0.98	124	99.6	0	0.00	100.00
		0.99	150	100.0	0	0.00	100.00
20	17						
		0.30	1	0.0	375	3.36	0.00
		0.60	1	0.0	375	3.36	0.00
		0.80	1	0.0	375	3.36	0.00
		0.85	2	0.1	366	3.28	2.31
		0.90	3	0.4	339	3.05	9.23
		0.95	11	53.3	108	2.07	38.32
		0.97	89	98.7	2	1.38	58.90
		0.98	149	99.9	0	0.00	100.00
		0.99	150	100.0	0	0.00	100.00
150	17						
		0.30	6	7.0	118	1.14	66.17
		0.60	9	16.2	85	0.91	72.94
		0.80	13	29.4	60	0.76	77.32
		0.85	15	35.6	53	0.74	78.07
		0.90	17	44.3	44	0.71	78.93
		0.95	23	60.4	34	0.77	77.10
		0.97	34	70.7	30	0.92	72.72
		0.98	44	77.5	21	0.84	75.09
		0.99	62	86.3	12	0.78	76.61

Table 2: Graph $G_1 = (V, E_1)$. Numerical description of the output of the qp-procedure applied with different values of n and q. See Table 1 for a description of columns.

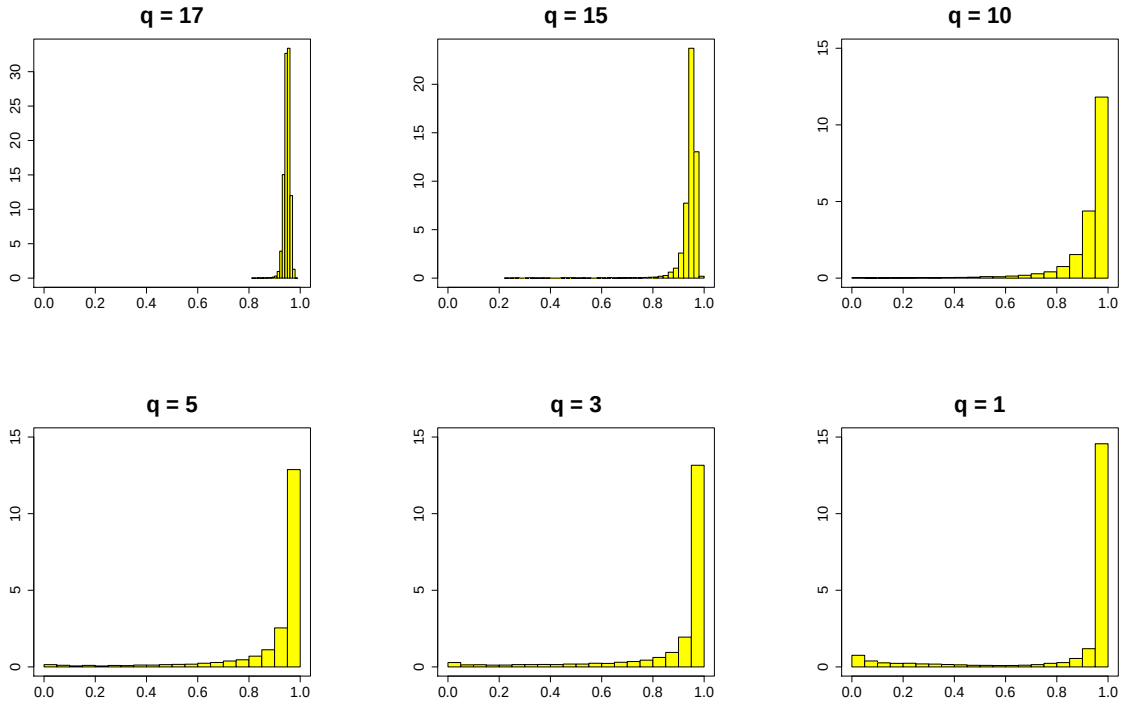


Figure 6: qp-hist plots for $G_1 = (V, E_1)$ with $n = 20$.

We now apply the qp-procedure for the case with concentration graph G_2, $n = 20, 50$ and $q = 5, 10$; see Figure 9 and Table 3. The graph G_2 is not sparse and both $G_2^{(5)}$ and $G_2^{(10)}$ are even more dense, and this affects the shape of the qp-hist plots in Figure 9. Indeed, all the three histograms are clearly less asymmetric than the corresponding histograms in Figure 6; note also that this is less evident in the case $n = 20$ and $q = 10$ because the quantity $(n - q)$ is smaller than in the other two cases.

We deem that this kind of behavior of the qp-hist plot should be read as an indication that the considered q-partial graphs do not provide satisfying approximations of the required concentration graphs. Hence, if the value of q cannot be increased then we suggest that the application of any learning procedure based on limited-order partial correlations should be avoided for the problem under analysis.

We close this section applying the qp-procedure to a subset of the gene expression data from the study by West et al. (2001). This subset was extracted and analysed originally by Jones et al. (2005) and contains the expression profiles for $p = 150$ genes associated with the estrogen receptor pathway coming from $n = 49$ breast tumor samples.

We have applied the qp-procedure with $q = 20$ and the qp-hist and qp-clique plots, given in Figure 10, provide a strong indication that $G^{(20)}$ is sparse. Hence, we set $\beta^* = 0.975$ and, in this way, we identify a graph with 7240 out of 11 175 possible edges and whose largest clique has size 24 which can be taken as an estimate of the maximum size of the highly interconnected sets of interacting genes. Such sets are a class of the so-called network motifs (Milo et al., 2002) which are characteristic network patterns whose identification can be used to draw hypotheses on basic cellular

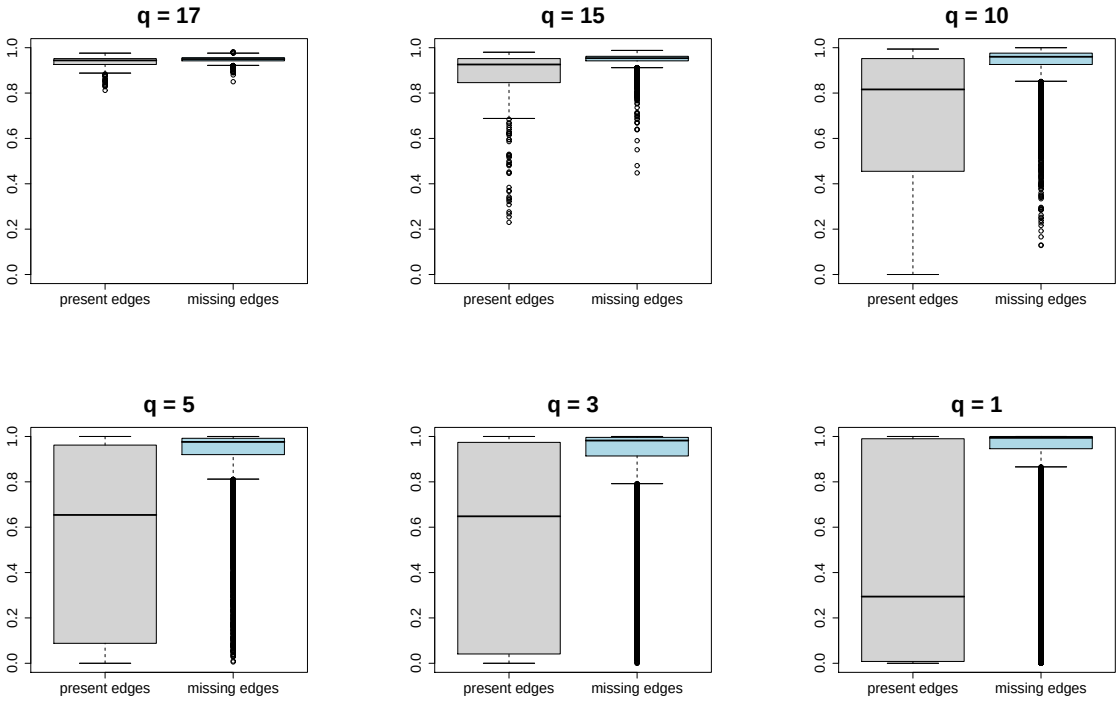


Figure 7: Distribution of the non-rejection rate for present and missing edges of $G_1 = (V, E_1)$, to be associated with the corresponding histograms in Figure 6.

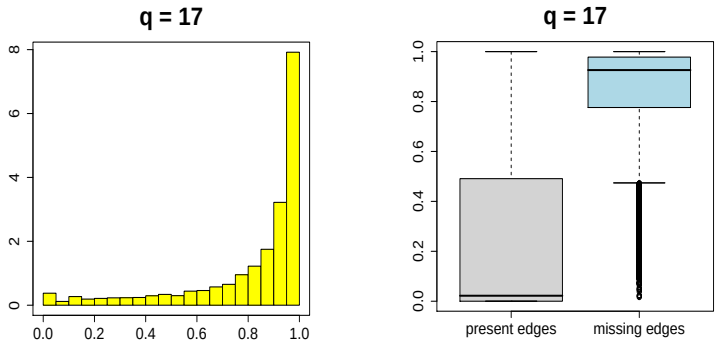


Figure 8: qp-hist plot and associated distributions of the non-rejection rate for present and missing edges of $G_1 = (V, E_1)$, resulting from the application of the qp-procedure where $n = 150$ and $q = 17$.

mechanisms (Yeger-Lotem et al., 2005). Note that the theory of q-partial graphs developed in this paper, and implemented through the qp-procedure, allows us to obtain this estimate, and eventually explore other ones, in relationship to the amount of true interactions we are willing to remove and

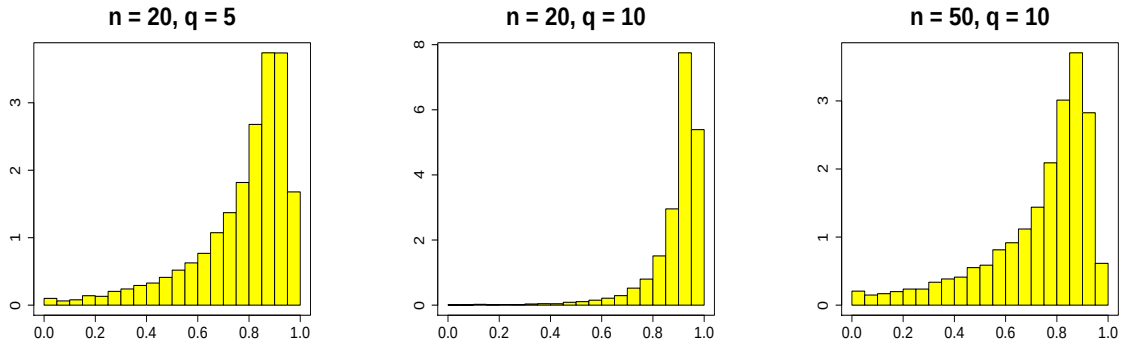


Figure 9: *qp*-hist plots and associated distributions of the non-rejection rate for present and missing edges of $G_2 = (V, E_2)$, resulting from the application of the *qp*-procedure for different values of n and q.

the dimension of the data. Such a feature may be a critical piece of information when dealing with real data for which we lack background knowledge on its underlying structure of interactions.

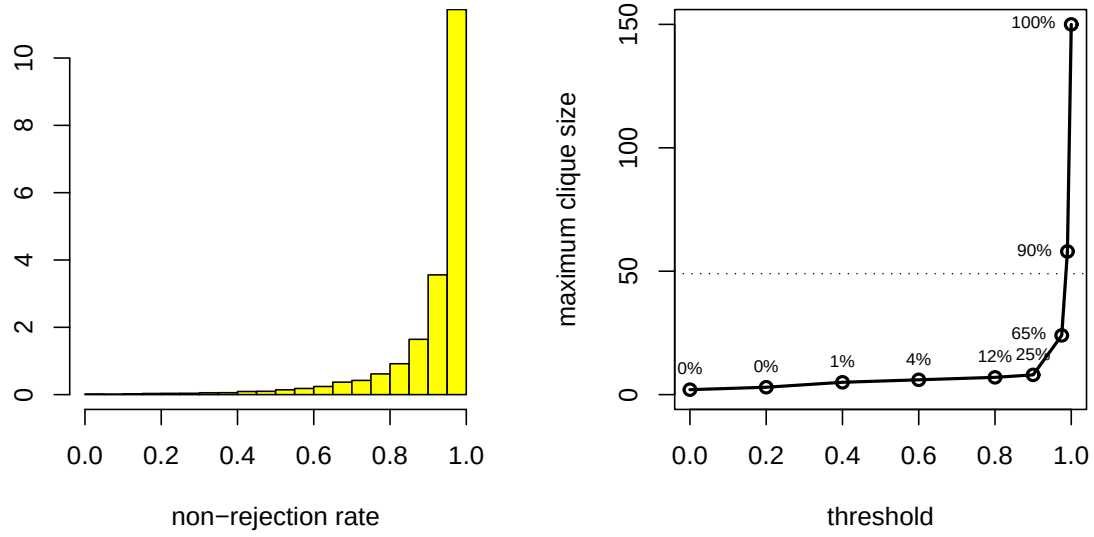


Figure 10: Estrogen receptor data of West et al. (2001): *qp*-hist and *qp*-clique plots for $q = 20$.

n	q	thr.	l.c.	% pre.	err.	% err.	% imp.
20	5						
		0.30	5	3.6	1342	12.45	6.78
		0.60	10	15.7	1099	11.66	12.72
		0.80	21	40.8	735	11.11	16.82
		0.85	29	54.2	580	11.33	15.16
		0.90	55	72.9	328	10.84	18.89
		0.95	103	91.6	90	9.59	28.18
		0.97	123	96.5	31	7.81	41.55
		0.98	134	98.3	23	12.30	7.94
		0.99	144	99.5	6	10.00	25.15
20	10						
		0.30	3	0.5	1451	13.05	2.36
		0.60	5	2.8	1333	12.27	8.13
		0.80	7	11.9	1094	11.12	16.77
		0.85	9	19.5	971	10.80	19.19
		0.90	12	34.3	758	10.32	22.72
		0.95	43	73.1	292	9.69	27.44
		0.97	88	92.4	76	8.91	33.31
		0.98	116	97.8	20	8.16	38.90
		0.99	141	99.7	2	6.90	48.38
50	10						
		0.30	6	6.0	1171	11.14	16.59
		0.60	9	21.4	869	9.89	25.96
		0.80	17	49.2	518	9.13	31.69
		0.85	27	64.3	351	8.79	34.20
		0.90	62	82.8	152	7.91	40.81
		0.95	120	96.9	27	7.87	41.08
		0.97	134	99.4	7	9.59	28.23
		0.98	143	99.8	3	12.50	6.44
		0.99	148	100.0	0	0.00	100.00

Table 3: Graph $G_2 = (V, E_2)$. Numerical description of the output of the qp-procedure applied for different values of n and q. See Table 1 for a description of columns.

6. Discussion

This paper provides two main contributions: the theory related to q-partial graphs and the qp-procedure.

The theory of q-partial graphs clarifies the connection between the sparseness of the concentration graph and the usefulness of marginal distributions in structure learning, under the assumption of faithfulness.

The qp-procedure is designed to learn q-partial graphs and overcomes the main drawbacks of the existing procedures based on limited-order partial correlations. Furthermore, our procedure has several advantages. Most importantly, it is robust with respect to the assumption of faithfulness because the estimation of the non-rejection rate is based on a large number of statistical tests involving different marginal distributions and, therefore, a zero q-order partial correlation deriving from the

lack of faithfulness has a very weak impact on the resulting estimate. Apart from faithfulness, the *qp*-procedure does not require any additional assumptions with respect to traditional structure learning procedures and, in particular, the sparseness of the concentration graph, despite being crucial for the effectiveness of the procedure, is not assumed but exploited when present. In the case the *qp*-hist and *qp*-clique plots provide and indication that the concentration graph is not sparse, then this should be read as a warning on the real usefulness of limited-order partial correlations in the problem under analysis. The fact that the *qp*-procedure is designed to select an overparameterized model might be regarded as a limitation, but in fact we deem that this is a useful feature that adds additional flexibility in its use. Indeed, the *qp*-procedure can be used as an explorative tool to assess the sparseness of the concentration graph and, therefore, the usefulness of *q*-partial correlations in structure learning. Furthermore, the result of the procedure may be applied to obtain a shrinkage estimate of the covariance matrix useful both in the case n is larger, but close, to p and in the case n is smaller than p. Finally, the set of all the submodels of the selected model may identify a restricted search space where a traditional structure learning procedure, either in a Bayesian or in a frequentist approach to inference, can be applied. In Gaussian graphical models it is assumed that X_V follows a multivariate normal distribution, and the normality of microarray data is a disputed question. We refer to Wit and McClure (2004; Section 6.2.2) for a discussion of this point, but we remark that the non-rejection rate is a quantity that can be obtained from any test for conditional independence computed on marginal distributions, and therefore it constitutes a general tool that can be used also outside the multivariate normal case.

The *qp*-procedure, jointly with other functions showing the *qp*-hist and *qp*-clique plots, has been implemented in a package, named *qp*, for the statistical software R (`http://www.r-project.org`). This package can be downloaded from The Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/src/contrib/PACKAGES.html`.

The *qp*-procedure is implemented in this package through the R and C programming languages requiring 10 minutes in a laptop 1.33GHz PowerPC G4 with 1.25 Gbyte RAM running Mac OS X, as well as in a desktop Intel 1.60GHz P4 with 1 Gbyte RAM running Linux, to perform the calculations of one of the simulations involving $p = 150$ variables, $n = 50$ observations, and $q = 15$ sampling 500 conditioning subsets to estimate the non-rejection rate for each of the 11 175 adjacencies. Note also that the $p \times (p-1)/2$ non-rejection rates could be estimated in parallel and thus such an implementation would greatly improve the performance.

Acknowledgments

We would like to thank David Madigan and David Edwards for useful discussions and the anonymous reviewers whose remarks and suggestions have improved this paper. Part of this paper was written when the second author was visiting the first author at the Universitat Pompeu Fabra supported by a mobility grant (ref. SAB2003–0197) from the Spanish Ministerio de Educación y Ciencia (MEC). Financial support to the second author has also been provided by MIUR, grant number 134079, 2005 and by the MIUR-FISR grant number 2982/Ric (Mitica). The first author is a researcher from the Ramon y Cajal program of the Spanish MEC (ref. RYC–2006–000932).

Appendix A. Graph Theory

In this appendix we present the graph theory required for this paper and, in particular, we introduce the novel concept of *outer connectivity* that is used in Section 4 to describe the properties of q-partial graphs. We refer to Cowell et al. (1999) for a full account of graph theory usually applied in graphical models, to Diestel (2005) for the theory relating separators and independent paths and, finally, to Rosenberg and Heath (2005) for a comprehensive description of the techniques for obtaining upper and lower bounds on the sizes of graph separators.

An undirected *graph* is a pair $G = (V,E)$, where $V = \{1,\ldots,p\}$ is a finite set of *vertices* and in this paper E, called the *edge set*, is a subset of the set of unordered distinct pair of vertices. If two vertices $i, j \in V$ form an edge then we say that i and j are *adjacent* and write $(i, j) \in E$; recall that edges are unordered pairs, so that $(i, j) = (j, i)$. Graphs are usually represented by drawing a dot for each vertex and joining two of these dots by a line if the corresponding two vertices form an edge; see Figure 11 for a few examples. For a subset $A \subseteq V$ the *subgraph* of G induced by A is $G_A = (A, E_A)$ with $E_A = E \cap (A \times A)$. For two graphs with common vertex set, $G = (V,E)$ and $G' = (V,E')$, we say that G' *is larger than* G, and write $G \subseteq G'$, if $E \subseteq E'$; when the inclusion is strict, that is, $E \subset E'$, we write $G \subset G'$. The *boundary* of a vertex $v \in V$, denoted by $\mathrm{bd}_G(v)$, is the set of vertices adjacent to v. A subset $C \subseteq V$ with all vertices being mutually adjacent is called *complete*, and when V is complete then we say that G is complete. A subset $C \subseteq V$ is called a *clique* if it is maximally complete, that is, C is complete, and if $C \subset D$, then D is not complete. An undirected graph can be identified by the set $\mathcal{C}$ of its cliques. The set $\bar{E}$ is the *set of missing edges* of G; that is, for a pair $i, j \in V$, $(i, j) \in \bar{E}$ if and only if $i \neq j$ and $(i, j) \notin E$. A *path* of length $l > 0$ from v_0 to v_l is a sequence $v_0, v_1, \ldots, v_l$ of distinct vertices such that $(v_{k-1}, v_k) \in E$ for all $k = 1,\ldots,l$. Two or more paths from v_0 to v_l are *independent* if they have no common vertices other then v_0 and v_l. We can define an equivalence relation on V as

$$i \sim_p j \Leftrightarrow \text{ there is a path } v_0, v_1, \ldots, v_l \text{ with } v_0 = i, v_l = j.$$

The subgraphs induced by the equivalence classes are the *connected components* of G. If there is only one equivalence class, we say that G is connected. The subset $U \subseteq V$ is said to separate $I \subseteq V$ from $J \subseteq V$ if for every $i \in I$ and $j \in J$ all paths from i to j have at least one vertex in U. For a pair of vertices $i \neq j$ with $(i, j) \in \bar{E}$, a set $U \subseteq V$ is called a $\{i, j\}$-*separator* if it separates $\{i\}$ and $\{j\}$ in G. If either $i \in U$ or $j \in U$ then we say that U is *trivial*. If no proper subset of U is a $\{i, j\}$-separator we say that U is *minimal*; see also Cowell et al. (1999). Note that the unique possible minimal $\{i, j\}$-separators that are trivial are $\{i\}$ and $\{j\}$. Hereafter, to stress that a separator is nontrivial and minimal we denote it by S; furthermore, we denote by $\mathcal{S}_{(i,j|G)}$ the set of all nontrivial minimal $\{i, j\}$-separators in G, so that $\mathcal{S}_{(i,j|G)} = \{\emptyset\}$ if and only if i and j are in different connected components. There is a close connection between the concepts of connectivity and separation: the dimension of the smallest $\{i, j\}$-separator, that is the cardinality of the smallest (possibly non unique) set in $\mathcal{S}_{(i,j|G)}$, is called the *connectivity of i and j* because it represents both the maximum number of independent paths between i and j in G and the minimum number of vertices that need to be removed from G to make i and j disconnected (see Theorem 3.3.1 of Diestel, 2005). In order to deal with q-partial graphs we need to introduce a slightly different definition of connectivity of two vertices.

Definition 3 *Let $i \neq j$ be a pair vertices of an undirected graph $G = (V,E)$. The* outer connectivity of i and j *is defined as*

$$d(i,j|G) = \min_{S \in \mathcal{S}_{(i,j|G_{ij})}} |S|$$

where G_{ij} is the graph with vertex set V and edge set $E_{ij} = E \setminus \{(i,j)\}$.

Hence, $d(i,j|G)$ is the connectivity of i and j in G_{ij}. The latter graph is constructed by removing the edge (i,j) from G, so that if $(i,j) \in \overline{E}$ then $G = G_{ij}$. The idea here is that the edge (i,j) represents an *inner*, or direct, connection between i and j and it should not be considered when *outer*, or indirect, connectivity is of concern.

Example 1 *For the vertex set $V = \{1,\ldots,6\}$ let $G_i = (V,E_i)$, $i = 1,\ldots,3$ be the graphs in Figure 11 and let G_4 be the complete graph. Then*

- $d(2,3|G_i) = 0$ *for* $i = 1,2,3$ *whereas* $d(2,3|G_4) = 4$*;*
- $d(1,6|G_1) = 0$, $d(1,6|G_i) = 1$ *for* $i = 2,3$ *whereas* $d(1,6|G_4) = 4$*;*
- $d(3,4|G_i) = 0$ *for* $i = 1,2$ *whereas* $d(3,4|G_3) = 1$*;*
- $d(3,6|G_1) = 0$, $d(3,6|G_2) = 1$, $d(3,6|G_3) = 2$.

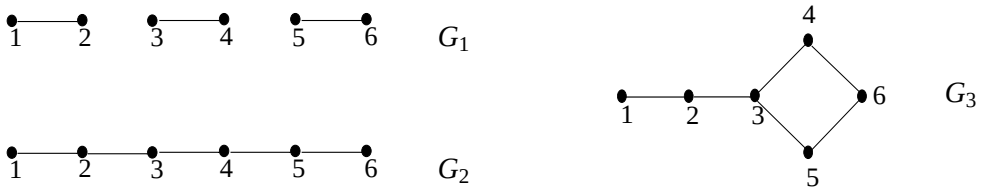


Figure 11: Examples of undirected graph.

Computing the connectivity of two vertices is known to be a NP-hard problem, however several algorithms are available to derive both upper and lower bounds to this number; see Rosenberg and Heath (2001). Here we remark that the cardinality of any $\{i,j\}$-separator in G_{ij} is an upper bound to the connectivity of i and j; consequently, since $\mathrm{bd}_{G_{ij}}(i)$ and $\mathrm{bd}_{G_{ij}}(j)$ are both $\{i,j\}$-separators in G_{ij}, then the number

$$\widetilde{d}(i,j|G) := \min\{|\mathrm{bd}_{G_{ij}}(i)|, |\mathrm{bd}_{G_{ij}}(j)|\} \tag{7}$$

provides an easy-to-compute upper bound to the outer connectivity of i and j; formally

$$d(i,j|G) \leq \widetilde{d}(i,j|G) \quad \text{for all} \quad i,j \in V;\ i \neq j. \tag{8}$$

It is useful to consider separately the pairs of vertices that define an edge in G from the pairs of vertices that are not adjacent in G. Hence, we define the *outer connectivity of the edges of $G = (V,E)$* as

$$d(E|G) := \max_{(i,j) \in E} d(i,j|G),$$

with the understanding that $d(E|G) = 0$ if $E = \emptyset$; that is if G as no edges. Similarly, the *outer connectivity of the missing edges of* $G = (V,E)$ is defined as

$$d(\bar{E}|G) := \max_{(i,j)\in E} d(i,j|G), \tag{9}$$

with the understanding that $d(\bar{E}|G) = 0$ if $\bar{E} = \emptyset$; that is if G is complete. Finally, the *outer connectivity of* $G = (V,E)$ is given by

$$\begin{aligned} d(G) &:= \max_{i,j\in V; i\neq j} d(i,j|G) \\ &= \max\{d(E|G), d(\bar{E}|G)\}. \end{aligned}$$

It is a straightforward consequence of (8) that the quantities

$$\widetilde{d}(\bar{E}|G) := \max_{(i,j)\in E} \widetilde{d}(i,j|G), \tag{10}$$

$$\widetilde{d}(E|G) := \max_{(i,j)\in E} \widetilde{d}(i,j|G), \tag{11}$$

and

$$\widetilde{d}(G) := \max\left\{\widetilde{d}(E|G), \widetilde{d}(\bar{E}|G)\right\} \tag{12}$$

are upper bounds to $d(\bar{E}|G)$, $d(E|G)$ and $d(G)$ respectively.

Example 2 *For the graphs in Figure 11 it holds that*

G_1: $d(\bar{E}|G_1) = 0,\ d(E|G_1) = 0,\ d(G_1) = 0$;

G_2: $d(\bar{E}|G_2) = 1,\ d(E|G_2) = 0,\ d(G_2) = 1$;

G_3: $d(\bar{E}|G_3) = 2,\ d(E|G_3) = 1,\ d(G_3) = 2$;

There is no strict distinction between *sparse* and *dense* graphs, however a sparse graph can be informally defined as a graph in which the number of edges is much less than the possible number of edges. Thus the complete graph is dense and the graph in which the edge set is empty is sparse; furthermore, if $G \subset G'$ than we can say that G is sparser than G'. Since G is obtained by removing edges from the larger graph G' the intuition suggests that G has a smaller number of independent paths between vertices and consequently smaller values of outer connectivity. This is formally stated in the following theorem.

Theorem 6 *Let $G = (V,E)$ and $G' = (V,E')$ be two undirected graphs such that $G \subseteq G'$. For any pair of vertices $i, j \in V$ with $i \neq j$ it holds that*

$$d(i,j|G) \leq d(i,j|G') \tag{13}$$

furthermore,

$$d(E|G) \leq d(E'|G') \qquad \textit{and} \qquad d(G) \leq d(G'). \tag{14}$$

Proof Let S be a smallest nontrivial $\{i,j\}$-separator in G'_{ij} so that $d(i,j|G') = |S|$ and every path from i to j in G'_{ij} has a vertex in S. By construction, every edge in G_{ij} is an edge in G'_{ij} and this implies that every path form i to j in G_{ij} is also a path from i to j in G'_{ij} and, consequently, that every path form i to j in G_{ij} has a vertex in S. Thus, S is a nontrivial $\{i,j\}$-separator in G_{ij} so that $d(i,j|G) \leq |S| = d(i,j|G')$, that proves (13). We consider now the first inequality in (14). Let $i,j \in V$ be two vertices such that $(i,j) \in E$ and $d(E|G) = d(i,j|G)$; recall that $(i,j) \in E$ implies $(i,j) \in E'$. Then, $d(E|G) = d(i,j|G) \leq d(i,j|G') \leq d(E'|G')$ where the first inequality holds by (13) and the second holds for every $(i,j) \in E'$. A similar reasoning can be used to prove the second inequality in (14): if i and j are such that $d(G) = d(i,j|G)$, then $d(G) = d(i,j|G) \leq d(i,j|G') \leq d(G')$ where the first inequality holds by (13) and the second is always true. ■

Note that neither the inequality $d(\bar{E}|G) \geq d(\bar{E}'|G')$ nor the inequality $d(\bar{E}|G) \leq d(\bar{E}'|G')$ are satisfied in general. For a counterexample, let $G_1 = (V,E_1)$ and $G_3 = (V,E_3)$ be the empty and the complete graph respectively, and let $G_2 = (V,E_2)$ be the graph with exactly one edge missing. Clearly, $G_1 \subseteq G_2 \subseteq G_3$, however

$$\{d(\bar{E}_1|G_1) = 0\} \leq \{d(\bar{E}_2|G_2) = p-2\} \quad \text{and} \quad \{d(\bar{E}_2|G_2) = p-2\} \geq \{d(\bar{E}_3|G_3) = 0\}.$$

References

A.J. Butte, P. Tamayo, D. Slonim, T.R. Golub and I.S. Kohane. Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. *Proceedings of the National Academy of Sciences*, 97(22): 12182-12186, 2000.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen and D.J. Spiegelhalter. *Probabilistic networks and expert systems*. Springer-Verlag, New York, 1999.

D.R. Cox and N. Wermuth. Linear dependencies represented by chain graphs (with discussion). *Statist. Sci.*, 8: 204–283, 1993.

D.R. Cox and N. Wermuth. *Multivariate dependencies: Models, analysis and interpretation*. Chapman and Hall, London, 1996.

A. de la Fuente, N. Bing, I. Hoeschele and P. Mendes. Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20: 3565-3574, 2004.

A.P. Dempster. *Elements of continuous multivariate analysis*. Addison-Wesley, Reading, Massachusetts, 1969.

A.P. Dempster. Covariance selection. *Biometrics*, 28: 157–75, 1972.

R. Diestel. (2005). *Graph theory*. Springer-Verlag, Heidelberg, 2005.

A. Dobra, C. Hans, B. Jones, J.R. Nevins and M. West. Sparse graphical models for exploring gene expression data. *J. Mult. Anal.* 90: 196-212, 2004.

M. Drton and M.D. Perlman. Model selection for Gaussian concentration graphs. *Biometrika*, 91(3): 591–602, 2004.

M. Drton and T. Richardson. Iterative conditional fitting for estimation of a covariance matrix with zeros. Technical Report no. 469, Department of Statistics, University of Washington, 2004.

R.L. Dykstra. Establishing the positive definiteness of the sample covariance matrix. *Ann. Math. Statist.*, 41(6): 2153–2154, 1970.

D.E. Edwards. *Introduction to graphical modelling*. Springer-Verlag, New York, 2000.

N. Friedman. Inferring cellular network using probabilistic graphical models. *Science*, 33: 799–805, 2004.

B. Jones, A. Dobra, C. Carvalho, C. Hans, C. Carter and M. West. Experiments in stochastic computation for high-dimensional graphical models. *Statistical Science*, 20(4): 388–400, 2005.

B. Jones and M. West. Covariance decomposition in undirected Gaussian graphical models. *Biometrika*, 92(4): 779-786, 2005.

S.L. Lauritzen. *Graphical models*. Oxford University Press, Oxford, 1996.

E.L. Lehmann. *Testing statistical hypotheses, 2nd edition*. Wiley, New York, 1986.

P.M. Magwene and J. Kim. Estimating genomic coexpression networks using first-order conditional independence. *Genome Biology*, 5: R100, 2004.

A.A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R.D. Favera and A. Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks ina mammalian cellular context. *BMC Bioinformatics*, 7(Suppl 1):S7, 2006.

R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298: 824–827, 2002.

J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, San Mateo, 1988.

A.L. Rosenberg and L.S. Heath. *Graph separators, with applications*. Kluwer Academic Publishers, New York, 2001.

A. Roverato. Hyper inverse Wishart distribution for non-decomposable graphs and its application to Bayesian inference for Gaussian graphical models. *Scand. J. Statist.*, 29: 391–411, 2002.

J. Schäfer and K. Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6): 754-764, 2005a.

J. Schäfer and K. Strimmer. Learning large-scale graphical Gaussian models from genomic data. In: J.F. Mendes. (Ed.). Proceeding of *Science of Complex Networks: from Biology to the Internet and WWW* (CNET 2004), Aveiro, PT, (Publisher: The American Institute of Physics), 2005b.

J. Schäfer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1): article 32, 2005c.

R. Steuer, J. Kurths, O. Fiehn and W. Weckwerth. Interpreting correlations in metabolomic networks. *Bioch. Soc. Trans.*, 31: 1476-1478, 2003a.

R. Steuer, J. Kurths, O. Fiehn and W. Weckwerth. Observing and interpreting correlations in metabolomic networks. *Bioinformatics*, 19: 1019-1026, 2003b.

M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J.A. Olson, J.R. Marks and J.R. Nevings. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences*, 98(20): 11462–11467, 2001.

A. Wille and P. Bühlmann. Low-order conditional independence graphs for inferring genetic networks. *Statistical Applications in Genetics and Molecular Biology*, 5(1): article 1, 2006.

A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelić, P. von Rohr, L. Thiele, E. Zitzler, W. Gruissem and P. Bühlmann. Sparse graphical Gaussian modeling of the isoprenoid gene network in *Arabidopsis thaliana*. *Genome Biology*, 5:R92, 2004.

E. Wit and J. McClure. *Statistics for microarrays. Design, analysis and inference.* Wiley, Chichester, 2004.

J. Whittaker. *Graphical models in applied multivariate statistics*. Wiley, Chichester, 1990.

F. Wong, C.K. Carter and R. Kohn. Efficient estimation of covariance selection models. *Biometrika*, 90: 809–830, 2003.

R. Yang, and J.O. Berger. Estimation of a covariance matrix using the reference priors. *Ann. Statist.*, 3: 1195–1211, 1994.

E. Yeger-Lotem, S. Sattath, N. Kashtan, S. Itzkovitz, R. Milo, R.Y. Pinter, U. Alon and H. Margalit. Network motifs in integrated cellular networks of transcription–regulation and protein–protein interaction. *Proc. Natl. Acad. Sci.*, 101(16): 5934–5939, 2004.

Journal of Machine Learning Research 7 (2006) 2651-2667 Submitted 7/06; Revised 10/06; Published 12/06

Universal Kernels

Charles A. Micchelli CAM@MATH.ALBANY.EDU
Department of Mathematics and Statistics
State University of New York
The University at Albany
Albany, New York 12222, USA

Yuesheng Xu YXU06@SYR.EDU
Haizhang Zhang HZHANG12@SYR.EDU
Department of Mathematics
Syracuse University
Syracuse, NY 13244, USA

Editor: Gabor Lugosi

Abstract

In this paper we investigate conditions on the features of a continuous kernel so that it may approximate an arbitrary continuous target function uniformly on any compact subset of the input space. A number of concrete examples are given of kernels with this universal approximating property.

Keywords: density, translation invariant kernels, radial kernels

1. Introduction

Let $\mathcal{X}$ be a prescribed input space and set $\mathbb{N}_n := \{1,2,\ldots,n\}$. We shall call a function K from $\mathcal{X}\times\mathcal{X}$ to $\mathbb{C}$ a *kernel* on $\mathcal{X}$ provided that for *any* finite sequence of inputs $\mathbf{x} := \{x_j : j \in \mathbb{N}_n\} \subseteq \mathcal{X}$ the matrix

$$K_{\mathbf{x}} := (K(x_j, x_k) : j,k \in \mathbb{N}_n) \tag{1}$$

is Hermitian and positive semi-definite. Kernels are an essential component in a multitude of novel algorithms for pattern analysis (Bishop, 1995; Hastie et al., 2001; Schölkopf and Smola, 2002). Besides their superior performance on a wide spectrum of learning tasks from data, they have a substantial theoretical basis, as they are reproducing kernels of Hilbert spaces of functions on $\mathcal{X}$ for which point evaluation is always continuous (Aronszajn, 1950). Such spaces are called *Reproducing Kernel Hilbert Spaces* (RKHS) and an important reason for the interest in kernels is the (essentially) unique correspondence between them and RKHS. This relationship leads, by means of the regularization approach to learning, functions having the representation

$$f := \sum_{j\in\mathbb{N}_n} c_j K(\cdot, x_j) \tag{2}$$

where $\{c_j : j \in \mathbb{N}_n\} \subseteq \mathbb{C}$ are parameters typically obtained from training data (Bishop, 1995; Evgeniou et al., 2000; Hastie et al., 2001; Schölkopf and Smola, 2002). This useful fact is known as the *Representer Theorem* and has wide applicability (Schölkopf et al., 1999; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004; Wahba, 1990). We shall refer to the function in the sum on the right hand side of (2) as *sections* of the kernel K.

Certainly, the choice of the kernel in (2) affects the performance of kernel based learning algorithms and so, is important. For recent work in this direction, see Argyriou et al. (2005, 2006), Bach et al. (2004), Lanckriet et al. (2004), Micchelli and Pontil (2005), Micchelli et al. (2006), Neumann et al. (2004), Ong et al. (2005), Sonnenburg et al. (2006) and references therein. Following Poggio et al. (2002), we ask a conceptually simpler, but very basic question about choosing the kernel : can the function representation (2), as the number of summands increases without bound, approximate any target function arbitrarily close? In the study of this question it is important which norm is used to compute the error between the function appearing in (2) and a given target function. Indeed, it is well-known that if we use the norm in the RKHS whose kernel is K then all members of this Hilbert space are approximable arbitrarily by functions of the type appearing in (2). Actually, this is the way the Hilbert space associated with a kernel is constructed from the kernel itself (Aronszajn, 1950).

Our concern here is with the *uniform norm*. To this end, we assume that the input space $\mathcal{X}$ is a Hausdorff topological space and that all kernels to be considered are *continuous* on $\mathcal{X} \times \mathcal{X}$. To begin to address the problem which interests us here, we let $\mathcal{Z}$ be a fixed but arbitrary compact subset of $\mathcal{X}$ and, as usual, let $C(\mathcal{Z})$ be the space of all continuous complex-valued functions from $\mathcal{Z}$ to $\mathbb{C}$ equipped with maximum norm $\|\cdot\|_{\mathcal{Z}}$. Our hypothesis that the input space is Hausdorff ensures that it has an abundance of compact subsets. We shall always enforce this hypothesis throughout and for simplicity of presentation we do not mention it again.

Given a kernel K we form the space of *kernel sections*

$$K(\mathcal{Z}) := \overline{\text{span}}\{K_y : y \in \mathcal{Z}\},$$

where $K_y : \mathcal{X} \to \mathbb{C}$ is the function defined at every $x \in \mathcal{X}$ by the equation $K_y(x) := K(x,y)$. The set $K(\mathcal{Z})$ consists of all functions in $C(\mathcal{Z})$ which are uniform limits of functions of the form (2) where $\{x_j : j \in \mathbb{N}_n\} \subseteq \mathcal{Z}$.

We want to identify kernels with the following *universal approximating property*: given *any* prescribed compact subset $\mathcal{Z}$ of $\mathcal{X}$, any positive number ε and any function $f \in C(\mathcal{Z})$ there is a function $g \in K(\mathcal{Z})$ such that $\|f - g\|_{\mathcal{Z}} \leq \varepsilon$. That is, for any choice of compact subset $\mathcal{Z}$ of the input space $\mathcal{X}$, the set $K(\mathcal{Z})$ is *dense* in $C(\mathcal{Z})$ in the maximum norm. When a kernel has this property we call it a *universal kernel*. In other words, a universal kernel K has the property that $K(\mathcal{Z}) = C(\mathcal{Z})$. It is this question of characterizing universal kernels that we address here. We shall demonstrate that it has a satisfactory resolution in terms of any feature map representation of the kernel K. Indeed, we provide a necessary and sufficient condition for K to have the universal approximating property in terms of its features, thereby completing preliminary remarks made about this problem in Micchelli and Pontil (2004) and Micchelli et al. (2003).

Concrete examples of kernels with their feature maps and observations about the associated density problem are investigated in Section 3. In Section 4, we stress translation invariant kernels on $\mathbb{R}^d$ and give several useful sufficient conditions for K to be a universal translation invariant kernel. This discussion includes the popular choice of the Gaussian kernel. We end the paper with a remark about issues for further investigation.

2. Kernels Defined by Feature Maps

We start from a Hilbert space $\mathcal{W}$ over $\mathbb{C}$ and a continuous kernel K on $\mathcal{X}\times\mathcal{X}$. A *feature map* for the kernel K is any *continuous* function $\Phi:\mathcal{X}\to\mathcal{W}$ such that for each $(x,y)\in\mathcal{X}\times\mathcal{X}$

$$K(x,y)=(\Phi(x),\Phi(y))_{\mathcal{W}} \tag{3}$$

where $(\cdot,\cdot)_{\mathcal{W}}$ is the inner product on $\mathcal{W}$. Every kernel has such a representation and conversely whenever it does then it is a kernel. However, a feature space representation is *not* unique. Let us elaborate on these well-known facts. First, it is straightforward to see that any function K which has the representation (3) is a kernel. The reason for this fact is that the input matrix appearing in (1) is formed by the mutual inner products of the set of vectors $\{\Phi(x_j):j\in\mathbb{N}_n\}$ and such a matrix is certainly Hermitian and positive semi-definite. To establish the converse, we first construct the Hilbert space $\mathcal{H}$ associated with the continuous kernel K and then observe for all $x,y\in\mathcal{X}$, by the reproducing kernel property, that

$$K(x,y):=(K_x,K_y)_{\mathcal{H}}.$$

Hence, we may choose $\mathcal{W}=\mathcal{H}$ and for any $x\in\mathcal{X}$ we let $\Phi(x)=K_x$. This feature space representation is continuous because for all $x,y\in\mathcal{X}$ we have that

$$\|\Phi(x)-\Phi(y)\|_{\mathcal{W}}^2=K(x,x)+K(y,y)-K(x,y)-K(y,x)$$

where $\|\cdot\|_{\mathcal{W}}$ denotes the norm on $\mathcal{W}$.

There are alternate means to construct a feature space representation for a continuous kernel K which has the advantage that the Hilbert space $\mathcal{W}$ can be chosen to be *separable*. To construct such a representation we must *choose* a compact subset $\mathcal{Z}$ of $\mathcal{X}$ and a finite Borel measure μ on $\mathcal{Z}$ with $\operatorname{supp}(\mu)=\mathcal{Z}$ (see (15) for the definition of the support of a Borel measure). This measure yields a linear operator $T:L^2(\mathcal{Z},\mu)\to L^2(\mathcal{Z},\mu)$ defined for $g\in L^2(\mathcal{Z},\mu)$ by the equation

$$Tg:=\int_{\mathcal{Z}}K(\cdot,y)g(y)d\mu(y). \tag{4}$$

Following the ideas of Mercer (1909), T has countably many nonnegative Eigenvalues (each of finite multiplicities with zero as the only accumulation point of nonnegative Eigenvalues) $\{\lambda_i:i\in\mathbb{N}\}$ and corresponding orthonormal Eigenfunctions $\{\phi_i:i\in\mathbb{N}\}\subseteq L^2(\mathcal{Z},\mu)$ such that

$$K(x,y):=\sum_{i\in\mathbb{N}}\lambda_i\phi_i(x)\overline{\phi_i(y)},\quad (x,y)\in\mathcal{Z}\times\mathcal{Z} \tag{5}$$

where the series above converges absolutely and uniformly on $\mathcal{Z}\times\mathcal{Z}$, see also Lax (2002).

To write K in the form (3), we let $\ell^2(\mathbb{N})$ be the Hilbert space of square summable sequences on $\mathbb{N}$ and define a feature map $\Psi:\mathcal{Z}\to\ell^2(\mathbb{N})$ at each $x\in\mathcal{Z}$ and $j\in\mathbb{N}$ as

$$\Psi(x)(j):=\sqrt{\lambda_j}\phi_j(x).$$

Therefore, the Mercer representation in Equation (5) establishes for each $x,y\in\mathcal{Z}$ that

$$K(x,y)=(\Psi(x),\Psi(y))_{\ell^2(\mathbb{N})}.$$

Since we have for all $x, y \in \mathcal{Z}$ that

$$\|\Psi(x) - \Psi(y)\|^2_{\ell^2(\mathbb{N})} = K(x,x) + K(y,y) - K(x,y) - K(y,x),$$

the continuity of the K implies that the feature map $\Psi : \mathcal{Z} \to \ell^2(\mathbb{N})$ is also continuous.

Of course, to find an Eigenfunction feature representation of a kernel, except in special circumstances, is a serious challenge both analytically and computationally. Moreover, it should be observed that this feature space representation depends on the measure μ. Recent extensions of the Mercer theorem can be found in Sun (2005) and the reference therein.

Let us now return to the general case of formula (3). To this end, we need to recall facts about the dual space of $C(\mathcal{Z})$, that is, the space of all continuous linear functionals on $C(\mathcal{Z})$. By the Riesz representation theorem the linear functionals in dual space of $C(\mathcal{Z})$ are identified as regular complex-valued measures on $\mathcal{Z}$ (see, for example, Lax, 2002; Royden, 1988). The norm of a complex-valued measure, which is inherited from the norm on $C(\mathcal{Z})$, is its *total variation* and is defined as

$$\mathrm{TV}(\nu) := \sup\{|\int_{\mathcal{Z}} g(x) d\nu(x)| : \|g\|_{\mathcal{Z}} \le 1, g \in C(\mathcal{Z})\}.$$

We denote the space of all regular complex-valued measures on $\mathcal{Z}$ with this norm by $B(\mathcal{Z})$. For any $\nu \in B(\mathcal{Z})$, we wish to define the integral $\int_{\mathcal{Z}} \Phi(x) d\nu(x)$ as an element of $\mathcal{W}$. This is done by noting that the *conjugate linear functional* L defined on $\mathcal{W}$ for each $u \in \mathcal{W}$ by the equation

$$L(u) := \int_{\mathcal{Z}} (\Phi(x), u)_{\mathcal{W}} d\nu(x) \tag{6}$$

has a norm satisfying the inequality

$$\|L\| \le \mathrm{TV}(\nu)\|\Phi\|_\infty < \infty,$$

where $\|\Phi\|_\infty := \max\{\|\Phi(x)\|_{\mathcal{W}} : x \in \mathcal{Z}\}$. Therefore, by the Riesz representation theorem, for the Hilbert space $\mathcal{W}$ (Lax, 2002; Rudin, 1991) there exists a *unique* element $w \in \mathcal{W}$ such that for each $u \in \mathcal{W}$ that

$$L(u) = (w, u)_{\mathcal{W}}.$$

It is this vector w which we shall denote by $\int_{\mathcal{Z}} \Phi(x) d\nu(x)$. Consequently, we have the useful formula

$$(\int_{\mathcal{Z}} \Phi(x) d\nu(x), u)_{\mathcal{W}} = \int_{\mathcal{Z}} (\Phi(x), u)_{\mathcal{W}} d\nu(x) \tag{7}$$

valid for all $u \in \mathcal{W}$.

Next, we introduce a map $U : B(\mathcal{Z}) \to \mathcal{W}$ by letting for each $\nu \in B(\mathcal{Z})$

$$U(\nu) := \int_{\mathcal{Z}} \Phi(x) d\nu(x) \tag{8}$$

and so the formula (7) becomes

$$(U(\nu), u)_{\mathcal{W}} = \int_{\mathcal{Z}} (\Phi(x), u)_{\mathcal{W}} d\nu(x). \tag{9}$$

For any $y \in \mathcal{Z}$ we set $u = \Phi(y)$ in formula (9) above to obtain by the definition of the kernel (3) that

$$(U(\nu), \Phi(y))_{\mathcal{W}} = \int_{\mathcal{Z}} K(x,y) d\nu(x). \tag{10}$$

We now conjugate both sides of this equation, integrate both sides of the resulting equation with respect to the complex-valued measure ν and then simplify the resulting left side by another application of Equation (7) with the choice $u = U(\nu)$. Next, we use the feature space representation of the kernel in (3) on the right hand side of the equation to obtain the equation

$$\|U(\nu)\|_{\mathcal{W}}^2 = \int_{\mathcal{Z}}\int_{\mathcal{Z}} K(x,y)d\overline{\nu}(y)d\nu(x) \tag{11}$$

where $\overline{\nu}$ is the conjugate of the complex-valued measure ν defined for each Borel set $\mathcal{S} \subseteq \mathcal{Z}$ by $\overline{\nu}(\mathcal{S}) := \overline{\nu(\mathcal{S})}$. We remark here that since the kernel K is continuous and $\mathcal{Z}$ is compact, Fubini's theorem assures that we can interchange the order in the integral on the right hand side of the above equation (see, for example, Royden, 1988). Moreover, from this formula it follows that the linear operator U is continuous. Indeed, its norm satisfies the inequality

$$\|U\| \leq \sqrt{\|K\|_\infty}$$

where $\|K\|_\infty$ denotes the maxium norm of K on $C(\mathcal{Z}\times\mathcal{Z})$.

To continue, we recall the notion of *annihilator* of a subset $\mathcal{V}$ of $C(\mathcal{Z})$. This consists of all elements in $B(\mathcal{Z})$ which are zero on all functions in $\mathcal{V}$. In other words, we have that

$$\mathcal{V}^\perp := \{\nu : \nu \in B(\mathcal{Z}), \int_{\mathcal{Z}} f(x)d\nu(x) = 0, f \in \mathcal{V}\}.$$

Note that the annihilator of a subset of $C(\mathcal{Z})$ is a subspace of $B(\mathcal{Z})$. Furthermore, the *closed linear span* of subset $\mathcal{V}$ of $C(\mathcal{Z})$, denoted by $\overline{\text{span}}\,\mathcal{V}$, has the same annihilator as the set $\mathcal{V}$ itself. Moreover, two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ of $C(\mathcal{Z})$ have the same annihilator if and only if $\overline{\text{span}}\,\mathcal{V}_1 = \overline{\text{span}}\,\mathcal{V}_2$. Also, recall that two closed subspaces are equal if and only if their annihilators are the same (see, for example, Lax, 2002; Royden, 1988; Rudin, 1991, for these facts).

We shall denote the null space of U by $\mathcal{N}(U)$, that is, the subspace of all elements ν in $B(\mathcal{Z})$ for which $U(\nu)$ is zero, given by

$$\mathcal{N}(U) := \{\nu : \nu \in B(\mathcal{Z}), U(\nu) = 0\}.$$

We remark here that the operator U depends on the set $\mathcal{Z}$.

Proposition 1 *If $\mathcal{Z}$ is a compact subset of the input space $\mathcal{X}$ then*

$$K(\mathcal{Z})^\perp = \mathcal{N}(U). \tag{12}$$

Consequently, $K(\mathcal{Z}) = C(\mathcal{Z})$ if and only if U is injective.

Proof By the Hahn Banach theorem, the linear span of a subset $\mathcal{V}$ is dense in $C(\mathcal{Z})$, that is, $\overline{\text{span}}\,\mathcal{V} = C(\mathcal{Z})$ if and only if $\mathcal{V}^\perp = \{0\}$ (Lax, 2002; Royden, 1988; Rudin, 1991). Therefore, the second claim follows from (12). We now turn to the proof of this equation. If $\nu \in K(\mathcal{Z})^\perp$ then by definition, for all $y \in \mathcal{Z}$ we have that

$$\int_{\mathcal{Z}} K(x,y)d\nu(x) = 0$$

and so by (11) we get that $\nu \in \mathcal{N}(U)$. Therefore, we have established that $K(\mathcal{Z})^\perp \subseteq \mathcal{N}(U)$. To prove the opposite inclusion we suppose that $\nu \in \mathcal{N}(U)$, then appeal to (10) and conclude that $\nu \in K(\mathcal{Z})^\perp$, thereby proving the theorem.

■

Equation (7) has another consequence. To this end, we introduce a subspace of $\mathcal{W}$ defined as

$$\Phi(\mathcal{Z}) := \overline{\operatorname{span}}\{\Phi(x) : x \in \mathcal{Z}\}.$$

If Q is a linear mapping between two linear spaces and $\mathcal{S}$ is a subset of its domain we use the standard notation $Q(\mathcal{S})$ for its image under Q. When the set $\mathcal{S}$ is the *domain* of Q then its image is the range of Q and is denoted by $\mathcal{R}(Q)$. From these definitions it follows that $Q(\operatorname{span}\mathcal{S}) = \operatorname{span} Q(\mathcal{S})$.

Proposition 2

$$\overline{\mathcal{R}(U)} = \Phi(\mathcal{Z}). \tag{13}$$

Proof We shall prove the proposition by showing that

$$\mathcal{R}(U)^\perp = \Phi(\mathcal{Z})^\perp.$$

If $u \in \Phi(\mathcal{Z})^\perp$ then (7) implies for any $\nu \in B(\mathcal{Z})$ that $u \in \mathcal{R}(U)^\perp$. Conversely, if $u \in \mathcal{R}(U)^\perp$ then again we get for any $\nu \in B(\mathcal{Z})$ from (7) that $\int_{\mathcal{Z}} (\Phi(x), u)_{\mathcal{W}} d\nu(x) = 0$. In particular, choosing ν to be the point evaluation at an arbitrary $x \in \mathcal{Z}$ we obtain that $(\Phi(x), u)_{\mathcal{W}} = 0$ and so we conclude that $u \in \Phi(\mathcal{Z})^\perp$, thereby establishing (13). ■

Let us now introduce another linear operator $V : \mathcal{W} \to C(\mathcal{Z})$ defined for any $u \in \mathcal{W}$ and $x \in \mathcal{Z}$ as $(V(u))(x) := (\Phi(x), u)_{\mathcal{W}}$. Certainly, V is bounded, as its norm satisfies the inequality $\|V\| \leq \|\Phi\|_\infty$. Moreover, according to (7) we have that

$$(U(\nu), u)_{\mathcal{W}} = \int_{\mathcal{Z}} (V(u))(x) d\nu(x) \tag{14}$$

which means that the *adjoint* of the operator V denoted by $V^* : B(\mathcal{Z}) \to \mathcal{W}$ is U, that is, $U = V^*$. Next, we point out a consequence of this fact and Proposition 1.

Corollary 3 $K(\mathcal{Z}) = \overline{\mathcal{R}(V)}$.

Proof It is generally true for any bounded linear operator that $\mathcal{N}(Q^*) = \mathcal{R}(Q)^\perp$ (Lax, 2002; Rudin, 1991) and, in particular, $\mathcal{N}(U) = \mathcal{R}(V)^\perp$ so the result follows directly from Proposition 1. ■

We recall that the linear span of a subset $\mathcal{S}$ is dense in $\mathcal{W}$, that is, $\overline{\operatorname{span}}S = \mathcal{W}$, if and only if the only $u \in \mathcal{W}$ with $(u, v) = 0$ for all $v \in \mathcal{S}$ is $u = 0$ (We already use a similar fact for the space $C(\mathcal{Z})$). It follows directly from Equation (14), for any subset $\mathcal{S}$ of $\mathcal{W}$ such that $\operatorname{span}\mathcal{S}$ is dense in $\mathcal{W}$, that $V(\mathcal{S})^\perp = \mathcal{N}(U)$ and so with this remark and Proposition 1 we conclude that

$$K(\mathcal{Z}) = \overline{\operatorname{span}} V(\mathcal{S}).$$

We use this equation in the following manner. Recall that a subset $\mathcal{Y}$ of $\mathcal{W}$ is *orthonormal* if for every distinct elements $u,v \in \mathcal{Y}$ we have that $(u,v) = 0$ and also $(u,u) = 1$. Every Hilbert space has an orthonormal basis $\mathcal{Y}$ (which may not be countable) such that for every $u \in \mathcal{W}$ the set $\{y : y \in \mathcal{Y}, (u,y) \neq 0\}$ is countable. Moreover, we have for $u \in \mathcal{W}$ the decomposition

$$u = \sum_{y \in \mathcal{Y}} (u,y)y,$$

where the sum on the right hand side of this equation converges in $\mathcal{W}$ for *any* ordering of elements of $\mathcal{Y}$ (Lax, 2002; Rudin, 1991). Corresponding to each element y in an orthonormal basis $\mathcal{Y}$ of $\mathcal{W}$ we define the function $F_y \in C(\mathcal{Z})$ at $x \in \mathcal{Z}$ by the equation $F_y(x) = (\Phi(x),y)_{\mathcal{W}}$ and introduce the corresponding subspace of $C(\mathcal{Z})$

$$\Phi(\mathcal{Y}) := \overline{\text{span}}\{F_y : y \in \mathcal{Y}\}.$$

Note the important difference between the sets $\Phi(\mathcal{Z})$ and $\Phi(\mathcal{Y})$. The first is in the Hilbert space $\mathcal{W}$ and the second is in $C(\mathcal{Z})$. Combining the above remarks we obtain the following equivalence between density of kernel representation and feature function density in $C(\mathcal{Z})$.

Theorem 4 *If $\mathcal{Z}$ is a compact subset of the input space $\mathcal{X}$, K a kernel with feature space representation (3) and $\mathcal{Y}$ an orthornormal basis for $\mathcal{W}$ then $K(\mathcal{Z}) = \Phi(\mathcal{Y})$.*

Parallel to our notion that a kernel is universal, we say a *feature map* Φ is *universal* provided that given any compact subset $\mathcal{Z}$ of the input space $\mathcal{X}$, any positive number ε and any function $f \in C(\mathcal{Z})$ there is a function $g \in \Phi(\mathcal{Y})$ such that $\|f - g\|_{\mathcal{Z}} \leq \varepsilon$. That is, for *any* choice of compact subset $\mathcal{Z}$ of the input space $\mathcal{X}$, the set $\{F_y : y \in \mathcal{Y}\}$ is *dense* in $C(\mathcal{Z})$. In other words, we have that $\Phi(\mathcal{Y}) = C(\mathcal{Z})$. Therefore, with this terminology we can succinctly summarize our conclusion in Theorem 4 by saying that *a kernel K expressed in feature space form (3) is universal if and only if its features are universal*!

We now consider an alternate way to express the universality of a kernel K in terms of the operator T and the corresponding measure μ defined in Equation (4) which determines it. To this end, we recall the that the *support* of a Borel measure ν on $\mathcal{Z}$ is defined to be the closed set

$$\text{supp}(\nu) := \bigcap \left\{ S \subseteq \mathcal{Z} : S \text{ is closed, } \nu(S^C) = 0 \right\}. \tag{15}$$

Consequently, if $\int_{\mathcal{Z}} f(x)d\nu(x) = 0$, ν a Borel measure with $\text{supp}(\nu) = \mathcal{Z}$ and f is a nonnegative and continuous function on $\mathcal{Z}$ then $f = 0$.

The first statement we make is about the mapping T defined in Equation (4) which is an immediate consequence of Theorem 4 concerning its Eigenfunctions.

Corollary 5 *If K is a kernel on an input space $\mathcal{X}$, $\mathcal{Z}$ a compact subset of $\mathcal{X}$, $\{\lambda_i : i \in \mathbb{N}\} \subseteq \mathbb{R}_+ \backslash \{0\}$ and $\{\phi_i : i \in \mathbb{N}\} \subseteq L^2(\mathcal{Z},\mu)$ are the nonzero Eigenvalues and corresponding orthonormal Eigenfunctions of the compact operator T where $\text{supp}(\mu) = \mathcal{Z}$ then $K(\mathcal{Z}) = C(\mathcal{Z})$ if and only if $\overline{\text{span}}\{\phi_i : i \in \mathbb{N}\} = C(\mathcal{Z})$.*

The next comment concerns the range of the operator T.

Theorem 6 *If supp(μ)= $\mathcal{Z}$ for the measure appearing in Equation (4) then $K(\mathcal{Z}) = \overline{\mathcal{R}(T)}$.*

Proof It suffices to show that $K(Z)^\perp = \mathcal{R}(T)^\perp$. If $\nu \in K(Z)^\perp$ then for each $y \in Z$

$$\int_Z K(x,y)d\nu(x) = 0.$$

By Fubini's theorem, we observe for each $g \in L^2(Z,\mu)$ that

$$\int_Z (Tg)(x)d\nu(x) = \int_Z g(y)\{\int_Z K(x,y)d\nu(x)\}d\mu(y) = 0$$

and so we conclude that $\nu \in \mathcal{R}(T)^\perp$. Conversely, $\nu \in \mathcal{R}(T)^\perp$ then by the above equation we obtain for any $g \in C(Z)$ that

$$\int_Z g(y)\{\int_Z K(x,y)d\nu(x)\}d\mu(y) = 0.$$

We now choose $g = \overline{\int_Z K(x,\cdot)d\nu(x)}$ in this equation and conclude that

$$\int_Z |g(y)|^2 d\mu(y) = 0.$$

Since $\operatorname{supp}(\mu) = Z$, we obtain that $g = 0$, that is, $\nu \in K(Z)^\perp$. ■

As a consequence of Theorem 6, we observe that $K(Z) = C(Z)$ if and only if $\overline{\mathcal{R}(T)} = C(Z)$.

We end this section by remarking that the results presented here may be extended from $C(Z)$ to L^p- spaces where $p \in [1,\infty)$. However, as we remarked in the introduction our focus here is on the maximum norm and so we do not go into this matter here.

3. Examples of Universal Kernels

In this section, we give examples of kernels defined by feature maps and study the corresponding density problem. We begin with a set $\{\phi_j : j \in I\}$ of continuous complex-valued functions on X where I is a countable set of indices and define the kernel K by

$$K(x,y) := \sum_{j\in I} \phi_j(x)\overline{\phi_j(y)}, \quad (x,y) \in X \times X, \tag{16}$$

where we assume that the series converges uniformly on $Z \times Z$ for every compact subset Z of X. To make use of the presentation in Section 2 we set $\mathcal{W} = \ell^2(\mathbb{N})$, choose the standard orthonormal basis $\mathcal{Y}$ for $\mathcal{W}$ in Theorem 4 and obtain the following result.

Theorem 7 *The kernel K defined by Equation (16) is universal if and only if the set of features $\{\phi_j : j \in I\}$ is universal.*

We shall now apply Theorem 7 to *dot product kernels* on various domains of $\mathbb{R}^d$ and $\mathbb{C}^d$. To this end, we start with an entire function G defined at any $z \in \mathbb{C}$ by the equation

$$G(z) := \sum_{n\in\mathbb{Z}_+} a_n z^n \tag{17}$$

where the coefficients $\{a_n : n \in \mathbb{Z}_+\}$ are assumed to be all *positive*. The function G induces the dot product kernel K defined at $x,y \in \mathbb{C}^d$ by the equation

$$K(x,y) := G((x,y)) \tag{18}$$

where we shall always use (x,y) for the standard inner product between the vectors x and y. For an extensive discussion of dot product kernels see FitzGerald et al. (1995).

Corollary 8 *The dot product kernel defined in Equation (18) is universal on $\mathbb{C}^d$ and $\mathbb{R}^d$.*

Proof For any lattice vector $\alpha := (\alpha_j : j \in \mathbb{N}_d) \in \mathbb{Z}_+^d$ we set $|\alpha| := \sum_{j\in\mathbb{N}_d} \alpha_j$. Using the multinomial expansion we conclude that the dot product kernel defined in Equation (18) can be expressed in the form

$$K(x,y) := \sum_{\alpha\in\mathbb{Z}_+^d} \phi_\alpha(x)\overline{\phi_\alpha(y)}, \;\; x,y \in \mathbb{C}^d$$

where the features are defined for $\alpha \in \mathbb{Z}_+^d$ at $x \in \mathbb{C}^d$ as

$$\phi_\alpha(x) := \sqrt{a_{|\alpha|}\binom{|\alpha|}{\alpha}} \;\; x^\alpha.$$

As is well-known, for example as a special case of the Stone-Weierstrass approximation theorem (Rudin, 1991, page 122), these features are universal on $\mathbb{C}^d$ and $\mathbb{R}^d$ so the result follows from Theorem 7. ■

The next result we present is a version of the above remark appropriate for the unit ball $\mathbb{B}^d := \{x : (x,x) < 1\}$ in $\mathbb{R}^d$. We have in mind the following fact. Again, we start with the function G defined above in (17) but in the next result we only assume that it is analytic in the unit disc $\Delta := \{z : |z| < 1, z \in \mathbb{C}\}$

Corollary 9 *If G is analytic in Δ and has all positive coefficients then K is universal on $\mathbb{B}^d$.*

The proof is identical to the proof of Corollary 8 and therefore is omitted.

We end our discussion of dot product kernels by considering the case of the unit sphere $\mathbb{S}^d$ in $\mathbb{R}^{d+1}$. To this end, we review the construction of *Schoenberg kernels* on $\mathbb{S}^d$ (Schoenberg, 1942). Let P_k^d, $k \in \mathbb{Z}_+$ be the k-th degree ultraspherical polynomial. When $d = 1$, P_k^1 is the k-th degree Chebyshev polynomial (Rivlin, 1990) and for $d > 1$, P_k^d is determined by the generating function

$$\frac{1}{(1-2zt+z^2)^{(d-1)/2}} = \sum_{k\in\mathbb{Z}_+} P_k^d(t) z^k, \;\; z \in \Delta, \, t \in [-1,1].$$

We assume that we have a sequence of *nonnegative* numbers $\{a_k : k \in \mathbb{Z}_+\}$ such that

$$\sum_{k\in\mathbb{Z}_+} a_k P_k^d(1) < \infty. \tag{19}$$

Let the function $g : [0,\pi] \to \mathbb{R}$ be given at $t \in [0,\pi]$ by the equation

$$g(t) := \sum_{k\in\mathbb{Z}_+} a_k P_k(\cos t). \tag{20}$$

The condition (19) ensures that the series in (20) converges uniformly on $[0,\pi]$, since P_k^d achieves its maximum in absolute value on the interval $[-1,1]$ at 1 (Szegö, 1959, page 166).

The geodesic distance between $x, y \in \mathbb{S}^d$ is given by

$$D_d(x,y) := \arccos(x,y)$$

and Schoenberg proved in Schoenberg (1942) that K is kernel on $\mathbb{S}^d$ if and only if it has this form

$$K(x,y) := g(D_d(x,y)), \quad x,y \in \mathbb{S}^d. \tag{21}$$

Theorem 10 *The kernel given by Equation (21) is universal on $\mathbb{S}^d$ if and only if for all $k \in \mathbb{Z}_+$, a_k is positive.*

Proof We write the kernel in (21) in the feature form. For this purpose, we recall some basic facts about *spherical harmonics* which can be found in Stein and Weiss (1971). Let $\mathcal{H}_k$ be the set of all homogeneous harmonic polynomials of total degree k on $\mathbb{R}^{d+1}$ restricted to $\mathbb{S}^d$ and set $h_k := \dim \mathcal{H}_k$. We view $\mathcal{H}_k$ as a subspace of the $L^2(\mathbb{S}^d, \omega_d)$ where ω_d is the Lebesgue measure on $\mathbb{S}^d$. Let $\{Y_j^k : j \in \mathbb{N}_{h_k}\}$ be an orthonormal basis for $\mathcal{H}_k$ and recall that $\mathcal{H}_k$ is orthogonal to $\mathcal{H}_k'$ if $k \neq k'$ (Stein and Weiss, 1971). For each $k \in \mathbb{Z}_+$, there exists a positive constant c_k such that for all $x, y \in \mathbb{S}^d$

$$P_k((x,y)) = c_k \sum_{j \in \mathbb{N}_{h_k}} Y_j^k(x) Y_j^k(y). \tag{22}$$

Therefore, by Equations (20) and (22), we have that

$$K(x,y) = \sum_{k \in \mathbb{Z}_+} a_k c_k \sum_{j \in \mathbb{N}_{h_k}} Y_j^k(x) Y_j^k(y), \;\; x,y \in \mathbb{S}^d.$$

We let $I := \{(k,j) : k \in \mathbb{Z}_+, j \in \mathbb{N}_{h_k}\}$ and introduce for each $\ell = (k,j) \in I$ the feature

$$\phi_\ell := \sqrt{a_k c_k} Y_j^k.$$

Now, if all the a_k, $k \in \mathbb{Z}_+$ are positive we conclude that $\operatorname{span}\{\phi_\ell : \ell \in I\}$ is the linear space of *all* polynomials and, in particular, is universal. However, if there exists a $m \in \mathbb{Z}_+$ such that $a_m = 0$ then $\operatorname{span}\{\phi_\ell : \ell \in I\}$ is orthogonal to $\mathcal{H}_m$ and hence is not universal. ■

4. Translation Invariant Kernels on $\mathbb{R}^d$

In the remaining part of this paper we shall focus on *translation invariant* kernels on $\mathbb{R}^d$ which have the form

$$K(x,y) = k(x-y), \quad x,y \in \mathbb{R}^d$$

for some function k which is continuous on $\mathbb{R}^d$. Recall that Bochner (1959) proved that K is a kernel if and only if there is a unique finite Borel measure μ on $\mathbb{R}^d$ such that k at any $x \in \mathbb{R}^d$ has the form

$$k(x) := \int_{\mathbb{R}^d} e^{i(x,y)} d\mu(y). \tag{23}$$

We shall study the question of the universality of the kernel K in terms of the properties of its corresponding finite Borel measure μ. We start by identifying the input space as $\mathcal{X} := \mathbb{R}^d$ and then introduce our Hilbert space $\mathcal{W}$ of all complex-valued functions on $\operatorname{supp}(\mu)$ with inner product

$$(f,g)_{\mathcal{W}} := \int_{\mathrm{supp}(\mu)} f(x)\overline{g(x)}d\mu(x).$$

Next, we introduce the feature map $\Phi : \mathbb{R}^d \to \mathcal{W}$ which is defined by setting for each $x, y \in \mathbb{R}^d$

$$\Phi(x)(y) := e^{i(x,y)} \tag{24}$$

so that $K(x,y) = (\Phi(x), \Phi(y))_{\mathcal{W}}$, $x, y \in \mathbb{R}^d$. Note that in this case the Hilbert space $\mathcal{W}$ is the usual $L^2(\mathrm{supp}(\mu), \mu)$ space of all square integrable complex-valued functions relative to the measure μ on $\mathrm{supp}(\mu)$. Since μ is a finite Borel measure every bounded continuous function on $\mathrm{supp}(\mu)$ is contained in $\mathcal{W}$.

Next, we introduce the set of exponentials

$$\mathcal{E}(\mu) := \{\Phi(x) : x \in \mathrm{supp}(\mu)\}.$$

We say $\mathcal{E}(\mu)$ is universal provided that $\mathcal{E}(\mu)$ is dense in $C(\mathcal{Z})$ for every compact subset $\mathcal{Z}$ of $\mathbb{R}^d$.

Lemma 11 *For each compact subset $\mathcal{Z}$ of $\mathbb{R}^d$, $K(\mathcal{Z}) = C(\mathcal{Z})$ if and only if $\overline{\mathrm{span}}\,\mathcal{E}(\mu) = C(\mathcal{Z})$.*

Proof We observe by Fubini's theorem that the map $U : B(\mathcal{Z}) \to \mathcal{W}$ corresponding to Φ in Equation (24) is identified by formula (7) for each $\nu \in B(\mathcal{Z})$ and $y \in \mathrm{supp}(\mu)$ to be

$$U(\nu)(y) := \int_{\mathcal{Z}} e^{i(x,y)} d\nu(x).$$

Hence, we see that $\mathcal{N}(U) = \mathcal{E}(\mu)^{\perp}$ and so U is injective if and only if $\mathrm{span}\,\mathcal{E}(\mu)$ is dense in $C(\mathcal{Z})$. Therefore, the result follows from Proposition 1. ■

As a consequence of Lemma 11, we find that the universality of K depends on the density of the set $\mathcal{E}(\mu)$ of complex exponentials.

Theorem 12 *The translation kernel K is universal if and only if the set of exponential features $\mathcal{E}(\mu)$ is universal.*

An interesting feature of this result is that whenever a translation kernel K is universal with corresponding measure μ then *any* kernel corresponding to *any* other measure ρ with the *same* support as μ is also universal! Another consequence of this result pertains to the integral operator T defined by Equation (4). Specifically, let $\mathcal{Z}$ be a compact subset of $\mathbb{R}^d$, ν a finite Borel measure on $\mathcal{Z}$ such that $\mathrm{supp}(\nu) = \mathcal{Z}$ and T the integral operator defined by (4) then Theorem 6 and Lemma 11 yield the following result.

Corollary 13 *If K is a translation kernel on $\mathbb{R}^d$ then $\overline{\mathcal{R}(T)} = C(\mathcal{Z})$ if and only if $\overline{\mathrm{span}}\,\mathcal{E}(\mu) = C(\mathcal{Z})$.*

We now turn our attention to describing various conditions on the support of the measure μ which ensures the corresponding set of exponential features $\mathcal{E}$ is universal. To this end, we say, as in Micchelli et al. (2003), that a subset $\mathcal{S}$ of $\mathbb{C}^d$ is a *uniqueness* set if an *entire function* on $\mathbb{C}^d$ vanishes on $\mathcal{S}$ then it is everywhere zero on $\mathbb{C}^d$. We recall the following result from Micchelli et al. (2003).

Proposition 14 *If* $\mathrm{supp}\,(\mu)$ *is a uniqueness subset of* $\mathbb{C}^d$ *then the translation kernel K is universal.*

Proof By Theorem 12, it suffices to show that for each compact set $\mathcal{Z}$ of $\mathbb{R}^d$ there does not exist nontrivial $\nu \in B(\mathcal{Z})$ satisfying for each $y \in \mathrm{supp}\,(\mu)$ that

$$\int_{\mathcal{Z}} e^{i(x,y)} d\nu(x) = 0. \tag{25}$$

Suppose there exists $\nu \in B(\mathcal{Z})$ that satisfies (25) for all $y \in \mathrm{supp}\,(\mu)$. Then the entire function F defined for each $z \in \mathbb{C}^d$ as

$$F(z) := \int_{\mathcal{Z}} e^{i(z,x)} d\nu(x)$$

vanishes on $\mathrm{supp}\,(\mu)$. Consequently, F must be everywhere zero and so $\nu = 0$. ■

We note here that the proof above adapts to show that for each finite Borel measure ω on $\mathcal{Z}$ and $p \in [1,\infty)$, $K(\mathcal{Z}) = L^p(\mathcal{Z},\omega)$ if and only if $\overline{\mathrm{span}}\,\mathcal{E}(\mu) = L^p(\mathcal{Z},\omega)$. This fact, together with the remark at the end of Section 2, implies that $\overline{\mathcal{R}(T)} = L^p(\mathcal{Z},\omega)$ if and only if $\overline{\mathrm{span}}\,\mathcal{E}(\mu) = L^p(\mathcal{Z},\omega)$.

Proposition 15 *If* $\mathrm{supp}\,(\mu)$ *has positive Lebesgue measure on* $\mathbb{R}^d$ *then the translation kernel K is universal.*

Proof By Proposition 14, we suffice to point out the well-known fact that the real zeros of any nontrivial entire function on $\mathbb{C}^d$ form a set of Lebesgue measure zero on $\mathbb{R}^d$. ■

By Proposition 15, the uniqueness condition is satisfied by a large class of finite Borel measures on $\mathbb{R}^d$. To elaborate on this point further, we apply the Lebesgue decomposition theorem to μ and write it uniquely as

$$\mu = \mu_c + \mu_s$$

where μ_c is the continuous part of μ (Royden, 1988), in other words, there is a nonnegative function $g \in L^1(\mathbb{R}^d)$, such that for all Borel sets $S \subseteq \mathbb{R}^d$

$$\mu_c(S) = \int_S g(x)dx \tag{26}$$

and μ_s is the singular part of μ so that the Lebesgue measure of its support is zero. Our next result makes use of this decomposition.

Proposition 16 *If the continuous part of* μ *in its Lebesgue decomposition is nonzero then the translation kernel K is universal.*

Proof We only need to show that $\mathrm{supp}\,(\mu)$ has positive Lebesgue measure if the continuous part μ_c of μ in its Lebesgue decomposition is nonzero. Let g be the nonnegative function in $L^1(\mathbb{R}^d)$ that determines μ_c by (26). The hypothesis that $\mu_c \neq 0$ implies $g \neq 0$. Since $\mathrm{supp}\,(\mu_c) = \mathrm{supp}\,(g) \subseteq \mathrm{supp}\,(\mu)$ it follows that $\mathrm{supp}\,(\mu)$ has positive Lebesgue measure. ■

We shall now turn our attention to the Schoenberg kernels on $\mathbb{R}^d \times \mathbb{R}^d$ (Schoenberg, 1938). A continuous function $g : \mathbb{R}_+ \to \mathbb{R}$ determines a *radial* kernel on $\mathbb{R}^d \times \mathbb{R}^d$ by the formula

$$K(x,y) := g(\|x-y\|^2), \;\; x,y \in \mathbb{R}^d \tag{27}$$

where $\|x\| := \sqrt{(x,x)}$ is the usual euclidean norm of $x \in \mathbb{R}^d$. It was proved in Schoenberg (1938) that K is a kernel on $\mathbb{R}^d \times \mathbb{R}^d$ for *all* $d \in \mathbb{N}$ if and only if there exists a finite Borel measure μ on $\mathbb{R}_+$ such that for all $t \in \mathbb{R}_+$

$$g(t) := \int_{\mathbb{R}_+} e^{-t\sigma} d\mu(\sigma). \tag{28}$$

All kernels of this type are *not* universal. Indeed, the choice of a measure concentrated only at $\sigma = 0$ gives a kernel K that is identically constant and therefore it is not universal. This is the only exceptional case as we shall explain in the next result.

Theorem 17 *If the measure μ in Equation (28) is not concentrated at zero then the radial kernel K in (27) is universal.*

Proof We first show how to prove the result using Proposition 16 when the measure μ has the additional property that for some $a > 0$ its support is contained in the ray $[a,\infty)$. In that case, we use the formula

$$e^{-\sigma\|x\|^2} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \left(\frac{\pi}{\sigma}\right)^{d/2} e^{i(x,\xi)} e^{-\frac{\|\xi\|^2}{4\sigma}} d\xi$$

valid for all $x \in \mathbb{R}^d$ and $\sigma > 0$. Using Fubini's theorem, we express the function k in (23) for the kernel K in (27) at $x \in \mathbb{R}^d$ as

$$k(x) := \int_{\mathbb{R}^d} e^{i(x,y)} f(y) dy$$

where the function f is defined at $y \in \mathbb{R}^d$ by the equation

$$f(y) := \frac{1}{(2\pi)^d} \int_a^\infty \left(\frac{\pi}{\sigma}\right)^{d/2} e^{-\frac{\|y\|^2}{4\sigma}} d\mu(\sigma).$$

The function f is strictly positive and

$$\int_{\mathbb{R}^d} f(y) dy = \int_a^\infty d\mu(\sigma) > 0,$$

so the theorem is a consequence of Proposition 16. If the support of the measure is *not* a subset of the open ray $\mathbb{R}_+$ we proceed differently. A direct computation using the power series for the exponential function and the multinomial expansion yields the formula

$$K(x,y) = \sum_{\alpha \in \mathbb{Z}_+^d} \binom{|\alpha|}{\alpha} \frac{2^{|\alpha|}}{|\alpha|!} \int_{\mathrm{supp}\,(\mu)} \sigma^{|\alpha|} x^\alpha e^{-\sigma\|x\|^2} y^\alpha e^{-\sigma\|y\|^2} d\mu(\sigma).$$

This suggests that we introduce the Hilbert space $\mathcal{W}$ of real-valued functions on the set $\mathbb{M} := \mathbb{Z}_+^d \times \mathrm{supp}\,(\mu)$ with inner product

$$(F,G)_{\mathcal{W}} := \sum_{\alpha \in \mathbb{Z}_+^d} \binom{|\alpha|}{\alpha} \frac{2^{|\alpha|}}{|\alpha|!} \int_{\mathrm{supp}\,(\mu)} F(\alpha,\sigma) G(\alpha,\sigma) d\mu(\sigma)$$

and a feature map $\Phi : \mathbb{R}^d \to \mathcal{W}$ defined at $(\alpha,\sigma) \in \mathbb{M}$ and $x \in \mathbb{R}^d$ as

$$\Phi(x)(\alpha,\sigma) := \sigma^{|\alpha|/2} x^\alpha e^{-\sigma\|x\|^2}.$$

Hence we have the feature space representation for the kernel K given for each $x,y \in \mathbb{R}^d$ as

$$K(x,y) = (\Phi(x),\Phi(y))_{\mathcal{W}}.$$

Now, we let $\mathcal{Z}$ be some prescribed compact subset of $\mathbb{R}^d$. As in the proof of Theorem 11 we identify the operator $U : B(\mathcal{Z}) \to \mathcal{W}$ in (7) at $\nu \in B(\mathcal{Z})$ and $(\alpha,\sigma) \in \mathbb{M}$ as

$$U(\nu)(\alpha,\sigma) = \int_{\mathcal{Z}} \sigma^{|\alpha|/2} x^\alpha e^{-\sigma\|x\|^2} d\nu(x).$$

Therefore, if there is a *positive* $\rho \in \operatorname{supp}(\mu)$ and $\nu \in \mathcal{N}(U)$ then for any $\alpha \in \mathbb{Z}_+^d$ we have that

$$\int_{\mathcal{Z}} x^\alpha e^{-\rho\|x\|^2} d\nu(x) = 0.$$

This implies, by the density of all polynomials in $C(\mathcal{Z})$, that $\nu = 0$. In other words, U is injective and so the result follows from Proposition 1. ■

As a consequence of Theorem 17 we conclude that the following two classes of kernels are universal:

$$K(x,y) := e^{-\alpha\|x-y\|^2}, \;\; x,y \in \mathbb{R}^d$$

and

$$K(x,y) := (\beta + \|x-y\|^2)^{-\alpha}, \;\; x,y \in \mathbb{R}^d$$

where α and β are arbitrary positive numbers.

Next, we give a quite different condition on the support of the measure μ so that the corresponding translation kernel is universal. For this discussion we shall use the celebrated Stone-Weierstrass theorem (Rudin, 1991).

Proposition 18 *If* $\operatorname{supp}(\mu)$ *is a subgroup of* $\mathbb{R}^d$ *such that for each* $x \in R^d \setminus \{0\}$ *the set* $\{(x,y) : y \in \operatorname{supp}(\mu)\} \not\subseteq \mathbb{Z}$ *then K is universal.*

Proof By Theorem 12, it suffices to show that $\operatorname{span}\mathcal{E}(\mu)$ is dense in $C(\mathcal{Z})$. Suppose all the hypotheses are satisfied and there exists some compact set $\mathcal{Z} \subseteq \mathbb{R}^d$ such that $K(\mathcal{Z})$ is not dense in $C(\mathcal{Z})$. Since $\operatorname{supp}(\mu)$ is a subgroup of $\mathbb{R}^d$ we see that $1 \in \operatorname{span}\mathcal{E}(\mu)$ and that for all $f,g \in \operatorname{span}\mathcal{E}(\mu)$, both fg and $\overline{f}$ belong to $\operatorname{span}\mathcal{E}(\mu)$. Therefore, by the Stone-Weierstrass theorem, there exist distinct points $x_1,x_2 \in \mathcal{Z}$ such that for all $f \in \operatorname{span}\mathcal{E}(\mu)$, $f(x_1) = f(x_2)$. That is, for each $y \in \operatorname{supp}(\mu)$ $e^{i(x_1-x_2,y)} = 1$ or in other words, $(x_1-x_2,y)/2\pi \in \mathbb{Z}$. This contradiction proves the proposition. ■

Corollary 19 *If* $d = 1$, $\operatorname{supp}(\mu)$ *is a subgroup of* $\mathbb{R}$ *and there exists* $y_1,y_2 \in \operatorname{supp}(\mu) \setminus \{0\}$ *such that* y_1/y_2 *is an irrational number then K is universal.*

Proof By the hypotheses of the corollary, it is clear that there does not exist $x \in \mathbb{R} \setminus \{0\}$ such that both xy_1 and xy_2 are integers. The result follows immediately from Proposition 18. ■

5. Conclusion

We have provided a variety of conditions for a kernel to be universal in terms of properties of its features. Several examples of universal dot product kernels are given. In the case of translation kernels we showed that universality depends on the density of a set of complex exponentials. This problem has attracted much interest in the literature. An extensive survey of existing results on the univariate case is given in Redheffer (1977) and additional information in Beurling and Malliavin (1967). With this available information a complete characterization of univariate translation kernels follows. We show that except in rare circumstances all Schoenberg radial kernels are universal.

Our study indicates that there is intimate relationship between uniformly approximating a prescribed target function by a kernel and approximating by its features. There is an important problem which is not treated here that deserves careful attention. Given a prescribed error $\varepsilon > 0$ and a prescribed target function f, what is the relationship between the number of features needed to represent f with error ε and the number of kernel sections needed for the same purpose. We intend to address this issue on another occasion.

Acknowledgments

The authors are indebted to one of the referees for bringing to their attention the relationship of Corollaries 3.2 and 3.3 with results in references Steinwart (2001) and Zhou (2003). This work was supported by the US National Science of Foundation under grant CCR-0407476, by the Natural Science Foundation of China under grant 10371122 and by the Ministry of Education of the People's Republic of China under the Changjiang Scholar Chair Professorship program. The corresponding author is Yuesheng Xu. He is also with the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, P. R. China.

References

A. Argyriou, C. A. Micchelli and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. *Proceeding of the 18th Annual Conference on Learning Theory (COLT'05)*, Bertinoro, Italy, 2005.

A. Argyriou, R. Hauser, C. A. Micchelli and M. Pontil. A DC-programming algorithm for kernel selection. *Proceeding of the 23rd International Conference on Machine Learning (ICML'06)*, forthcoming (see also Research Note RN/06/04, Department of Computer Science, UCL, 2006).

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68: 337–404, 1950.

F. R. Bach, G. R. G. Lanckriet and M. I. Jordan. Multiple kernel learning, conic duality and the SMO algorithm. *Proceeding of the 21st International Conference on Machine learning (ICML'04)*, 2004.

A. Beurling and P. Malliavin. On the closure of characters and the zeros of entire functions. *Acta. Math.*, 118: 79–93, 1967.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

S. Bochner. *Lectures on Fourier Integrals With an author's supplement on monotonic functions, Stieltjes integrals, and harmonic analysis*. Annals of Mathematics Studies, no. 42, Princeton University Press, New Jersey, 1959.

T. Evgeniou, M. Pontil and T. Poggio. Regularization networks and support vector machines. *Adv. Comput. Math.*, 13: 1–50, 2000.

C. H. FitzGerald, C. A. Micchelli and A. Pinkus. Functions that preserve families of positive semidefinite matrices. *Linear Algebra Appl.*, 221: 83–102, 1995.

T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El-Ghaoui and M. I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5: 27–72, 2004.

P. Lax. *Functional Analysis*. Wiley, New York, 2002.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Royal Soc. London*, 209: 415–446, 1909.

C. A. Micchelli and M. Pontil. A function representation for learning in Banach spaces. *Proceeding of the 17th Annual Conference on Learning (COLT'04)*, 2004.

C. A. Micchelli and M. Pontil. Feature space perspectives for learning the kernel. *Machine Learning*, forthcoming (see also: Research Note RN/05/11, Department of Computer Science, UCL, June, 2005).

C. A. Micchelli, M. Pontil, Q. Wu and D. X. Zhou. Error bounds for learning the kernel. Research Note RN/05/04, Department of Computer Science, UCL, 2006.

C. A. Micchelli, Y. Xu and P. Ye. Cucker Smale learning theory in Besov spaces. *Advances in Learning Theory: Methods, Models and Applications*. J. Suykens, G. Horvath, S. Basu, C. A. Micchelli and J. Vandewalle, editors. IOS Press, Amsterdam, The Netherlands, 2003, 47–68.

J. Neumann, C. Schnörr and G. Steidl. SVM-based feature selection by direct objective minimization. C.E. Rasmussen, H. H. Bülthoff, B. Schölkopf and M. A. Giese, editors. Lecture Notes in Computer Science, 3175: 212–219, *Proceeding of the 26th DAGM Symposium*, 2004.

C. S. Ong, A. J. Smola and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6: 1043–1071, 2005.

T. Poggio, S. Mukherjee, R. Rifkin, A. Raklin and A. Verri. B. *Uncertainty in geometric computations*, J. Winkler and M. Niranjan, editors. Kluwer Academic Publishers, 22: 131–141, 2002.

R. M. Redheffer. Completeness of sets of complex exponentials. *Adv. Math.*, 24: 1–62, 1977.

T. J. Rivlin. *Chebyshev Polynomials*. 2nd Edition, John Wiley, New York, 1990.

H. Royden. *Real Analysis*. 3rd Edition, Macmillan Publishing Company, New York, 1988.

W. Rudin. *Functional Analysis*. 2nd Edition, McGraw Hill, New York, 1991.

I. J. Schoenberg. Metric spaces and completely monotone functions. *Ann. of Math. (2)*, 39: 811–841, 1938.

I. J. Schoenberg. Positive definite functions on spheres. *Duke. Math. J.*, 9: 96–108, 1942.

B. Schölkopf, C. J. C. Burges and A. Smola. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, Mass, 1999.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, Mass, 2002.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.

S. Sonnenburg, G. Rätsch and C. Schäfer. A general and efficient multiple kernel learning algorithm. Y. Weiss, B. Schölkopf and J. Platt, editors. *Advances in Neural Information Processing Systems*, 18. MIT Press, Cambridge, Mass, 2006.

E. M. Stein and G. Weiss. *Introduction to Fourier Analysis on Euclidean Spaces*. Princeton University Press, New Jersey, 1971.

I. Steinwart. On the influence of kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2: 67–93, 2001.

H. Sun. Mercer theorem for RKHS on noncompact sets. *J. Complexity*, 21: 337–349, 2005.

G. Szegö. *Orthogonal Polynomials*. American Mathematical Society Colloquium Publications 23. Revised Edition, Providence, RI, 1959.

G. Wahba. *Splines Models for Observational Data*. Series in Applied Mathematics 59. SIAM, Philadelphia, 1990.

D. X. Zhou. Density problem and approximation error in learning theory. *preprint*, 2003.

Journal of Machine Learning Research 7 (2006) 2669-2672 Submitted 12/06; Published 12/06

Machine Learning for Computer Security*

Philip K. Chan PKC@CS.FIT.EDU
Department of Computer Sciences
Florida Institute of Technology
Melbourne, FL 32901, USA

Richard P. Lippmann LIPPMANN@LL.MIT.EDU
Lincoln Lab, MIT
244 Wood Street
Lexington, MA 02173, USA

Editors: Philip K. Chan and Richard P. Lippman

Abstract

The prevalent use of computers and internet has enhanced the quality of life for many people, but it has also attracted undesired attempts to undermine these systems. This special topic contains several research studies on how machine learning algorithms can help improve the security of computer systems.

Keywords: computer security, spam, images with embedded text, malicious executables, network protocols, encrypted traffic

1. Introduction

As computers have become more ubiquitous and connected, their security has become a major concern. Attacks are more pervasive and diverse—they range from unsolicited email messages that can trick users in providing personal information to dangerous viruses that can erase data and shut down computer systems. Consequently, security breaches are not rare topics in the news.

Conventional security software requires a lot of human effort to identity threats, extract characteristics from the threats, and encode the characteristics into software to detect the threats. This labor-intensive process can be more efficient by applying machine learning algorithms. As a result, a number of researchers have investigated various machine learning algorithms to detect attacks more efficiently and reliably. Two edited books (Barbara and Jajodia, 2002; Maloof, 2006) have been published and two workshops at research conferences (Chan et al., 2003; Brodley et al., 2004) have been conducted in recent years. Due to the level of interest from the researchers and maturity of some of their studies, we decided to organize a special topic on "Machine Learning for Computer Security" for this journal.

*. This work is sponsored by the U.S. Air Force under Air Force Contract FA 8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Figure 1: Adversarial spam image designed to defeat OCR text extraction

2. JMLR Special Topic

We received nineteen submissions for this special topic. After considering the reviews for each submission, we selected four papers to be included in this special topic.

Bratko et al. (2006) describe a recent advance in the ongoing battle between those that generate and those that want to block unwanted spam email. They apply adaptive statistical compression algorithms (Dynamic Markov Compression (DMC) and Prediction by Partial Matching (PPM)) to build models for email messages. DMC learns a Markov model incrementally via a cloning technique to introduce new states in the model. PPM learns a table of contexts and the frequency of the symbol following the contexts. To classify if a message is spam, they use minimum cross entropy (MCE) and minimal description length (MDL). MCE calculates the number of bits to encode a message based on competing models learned from normal and spam messages, and classifies the message to the class whose model requires fewer encoded bits. MDL measures the additional number of bits needed to encode a message after adding it to the competing models, and classifies the message to the class whose model needs fewer additional bits. The authors evaluated their techniques on three datasets and against six open source spam filters. For both DMC and PPM models, they found that MDL yields lower 1-AUC (1 - Area under ROC) than MCE. They also reported that DMC consistently outperforms the six open source spam filters.

Similar to Bratko et al. (2006), Fumera et al. (2006) tackle the problem of spam email, however, they consider spam messages with embedded images. They developed an approach to analyze spam email when spam text messages are embedded in attached images instead of in the text email body (for example, Figure 1). Standard optical character recognition (OCR) software is used to extract words embedded in images and these extra words are used in addition to text in the email header and body to improve performance of a support vector machine spam classifier. At a false alarm rate of 1%, this technique often reduced the miss rate by a factor of two for spam email that contained embedded images. Evidently, this approach has been adopted by commercial spam filtering companies. Spammers have reacted by adding varied background and distorting text embedded in images

to make it difficult for OCR systems to extract spam messages but easy for humans to interpret these messages. Figure 1 shows an example of an image from a recent spam email suggesting a stock to purchase. This paper illustrates that pattern classification techniques can be effective for complex problems such as spam, but that it can be difficult to obtain a long-standing advantage in adversarial environments.

Instead of email messages, Kolter and Maloof (2006) analyze executables. They demonstrate that N-gram analysis of executables can be used to distinguish between normal computer programs and malicious virus, worm, and Trojan horse programs. Even though roughly 20% of the malicious software samples used were obfuscated with either compression or encryption, detection accuracy for 291 previously unseen malicious executables was roughly 98% correct at a false alarm rate of 5%. These good results were made possible by collecting and carefully confirming and labeling a training corpus of 1971 benign and 1651 malicious executables and using 10-fold cross-validation to select both the top-performing N-grams and the best performing classifier which in this case was a boosted tree classifier.

As network traffic is increasingly encrypted, Wright et al. (2006) address the problem of inferring application protocol behaviors in encrypted traffic to help intrusion detection systems. The authors first propose using k-nearest neighbor methods for identifying protocols in data instances, each of which is known to belong to one protocol. Experiments on eight protocols indicate 75-100% true detection rate. They then propose Hidden Markov Models (HMMs) for identifying protocols in traffic with *mixed* protocols. Each protocol has an HMM model. Each model has a group of states (a pair of client and server states, and a pair of insert and delete states) and the number of groups is equal to the average number of packets in a connection for the protocol. The emitting symbols are codewords for packet size and inter-arrival time. To classify, they pick the protocol, whose model has the best Viterbi path that explains the observation. Their empirical results indicate that HMMs can achieve 58-87% true detection rate on eight protocols. They last propose methods for identifying the number of connections in encrypted *tunnels*. They assume the number of connections is Gaussian and the number of packets of a certain type is Poisson. Each HMM state corresponds to a connection count, the output is a tuple of counts for the different types of packets. They use the Gaussian and Poisson assumptions to estimate standard deviations of the number of connections and rates of each packet type. The number of connections at a certain time is predicted by the most probable state at that time. They evaluated their techniques on four tunnels.

3. Concluding Remarks

These four papers demonstrate the need for carefully constructed training and test corpora, effective feature extraction and selection, and valid evaluations on representative corpora when applying pattern classification to computer security problems. They also suggest a new important direction for pattern classification research. This is to develop approaches that provide sustained good performance in adversarial environments where a malicious adversary takes actions to subvert a classifier. Some of these actions could be: (1) obscure important discriminating input features, for example by modifying text in images to be difficult for an OCR to extract, (2) add extraneous additional features to make an input appear more normal, for example by adding sentences extracted from normal emails to the end of spam emails, (3) alter the prior probabilities of abnormal inputs, and (4) take all of these actions over time in a way designed to thwart systems that learn and adapt over time. The paper on spam detection (Fumera et al., 2006) mentions this problem and another recent paper

(Newsome et al., 2006) shows how an adversary can defeat a system that learns to automatically extract signatures to detect computer worms. Further research is needed to determine if there are any systematic approaches that can lead to classifiers that are more robust in adversarial environments.

Acknowledgments

We would like to acknowledge the time and effort of the reviewers: Wei Fan (IBM Watson Research Center), Anup Ghosh (DARPA), Tom Goldring (NSA), Sushil Jajodia (George Mason University), Chris Kruegel (Technical University Vienna), Vipin Kumar (University of Minnesota), Terran Lane (University of New Mexico), Wenke Lee (Georgia Institute of Technology), Matthew Mahoney (Florida Institute of Technology), Roy Maxion (Carnegie Mellon University), Chris Michael (Cigital), Srinivasan Parthasarathy (Ohio State University), R. Sekar (Stony Brook University), Jude Shavlik (University of Wisconsin), Marius Silaghi (Florida Institute of Technology), Salvatore Stolfo (Columbia University), and Alfonso Valdes (SRI). Their diligence makes this special topic a reality.

References

D. Barbara and S. Jajodia, editors. *Applications of Data Mining in Computer Security*. Kluwer, 2002.

A. Bratko, B. Filipic, G. Cormack, T. Lynam, and B. Zupan. Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 7:2673–2698, 2006.

C. Brodley, P. Chan, R. Lippmann, and B. Yurcik, editors. *Workshop Notes of Visualization and Data Mining for Computer Security*, 2004. ACM Intl. Conf. Computer and Communications Security (CCS).

P. Chan, V. Kumar, W. Lee, and S. Parthasarathy, editors. *Workshop Notes of Data Mining for Computer Security*, 2003. IEEE Intl. Conf. Data Mining (ICDM).

G. Fumera, I. Pillai, and F. Roli. Spam filtering based on the analysis of text information embedded into images. *Journal of Machine Learning Research*, 7:2699–2720, 2006.

J. Kolter and M. Maloof. Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 7:2721–2744, 2006.

M. Maloof, editor. *Machine Learning and Data Mining for Computer Security*. Springer, 2006.

J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In D. Zamboni and C. Kruegel, editors, *Recent Advances in Intrusion Detection (RAID) 2006 (LNCS 4219)*, pages 81–105, Berlin, 2006. Springer-Verlag.

C. Wright, F. Monrose, and G. Masson. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research*, 7:2745–2769, 2006.

Journal of Machine Learning Research 7 (2006) 2673-2698 Submitted 03/06; Revised 09/06; Published 12/06

Spam Filtering Using Statistical Data Compression Models

Andrej Bratko ANDREJ.BRATKO@IJS.SI
Department of Intelligent Systems
Jožef Stefan Institute
Jamova 39, Ljubljana, Slovenia SI-1000

Gordon V. Cormack GVCORMACK@UWATERLOO.CA
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

Bogdan Filipič BOGDAN.FILIPIC@IJS.SI
Department of Intelligent Systems
Jožef Stefan Institute
Jamova 39, Ljubljana, Slovenia SI-1000

Thomas R. Lynam TRLYNAM@PLG.UWATERLOO.CA
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

Blaž Zupan BLAZ.ZUPAN@FRI.UNI-LJ.SI
Faculty of Computer and Information Science
University of Ljubljana
Tržaška 25, Ljubljana, Slovenia SI-1000

Editor: Philip Chan

Abstract

Spam filtering poses a special problem in text categorization, of which the defining characteristic is that filters face an active adversary, which constantly attempts to evade filtering. Since spam evolves continuously and most practical applications are based on online user feedback, the task calls for fast, incremental and robust learning algorithms. In this paper, we investigate a novel approach to spam filtering based on adaptive statistical data compression models. The nature of these models allows them to be employed as probabilistic text classifiers based on character-level or binary sequences. By modeling messages as sequences, tokenization and other error-prone preprocessing steps are omitted altogether, resulting in a method that is very robust. The models are also fast to construct and incrementally updateable. We evaluate the filtering performance of two different compression algorithms; dynamic Markov compression and prediction by partial matching. The results of our empirical evaluation indicate that compression models outperform currently established spam filters, as well as a number of methods proposed in previous studies.

Keywords: text categorization, spam filtering, Markov models, dynamic Markov compression, prediction by partial matching

1. Introduction

Electronic mail is arguably the "*killer app*" of the internet. It is used daily by millions of people to communicate around the globe and is a mission-critical application for many businesses. Over the last decade, unsolicited bulk email has become a major problem for email users. An overwhelming amount of spam is flowing into users' mailboxes daily. In 2004, an estimated 62% of all email was attributed to spam, according to the anti-spam outfit Brightmail.[1] Not only is spam frustrating for most email users, it strains the IT infrastructure of organizations and costs businesses billions of dollars in lost productivity. In recent years, spam has evolved from an annoyance into a serious security threat, and is now a prime medium for phishing of sensitive information, as well the spread of malicious software.

Many different approaches for fighting spam have been proposed, ranging from various sender authentication protocols to charging senders indiscriminately, in money or computational resources (Goodman et al., 2005). A promising approach is the use of content-based filters, capable of discerning spam and legitimate email messages automatically. Machine learning methods are particularly attractive for this task, since they are capable of adapting to the evolving characteristics of spam, and data is often available for training such models. Nevertheless, spam filtering poses a special problem for automated text categorization, of which the defining characteristic is that filters face an active adversary, which constantly attempts to evade filtering. Unlike most text categorization tasks, the cost of misclassification is heavily skewed: Labeling a legitimate email as spam, usually referred to as a *false positive*, carries a much greater penalty than vice-versa. Since spam evolves continuously and most practical applications are based on online user feedback, the task calls for fast, incremental and robust learning algorithms.

In this paper, we consider the use of adaptive data compression models for spam filtering. Specifically, we employ the dynamic Markov compression (Cormack and Horspool, 1987) and prediction by partial matching (Cleary and Witten, 1984) algorithms. Classification is done by first building two compression models from the training corpus, one from examples of spam and one from legitimate email. The compression rate achieved using these two models on the target message determines the classification outcome. Two variants of the method with different theoretical underpinnings are evaluated. The first variant (Frank et al., 2000; Teahan, 2000) estimates the probability of a document using compression models derived from the training data, and assigns the class label based on the model that deems the target document most probable. The second variant, which is introduced in this paper, selects the class for which the addition of the target document results in a minimal increase in the description length of the entire data set. The difference in practice is that the models are used adaptively in this second approach, that is, they are updated at each position in the target document so that statistics from the initial part of the document are taken into account when examining the remainder. In this way, repeated occurrences of similar, "*already seen*", patterns, have diminishing influence on the classification outcome, thereby putting greater weight on heterogeneous evidence.

While the idea of using data compression algorithms for text categorization is not new, we are aware of no existing research that considers such methods for spam filtering. In the present paper, we demonstrate that compression models are extremely well suited to the spam filtering problem. We propose a simple, yet effective modification of the original method, which substantially improves

1. `http://brightmail.com/`, 2004-03-12

filtering performance in our experiments. We generalize the results to compression algorithms not considered in other studies, showing that they exhibit similar, strong performance.

In online learning experiments on three large email collections, we find that compression models generally outperform established spam filters according to several measures. On the TREC public corpus, currently the largest publicly available spam data set, spam misclassification at a false positive rate of 1 in 1000 is 1.2% – 1.8% (depending on the compression algorithm), compared to 2.6% – 6.9% achieved by six reference systems. We also conducted cross validation experiments on the Ling-Spam, PU1 and PU3 data sets, in which compression models compare favorably to a variety of methods considered in previous studies on the same data. Finally, we show that compression models are robust to the type of noise introduced in text by obfuscation tactics which are commonly used by spammers against tokenization-based filters.

2. Related Work

The basic idea of using data compression in classification and other machine learning tasks has been reinvented many times. The intuition arises from the principal observation that compact representations of objects are only possible after some recurring patterns or statistical regularities are detected. This has motivated many applications of general-purpose data compression algorithms in machine learning and data mining problems, in which data compression algorithms are most often used to produce a distance or (dis)similarity measure between pairs of data points (Li et al., 2003; Keogh et al., 2004; Sculley and Brodley, 2006).

Most relevant to our study is the work of Frank et al. (2000), who first proposed the use of compression models for automated text categorization. They investigate using the prediction by partial matching (PPM) algorithm as a Bayesian text classifier, that is, they build a PPM model of each class and use these models to estimate the class-conditional probabilities of the target document. They find that the compression-based method is inferior to support vector machines (SVM) and roughly on par with naive Bayes on the classical Reuters-21578 data set. The same method was investigated by Teahan (2000), and later applied to a number of text categorization problems, such as authorship attribution, dialect identification and genre classification (Teahan and Harper, 2003). They find the compression-based method particularly suitable for dialect identification and authorship attribution, and report fair performance for genre classification and topic detection.

Peng et al. (2004) propose augmenting a word-based naive Bayes classifier with statistical language models to account for word dependencies. They also consider training the language models on characters instead of words, resulting in a method that is effectively very similar to the compression-based method of Frank et al. (2000). In experiments on a number of categorization tasks, they find that the character-based method often outperforms the word-based approach. They also find that character-level language models reach or improve previously published results on four of the six classification tasks considered in the study.

In most spam filtering work, text is modeled with the bag-of-words (BOW) representation, even though it is widely accepted that tokenization is a vulnerability of keyword-based spam filters. Some filters use character n-grams instead of word tokens and apply standard machine learning algorithms on the resulting feature vector (Goodman et al., 2005).

Two particular approaches (that we are aware of) go a step further towards operating directly on character sequences. IBM's Chung-Kwei system (Rigoutsos and Huynh, 2004) uses pattern matching techniques originally developed for DNA sequences. Messages are filtered based on the

number of occurrences of patterns associated with spam and the extent to which they cover the target document. Recently, Pampapathi et al. (2006) proposed a filtering technique based on the suffix tree data structure. They investigate a number of ad hoc scoring functions on matches found in the target document against suffix trees constructed from spam and legitimate email. However, the techniques proposed in these studies are different to the statistical data compression models that are evaluated here. In particular, while these systems use character-based features in combination with some ad hoc scoring functions, compression models were designed from the ground up for the specific purpose of *probabilistic* modeling of sequential data. This property of data compression models allows them to be employed in an intuitively appealing and principled way.

The methods presented in this paper were first evaluated on a number of real-world email collections in the framework of the 2005 Text REtrieval Conference (TREC). The results of this evaluation showed promise in the use of statistical data compression models for spam filtering (Bratko and Filipič, 2005). This article describes the methods used at TREC in greater detail and extends the TREC paper in a number of ways: We compare the use of adaptive and static models for classification, extend our analysis to compression algorithms not considered in previous work, compare the performance of compression models with results published in other studies, and evaluate the effect of noise in text on the filtering performance of compression models and tokenization-based filters.

3. Statistical Data Compression

Probability plays a central role in data compression: Knowing the exact probability distribution governing an information source allows us to construct optimal or near-optimal codes for messages produced by the source. A statistical data compression algorithm exploits this relationship by building a statistical model of the information source, which can be used to estimate the probability of each possible message. This model is coupled with an encoder that uses these probability estimates to construct the final binary representation. For our purposes, the encoding problem is irrelevant. We therefore focus on the source modeling task.

3.1 Preliminaries

We denote by X the random variable associated with the source, which may take the value of any message the source is capable of producing, and by P the probability distribution over the values of X with the corresponding probability mass function p. We are particularly interested in modeling of text generating sources. Each message $\mathbf{x}$ produced by such a source is naturally represented as a sequence $\mathbf{x} = x_1^n = x_1 \ldots x_n \in \Sigma^*$ of symbols over the source alphabet Σ. The length $|\mathbf{x}|$ of a sequence can be arbitrary. For text generating sources, it is common to interpret a symbol as a single character, but other schemes are possible, such as binary (bitwise) or word-level models.

The entropy $H(X)$ of a source X gives a lower bound on the average per-symbol code length required to encode a message without loss of information:

$$H(X) = E_{\mathbf{x}\sim P}\left(-\frac{1}{|\mathbf{x}|}\log p(\mathbf{x})\right).$$

This bound is achievable *only* when the true probability distribution P governing the source is known. In this case, an average message could be encoded using no less than $H(X)$ bits per symbol. However, the true distribution over all possible messages is typically unknown. The goal of any statistical data compression algorithm is then to infer a probability mass function over sequences

$f : \Sigma^* \to [0,1]$, which matches the true distribution of the source as accurately as possible. Ideally[2], a sequence $\mathbf{x}$ is then encoded with $L(\mathbf{x})$ bits, where

$$L(\mathbf{x}) = -\log f(\mathbf{x}).$$

The compression algorithm must therefore *learn* an approximation of P in order to encode messages efficiently. A better approximation will, on average, lead to shorter code lengths. This simple observation alone gives compelling motivation for the use of compression algorithms in text categorization, but we defer this discussion until Section 5, and first describe the compression models used in the study.

3.2 Finite Memory Markov Sources

To make the inference problem over all (possibly infinite) sequences tractable, sources are usually modeled as stationary and ergodic Markov sources[3] with limited memory k. Each symbol in a message $\mathbf{x}$ is therefore assumed to be independent of all but the preceding k symbols (the *context*):

$$p(\mathbf{x}) = \prod_{i=1}^{|\mathbf{x}|} p(x_i|x_1^{i-1}) \approx \prod_{i=1}^{|\mathbf{x}|} p(x_i|x_{i-k}^{i-1}).$$

We assume that a string of k leading symbols that otherwise cannot occur in any sequence is prepended to $\mathbf{x}$ to overcome the technical difficulty in estimating the first k symbols. In practice, a compression algorithm would normally use a shorter context for prediction here.

The number of context symbols k is referred to as the *order* of the Markov model. Higher order models have the potential to better approximate the characteristics of a complex source. However, since the number of possible contexts increases exponentially with context length, accurate parameter estimates are hard to obtain. For example, an order-k model requires $\Sigma^k(\Sigma-1)$ parameters. To tackle this problem, different strategies are employed by different algorithms. The common ground to all such algorithms is that the complexity of the model is increased only after the amount of training data is sufficient to support a more complex model. We describe two particular algorithms that were also used in our experiments later in this section.

3.3 Two-part vs. Adaptive Coding

Two-part data compression methods first transmit the model which is used for encoding, followed by the encoded data. The decoder reads the model first and then uses this information to decode the remaining part of the message. Such methods require two passes over the data. The first pass is required to train the model and the second pass is required for the encoding.

Adaptive methods do not explicitly include the model in the encoded message. Rather, they start encoding the message using an empty model, for example, a uniform distribution over all symbols. The model is incrementally updated after each symbol, gradually adapting to an ever

2. The "ideal" code length ignores the practical requirement that codes have an integer length.
3. The stationarity and ergodicity properties are necessary preconditions for learning to stabilize arbitrarily close to (the best approximation of) the data generating process asymptotically, that is, as the length of the training sequence increases without bound. They are stated here for completeness and are usually taken for granted in a typical machine learning setting, where we expect to learn from past observations.

closer approximation of the data generating process. The probability assigned to a sequence **x** by an order-*k* adaptive algorithm is then

$$f(\mathbf{x}) = \prod_{i=1}^{|\mathbf{x}|} f(x_i | x_{i-k}^{i-1}, M(x_1^{i-1})) \tag{1}$$

where $M(x_1^{i-1})$ denotes the current model at time *i*, constructed from the input sequence x_1^{i-1}. The decoder repeats the learning process, building its own version of the model as the message is decoded. It is important to note that adaptive methods require only a single pass over the data, a property that we will turn to our advantage in subsequent developments.

3.4 Algorithms

In this section, we describe the dynamic Markov compression and prediction by partial matching compression algorithms with which we obtain our main results. Both of the algorithms are adaptive, that is, the model used for prediction may be updated efficiently after each symbol in a sequence.

3.4.1 DYNAMIC MARKOV COMPRESSION

The dynamic Markov compression (DMC) algorithm (Cormack and Horspool, 1987) models an information source with a finite state machine (FSM). A probability distribution over symbols is associated with each state and is used to predict the next binary digit emitted by the source. The algorithm begins in a predefined initial state and moves to the next state after each digit in the sequence. An example FSM structure, corresponding to an order-1 binary Markov model, is shown in the left side of Figure 1. Each state *S* in the FSM has two outbound transitions, one for each binary symbol. These transitions are equipped with frequency counts, from which next-symbol probabilities for the state *S* are calculated (as relative frequencies).

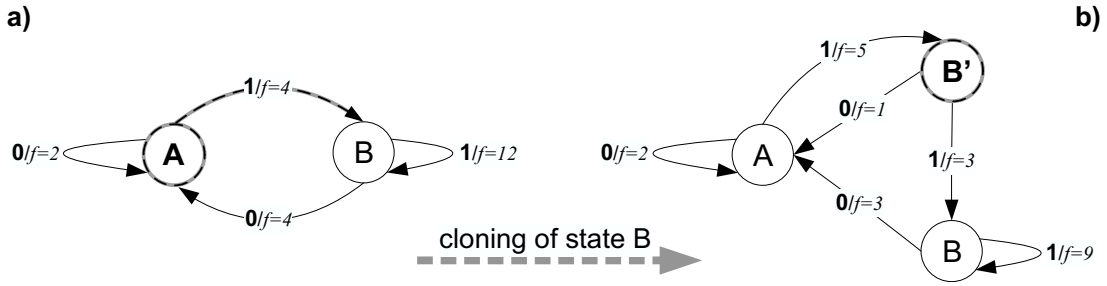


Figure 1: An example of DMC's state cloning operation. The active state and transition at time a) and b) are highlighted. The left hand side shows the model when state *A* is active and the observed symbol is '1'. This triggers the cloning of state *B* and a state transition to the new state *B*′, as shown on the right hand side of the figure. The transition frequencies (visit counts) before and after the cloning operation are also shown.

In an adaptive DMC algorithm, the frequency counts of transitions are incremented whenever a transition fires (once after each symbol in the sequence). The structure of the state machine may also be built incrementally, by using a special state *cloning* operation. Specifically, as soon as the algorithm finds that a transition from some state *A* to some other state *B* in the FSM is used often,

the target state of the transition is considered for cloning. Figure 1 depicts this cloning operation, in which a new state B' is spawned. The new state B' has a single inbound transition, which replaces the former transition from A to B. All outbound transitions of B are copied to B'.

After cloning state B, the FSM maintains separate statistics for situations when state B is reached from state A, and when it is reached from other states. Without loss of generality, suppose the former transition from A to B is associated with the symbol '1', as in the example. The new state B' then corresponds to the situation when state A is followed by '1'. As cloning continues, new states begin to express more and more specific situations, allowing the algorithm to incorporate richer context information when predicting the next symbol. The context used for prediction is implicitly determined by the longest string of symbols that matches all suffixes of paths leading to the current state of the FSM.

After cloning, the statistics associated with the cloned state B are distributed among B' and B in proportion to the number of times state B was reached from state A, relative to the number of times state B was reached from other states (again, refer to Figure 1). Two parameters control the state cloning operation in DMC. These are the minimal frequencies of B' and B after cloning. Both of the new states must exceed this minimal frequency that is required for stable probability estimates in order to trigger the cloning operation. At each position in the sequence only one state needs to be considered for cloning: The target state of the transition in the FSM that is triggered by the current symbol.

In the most basic version, the initial model contains a single state, corresponding to a memoryless source. When dealing with byte-aligned data, it is customary to start with a slightly more complex initial state machine which is capable of expressing within-byte dependencies. This initial FSM structure corresponds to an order-7 binary Markov model. All transitions in the initial FSM are primed with a small initial visit count to avoid singular probabilities. We note that although DMC restricts the source alphabet to binary symbols, it nevertheless achieves state-of-the-art performance on typical ASCII encoded text sequences (Cormack and Horspool, 1987).

3.4.2 PREDICTION BY PARTIAL MATCHING

The prediction by partial matching (PPM) algorithm (Cleary and Witten, 1984) has set the standard for lossless text compression since its introduction over two decades ago (Cleary and Teahan, 1997). Essentially, the PPM algorithm is a back-off smoothing technique for finite-order Markov models, similar to back-off models used in natural language processing.

It is convenient to assume that an order-k PPM model stores a table of all contexts (up to length k) that occur anywhere in the training text. For each such context, the frequency counts of symbols that immediately follow it is maintained. When predicting the next symbol x_i in a sequence x_1^n, its context x_{i-k}^{i-1} is matched against the stored statistics. The longest matching context x_{i-l}^{i-1} found in the table is examined first (note that $l \leq k$). If the target symbol has appeared in this context in the training text, its relative frequency within the context is used for prediction. However, this probability is *discounted* by a small amount to reserve some probability mass, which is called the *escape probability*. The escape probability that is accumulated in this way estimates the probability of observing a zero-frequency symbol in the context x_{i-l}^{i-1}. The escape probability is distributed among symbols *not* seen in the current context, according to a lower-order model, that is, according to statistics for the context x_{i-l+1}^{i-1}. The procedure is applied recursively until all symbols receive

a non-zero probability. If necessary, a default model of order -1 is used, which always predicts a uniform distribution among all possible symbols.

An adaptive compression algorithm based on the PPM model starts with an empty model which always defaults to the uniform distribution among all symbols. After each symbol is encoded, the algorithm updates the statistics of all contexts (up to order k) of the current symbol.

Many versions of the PPM algorithm exist, differing mainly in the way the escape probability is estimated. In our implementation, we used escape method D (Howard, 1993), which simply discounts the frequency of each observed character by $1/2$ occurrence and uses the gained probability mass for the escape probability.

4. Minimum Description Length Principle

The minimum description length (MDL) principle (Rissanen, 1978; Barron et al., 1998; Grünwald, 2005) favors models that yield compact representations of the data. The traditional two-part MDL principle states that the preferred model results in the shortest description of the model *and* the data, given this model. In other words, the model that best *compresses* the data is selected. This model selection criterion naturally balances the complexity of the model and the degree to which this model fits the data.

A problem of the two-part MDL principle is that it gives no guidelines as to how the model should be encoded. The refined MDL principle which is described later in this section aims to remedy this problem.

4.1 Universal Codes

A universal code relative to a class of source models has the property that it compresses data "almost" as well as the best model in the model class. More precisely, the difference in code length between a universal code and the best model in the model class increases sublinearly with the length of the sequence. Rissanen gives a precise non-asymptotic lower bound on this difference in the worst case (Rissanen, 1986), which turns out to be linearly related to the complexity of the data generating process (in terms of the number of parameters). He also shows that codes exist that achieve this bound.

Two-part codes are universal, since only a finite code length is required to specify the model. It turns out that adaptive codes are also universal codes (Rissanen, 1984). In fact, adaptive compression algorithms exist that are proven to achieve Rissanen's lower bound relative to the class of all finite-memory tree sources (e.g., Willems et al., 1995). The redundancy incurred due to the fact that adaptive methods start with an empty, uninformed model, can be compared to the cost of separately encoding the model in two-part codes.

4.2 Predictive MDL

The limitations of the original two-part MDL principle were largely overcome with the modern version of the principle (Rissanen, 1996), which advocates the use of one-part universal codes for measuring description length relative to a chosen model class. The use of adaptive codes for this task is sometimes denoted predictive MDL and is encouraged when the data is sequential in nature (Grünwald, 2005).

We aim to measure the description length of a set of documents relative to the class of Markov models of a certain order by using adaptive universal data compression algorithms, and to employ this measure as a criterion for classification. It is necessary to mention here that while PPM is universal in this sense, the same cannot be said for DMC. This is due to its "greedy" strategy of adapting its model without bound, that is, increasing the order of the model as soon as possible. On the other hand, this strategy might well lead to a better approximation of the source and thus more accurate prediction. In terms of data compression performance, DMC is competitive to PPM on the types of sequences that are of practical interest to us, particularly for natural language text and binary computer files (Cormack and Horspool, 1987).[4]

5. Text Classification Using Compression Models

In essence, compression algorithms can be applied to text categorization by building one compression model from the training documents of each class and using these models to evaluate the target document.

In the following subsections, we describe two approaches to classification. Both approaches model a class as an information source, and consider the training data for each class a sample of the type of data generated by the source. They differ in the way classification is performed. We first describe the minimum cross-entropy (MCE) approach (Frank et al., 2000; Teahan, 2000). This method chooses the class for which the associated compression model assigns the highest probability to the target document. We then propose a simple modification to this method, in which the model is *adapted* while evaluating the target document in the sense of Equation 1. Unlike the former approach, this method measures the increase of the description length of the data set as a result of the addition of the target document. It chooses the class for which the description length increase is minimal, which is why we consider this a minimum description length (MDL) approach. In subsequent sections, we also refer to this approach as using *adaptive* models and the MCE approach as using *static* models for obvious reasons.

We denote by C the set of classes and by $c : \Sigma^* \to C$ the (partially specified) function mapping documents to class labels. Given a set of pre-classified training documents D, the task is to assign a target document $\mathbf{d}$ with an unknown label to one of the classes $c \in C$.

5.1 Classification by Minimum Cross-entropy

The *cross-entropy* $H(X,M)$ determines the average number of bits per symbol required to encode messages produced by a source X when using a model M for compression:

$$H(X,M) = E_{\mathbf{x}\sim P}\left(\frac{1}{|\mathbf{x}|}L(\mathbf{x}|M)\right).$$

In the above equation, $L(\mathbf{x}|M)$ denotes the ideal code length for $\mathbf{x}$ under model M. Note that $H(X,M) \geq H(X)$ always holds, that is, the best possible model achieves a compression rate equal to the entropy.

4. As a side-note, we mention here that the techniques presented in this paper were also evaluated in combination with the Context Tree Weighting (CTW) compression algorithm (Willems et al., 1995) in the framework of the TREC 2005 spam track (Bratko and Filipič, 2005). Although the CTW algorithm is universal and provably achieves Rissanen's optimal minimax regret for the class of sources it considers, its performance for spam filtering in the TREC evaluation was comparable, although slightly inferior, to PPM. Since the CTW algorithm is also computationally less efficient, we omit the CTW algorithm from the present paper.

The exact cross-entropy is hard to compute, since it would require knowing the source distribution P. It can, however, be approximated by applying the model M to sufficiently long sequences of symbols, with the expectation that these sequences are representative samples of all possible sequences generated by the source (Brown et al., 1992; Teahan, 2000):

$$H(X,M) \approx \frac{1}{|\mathbf{x}|} L(\mathbf{x}|M). \tag{2}$$

As $|\mathbf{x}|$ becomes large, this estimate will approach the actual cross-entropy in the limit almost surely if the source is ergodic (Algoet and Cover, 1988). Recall that if M is a Markov model with limited memory k, then

$$L(\mathbf{x}|M) = -\log \prod_{i=1}^{|\mathbf{x}|} f(x_i|x_{i-k}^{i-1}, M)$$

where $f(x_i|x_{i-k}^{i-1}, M)$ is the probability assigned to x_i given x_{i-k}^{i-1} by M.

Following Teahan (2000), we refer to the cross-entropy estimated on the target document $\mathbf{d}$ as the *document* cross-entropy $H(X,M,\mathbf{d})$. This is simply a substitution of $\mathbf{x}$ with $\mathbf{d}$ in the right hand of Equation 2. We expect that a model that achieves a low cross-entropy on the target document approximates the information source that actually generated the document well. This is therefore our measure for classification:

$$\begin{aligned} c(\mathbf{d}) &= \arg\min_{c \in C} H(X, M_c, \mathbf{d}) \\ &= \arg\min_{c \in C} -\frac{1}{|\mathbf{d}|} \log \prod_{i=1}^{|\mathbf{d}|} f(d_i|d_{i-k}^{i-1}, M_c). \end{aligned} \tag{3}$$

In the above equation, M_c denotes the compression model built from all examples of class c in the training data.

5.2 Classification by Minimum Description Length

The MCE criterion assumes that the test document $\mathbf{d}$ was generated by some unknown information source. The document is considered a sample of the type of data generated by the unknown source. Classification is based on the distance between each class and the source that generated the document. This distance is measured with the document cross-entropy, which serves as an estimate of the cross-entropy between the unknown source and each of the class information sources.

However, we know that the document did not originate from some *unknown* source and that it ultimately must be attributed to one of the classes. The MDL classification criterion tests, for each class $c \in C$, the hypothesis that $c(d) = c$, by adding the document to the training data of the class and estimating how much this addition increases the description length of the data set:

$$\Delta L(D,c,\mathbf{d}) = L(\{\mathbf{x}\,;\mathbf{x} \in D, c(\mathbf{x}) = c)\} \cup \{\mathbf{d}\}) - L(\{\mathbf{x}\,;\mathbf{x} \in D, c(\mathbf{x}) = c)\}).$$

We are searching for the classification hypothesis that yields the most compact description of the observed data. The resulting description length is measured with adaptive compression algorithms which allow efficient estimation of this quantity, although other universal codes could also be used to measure the description length increase. This is in line with the approach suggested by Kontkanen

et al. (2005) in their MDL framework for clustering, in which the cluster assignment should be such that it results in a minimal description length of the data, measured by a suitable universal model.

Adaptive models are particularly suitable for this type of classification, since they can be used to estimate the increase in description length without re-evaluating the entire data set:

$$\Delta L(D,c,\mathbf{d}) = -\log \prod_{i=1}^{|\mathbf{d}|} f(d_i | d_{i-k}^{i-1}, M_c(d_1^{i-1})).$$

In the above equation, $M_c(d_1^{i-1})$ denotes the current model at position i, constructed from the training examples for class c and the input sequence d_1^{i-1}.

Typically, the description length increase $\Delta L(D,c,\mathbf{d})$ will be larger for longer documents. We therefore use the per-symbol description length increase in the final class selection rule:

$$\begin{aligned} c(\mathbf{d}) &= \arg\min_{c \in C} \frac{1}{|\mathbf{d}|} \Delta L(D,c,\mathbf{d}) \\ &= \arg\min_{c \in C} -\frac{1}{|\mathbf{d}|} \log \prod_{i=1}^{|\mathbf{d}|} f(d_i | d_{i-k}^{i-1}, M_c(d_1^{i-1})). \end{aligned} \quad (4)$$

The additional $1/|\mathbf{d}|$ factor does not affect the classification outcome for any target document, but it does help to produce scores that are comparable across documents of different length. This is crucial when thresholding is used to reach a desirable tradeoff in misclassification rates. Note that the only difference in implementation in comparison to the MCE criterion in Equation 3 is that the model is adapted while evaluating the target. It is clear, however, that Equation 4 no longer amounts to measuring the document cross-entropy $H(X, M_c, \mathbf{d})$ with respect to model M_c, since a different model is used at each position of the sequence $\mathbf{d}$.

Intuitively, the description length increase $\Delta L(D,c,\mathbf{d})$ measures the "*surprise*" at observing d under the hypothesis $c(d) = c$, which is proportional to the (im)probability of d under the model for c. The intuition behind adapting the model M_c is that it *conditionally* continues to learn about c from the target document: If the hypothesis $c(d) = c$ actually holds and an improbable pattern is found in the initial part of d, then the probability that the pattern reoccurs in the remainder of the document should increase. Although we do not know whether the hypothesis $c(d) = c$ is true or not, it is assumed to be true for the purpose of testing its tenability.

Let us conclude this section with an illustrative example as to why the MDL classification criterion might be preferable to the MCE approach. Consider a hypothetical spam filtering problem in which a machine learning researcher uses a compression-based classifier to filter spam from his email. In addition to research-related email, our researcher also receives an abundant amount of spam that advertises prescription drugs. At some point, he receives an email on machine learning methods for drug discovery. This is a legitimate email, but it contains many occurrences of two particular terms that the filter strongly associates with spam: "medicine" and "drugs". In this scenario, the prevalence of these two terms might cause the MCE criterion to label the email as spam, but the MDL criterion would probably consider the email legitimate. This is because while the first occurrence of the terms "medicine" and "drugs" are surprising under the hypothesis "document is legitimate", subsequent occurrences are less surprising. They are, in a sense, redundant. The classifier will learn this as a direct consequence of allowing the model to adapt to the target.

It is interesting to note that Benedetto et al. (2002), who consider the use of the LZ77 compression algorithm (zip) for language identification and authorship attribution, notice that LZ77 adapts

to the target text and take measures to prevent this behavior. We, on the other hand, believe this effect is beneficial, which is supported in the results of our experiments.

6. Experimental Setup and Evaluation Methodology

Our primary concern is the use of compression models in spam filtering. This problem differs from classical text categorization tasks in a number of ways:

- The cost of misclassification is highly unbalanced. Although the exact tradeoff will vary in different deployment environments, it tends to be biased toward minimizing false positives (i.e., misclassified legitimate messages).
- Messages in an email stream arrive in chronological order and must be classified upon delivery. It is also common to deploy a filter without *any* training data. Although previous studies typically use cross validation experiments, the appropriateness of cross validation is questionable in this setting.
- Many useful features may be gleaned from various message headers, formats and encodings, punctuation patterns and structural features. It is therefore desirable to use raw, unobfuscated messages with accompanying meta data intact for evaluation.

These unique characteristics of the spam filtering problem are reflected in the design of our experiments and the choice of measures that were used for classifier evaluation. This section gives an overview of the test corpora and evaluation methodology used to obtain our results.

6.1 Online Spam Filter Evaluation

An online learning scheme that lends itself well to typical usage of spam filters was adopted as the primary means of classifier evaluation. In this setup, messages are presented to the classifier in chronological order. For each message, the classifier must produce a score as to how likely it is that the message is spam, after which it is communicated the gold standard judgment. This allows the classifier to update its model before assessing the next message.

The setup aims to simulate a typical setting in personal email filtering, which is usually based on online user feedback, with the additional assumption that the user promptly corrects the classifier after every misclassification. The same evaluation method was used in the large-scale spam filter evaluation at TREC 2005, an overview of which can be found in the TREC proceedings (Cormack and Lynam, 2005).

The performance of different compression models and classification criteria were evaluated using the described scheme. We also compared compression models to a selection of established spam filters in this manner. Standard cross validation experiments on predefined splits were performed to compare compression models to previously published results which were also obtained with cross validation.

6.2 Evaluation Measures

Special care must be taken in the choice of evaluation measures for spam filtering. Classification accuracy, that is, the total proportion of misclassified messages, is a poor performance measure in this application domain, since all errors are treated on equal footing (Androutsopoulos et al., 2000).

In the binary spam filtering problem, spam messages are usually associated with the positive class, since these are the messages filtered by the system. Legitimate messages are thus designated to the negative class. If p is the total number of positive examples in the test set and n is the total number of negative examples, four classification outcomes are defined by the standard binary contingency table. Legitimate message may be incorrectly labeled as spam (fp – false positives) or correctly identified as legitimate (tn – true negatives). Similarly, spam messages may be incorrectly labeled as legitimate (fn – false negatives) or correctly identified as spam (tp – true positives). The false positive rate (FPR) and spam misclassification rate (SMR) are then defined as follows:

$$\text{FPR} = \frac{fp}{n}, \qquad \text{SMR} = \frac{fn}{p}.$$

FPR and SMR measures are intuitive and appealing, however, it is difficult to compare systems based on these measures alone, since one of them can always be improved at the expense of the other.

It is assumed that the scores produced by a learning system are comparable across messages, so that a fixed filtering threshold can be used to balance between spam misclassification and false positive rates. Such scores lend themselves well to Receiver Operating Characteristic (ROC) curve analysis, which was the primary means of classifier evaluation in the study. The ROC curve is a plot of spam accuracy ($1 - \text{SMR}$) on the Y axis, as a function of the false positive rate on the X axis.[5] Each point on the curve corresponds to an actual (FPR, SMR) pair achieved by the classifier at a certain threshold value. The curve thus captures the behavior of the system at all possible filtering thresholds.

A good performance is characterized by a curve that reaches well into the upper left quadrant of the graph. The area under the ROC curve (AUC) is then a meaningful statistic for comparing filters. If we assume that high score values are associated with the positive class, the area under the curve equals the probability that a random positive example receives a higher score than a random negative example:

$$\text{AUC} = P(score(\mathbf{x}) > score(\mathbf{y}) \mid c(\mathbf{x}) = positive,\ c(\mathbf{y}) = negative).$$

Typical spam filters achieve very high values in the AUC statistic. For this reason, we report on the complement of the AUC value, that is, the area *above* the curve (1-AUC). Bootstrap resampling was used to compute confidence intervals for logit-transformed AUC values and to test for significance in paired comparisons. Where suitable, we also report SMR at filtering thresholds that result in "acceptable" false positives rates (0.01%, 0.1% and 1%). This measure is easier to interpret and gives insight in the kind of performance one can expect from a spam filter.

6.3 Data Sets

We report experimental results on five publicly available data sets and a private collection of email compiled by one of the authors. The basic statistics for all six corpora are given in Table 1.

The TREC public[6] corpus contains messages received by employees of the Enron corporation over a one year period. The original Enron data was carefully augmented with the addition of

5. It is traditional to name the axes of an ROC plot 1-specificity (X axis) and sensitivity (Y axis). Sensitivity is the proportion of correctly identified positive examples and specificity is the proportion of correctly identified negative examples.

6. The TREC corpus is available for download at `http://plg.uwaterloo.ca/~gvcormac/treccorpus/`.

Data Set	*Messages*	*Spam*	*Legitimate*	*Spam proportion*
TREC public	92189	52790	39399	57.3%
MrX	49086	40048	9038	81.6%
SpamAssassin	6033	1884	4149	31.2%
Ling-Spam	2893	481	2412	16.6%
PU1	1090	480	610	44.0%
PU3	4130	1820	2310	44.1%

Table 1: Basic statistics for the evaluation data sets.

approximately 50,000 spam messages, so that they appear to be delivered to the Enron employees in the same time period as the legitimate email.

The MrX data set contains email messages received by a single email user over a period of 8 months. This data set and the TREC corpus were recently used for the spam filter evaluation track in TREC 2005. Results from the TREC evaluation that are most relevant to our study are reproduced in this paper.

The SpamAssassin[7] data set contains legitimate and spam email collected from the SpamAssassin developer mailing list. This data set is arguably the most widely used resource in popular evaluations of publicly available spam filters, often conducted by enthusiasts or system authors.

Ling-Spam[8] is a collection of email messages posted to a linguistics newsgroup, which were augmented with spam received by the authors of the data set. The messages are stripped of all attachments and headers, except for the subject field. A fair number of research studies report results on the Ling-Spam corpus. We used the "bare" version of this data set in our evaluation.

The PU1 and PU3[9] data sets are relatively small personal email collections. In order to preserve privacy, the words in the messages are replaced with numerical identifiers and punctuation is discarded. Non-textual message headers, sender and recipient fields, attachments and HTML tags are not included in these data sets. Duplicate spam messages received on the same day are also removed.

We used the online evaluation scheme described in the previous subsection to evaluate performance on the TREC public, MrX and SpamAssassin corpora. We performed cross validation experiments on the remaining three data sets, as was done in previous studies. The Ling-Spam, PU1 and PU3 data sets contain predefined 10-fold cross validation splits, which we used in our experiments. These data sets do not contain message headers, so the original chronological order required for online evaluation could not be recovered.

6.4 Implementation and Parameters of DMC and PPM Models

All results reported in the study were achieved using our own implementations of DMC and PPM compression models. Classifiers based on the DMC and PPM compression models were developed independently by the authors and differ in preprocessing strategies and certain implementation details.

7. The SpamAssassin data set is available at `http://spamassassin.org/publiccorpus/`.
8. The Ling-Spam data set is available at `http://www.aueb.gr/users/ion/data/`.
9. The PU1 and PU3 data sets are available for download at `http://www.iit.demokritos.gr/skel/i-config/downloads/PU123ACorpora.tar.gz`.

The DMC model was primed with an initial braid structure (Cormack and Horspool, 1987), corresponding to an order-7 binary Markov model. DMC uses two parameters that control its state cloning mechanism. These parameters were set somewhat arbitrarily to $(2,2)$, since such values were known by the authors to perform well for data compression. The initial transition counts were set to 0.2, following a similar argument. The DMC implementation does not include MIME decoding. It also truncates all messages to 2500 bytes.

The PPM implementation used an order-6 PPM-D model in all trials. Order-4 and order-8 models were also tested in the TREC evaluation, from which it was concluded that performance is robust to the choice of this parameter (Bratko and Filipič, 2005). In data compression, an order-6 model would also be considered suitable for compression of English text. The source alphabet for PPM was restricted to 72 ASCII characters including alphanumerical symbols and commonly used punctuation. This alphabet was complemented with an additional symbol that was used for all other ASCII codes found in the text. Our PPM-based classifier decodes base64-encoded message parts and discards all non-text attachments before evaluation.

The PPM-based classifier used a memory buffer of approximately 800MB, substantially less than the DMC implementation which was limited to 2GB of RAM. Both algorithms used the same retraining strategy when this memory limit was reached in online evaluation experiments. Specifically, half of the training data was discarded and models were retrained from the more recent part of the email stream. This mechanism was invoked up to twice during online evaluation on the two larger data sets (MrX and the TREC public corpus), but was not used in any of the other trials.

We realize that this setup does not facilitate a fair comparison between the two compression algorithms in the online experiments (on raw email data), as the different preprocessing schemes were found to have an effect on performance in some of these experiments. However, the aim of this paper is the evaluation of compression models against existing spam filtering techniques, as well as a comparison of the two classification criteria discussed in Section 5. We are satisfied with the general observation that both algorithms exhibit similar performance, which strengthens our confidence in the applicability of the proposed methods for the spam filtering problem.

6.5 Reference Systems Used for Comparative Evaluation

A number of freely available open source spam filters have been introduced in recent years, motivated mainly by the influential essays of Graham (2004) and Robinson (2003). A wide variety of learning algorithms, training strategies, preprocessing schemes and recipes for feature engineering are employed in these systems. It is interesting to note that most publications that address spam filtering do not compare their proposed methods to these established alternatives.

We evaluate the performance of compression models against six popular open source filters. We also summarize results obtained in other studies that use the Ling-Spam, PU1 and PU3 corpora, and from which it is possible to determine misclassification rates from the published results. Table 2 lists all systems that were included in any of the comparisons.

7. Results

In this section, we report the main results of our evaluation. We first evaluate the performance of the MCE and MDL classification criteria, that is, the effect of adapting the model to the target, for both compression algorithms. This is followed by an extensive evaluation of compression-based classifiers in comparison to established spam filters and results published in other studies. We

Label		Description
Bogofilter[a]	⋆	Version 0.94.0, default parameters (`http://www.bogofilter.org`).
Bogofilter[b]	•	Bogofilter version 0.95.2 as configured for TREC 2005 by the track organizers.
CRM114[a]	⋆	Version 20041231, default parameters (`http://crm114.sourceforge.net`).
CRM114[b]	•	CRM114 specially configured by Assis et al. (2005) for TREC. Labeled *"CRMSPAM2"* at TREC 2005.
dbacl[a]	⋆	Version 1.91, default parameters (`http://dbacl.sourceforge.net`).
dbacl[b]	•	A custom version of dbacl prepared by the author for evaluation at TREC (Breyer, 2005). Labeled *"lbSPAM2"* at TREC 2005.
SpamAssassin[a]	⋆	Version 3.0.2, combination of rule-based and learning components (`http://spamassassin.apache.org`).
SpamAssassin[b]	•	Version 3.0.2, learning component only, as configured for TREC 2005 by the track organizers.
SpamBayes[a]	⋆	Version 1.03, default parameters (`http://spambayes.sourceforge.net`).
SpamBayes[b]	•	SpamBayes specially configured by Meyer (2005) for TREC. Labeled *"tamSPAM1"* at TREC 2005.
SpamProbe	•⋆	Version 1.0a, default parameters (`http://spamprobe.sourceforge.net`).
a-Bayes	◇	Naive Bayes, multi-variate Bernoulli model with binary features (Androutsopoulos et al., 2000).
a-FlexBayes	◇	Flexible naive Bayes—uses kernel density estimation for estimating class-conditional probabilities of continuous valued attributes (Androutsopoulos et al., 2004).
a-LogitBoost	◇	LogitBoost (variant of boosting) with decision stumps as base classifiers (Androutsopoulos et al., 2004).
a-SVM	◇	Linear kernel support vector machines (Androutsopoulos et al., 2004).
b-Stack	◇	Stacking of linear support vector machine classifiers built from different message fields (Bratko and Filipič, 2006).
c-AdaBoost	◇	Boosting of decision trees with real-valued predictions (Carreras and Márquez, 2001).
gh-Bayes	◇	Naive Bayes (exact model unknown) with weighting of training instances according to misclassification cost ratio (Hidalgo, 2002).
gh-SVM	◇	Linear support vector machine with weighting of training instances according to misclassification cost ratio (Hidalgo, 2002).
h-Bayes	◇	Multinomial naive Bayes (Hovold, 2005).
ks-Bayes	◇	Multinomial naive Bayes (Schneider, 2003).
p-Suffix	◇	Pattern matching of character sequences based on the suffix tree data structure and various heuristic scoring functions (Pampapathi et al., 2006).
m-Filtron	◇	Support vector machines with linear kernels (Michelakis et al., 2004).
s-Stack	◇	Stacking of naive Bayes and *k*-nearest neighbors (Sakkis et al., 2001).
s-kNN	◇	*k*-nearest neighbors with attribute and distance weighting (Sakkis et al., 2003).
SVM	⋆	An adaptation of the SVM^{light} package (Joachims, 1998) for the PU1 data set due to Tretyakov (2004), linear kernel with C = 1.
Perceptron	⋆	Implementation of the perceptron algorithm due to Tretyakov (2004).

Table 2: Reference systems and results of previous studies reproduced for comparison. Entries are delimited by primary authors. Symbols indicate the source of reported results:
⋆ – this study • – TREC 2005 evaluation ◇ – reproduced from other studies

conclude the section with experiments that study the effect of noise introduced in data by typical obfuscation tactics employed by spammers to evade filtering. Additional results from our evaluation are available in Online Appendix 1.[10]

7.1 Performance of MCE vs. MDL Classification Criteria

We evaluated the effect of adapting the compression model to the target document on the TREC public, MrX and SpamAssassin data sets. AUC scores achieved by the static and adaptive DMC and PPM models are listed in Table 3. The adaptive models clearly outperform their static counterparts on all data sets, sometimes strikingly so. The area above the ROC curve is more than halved in two of the six experiments and substantially improved in three of the remaining four trials. The improvement is smallest for the DMC model tested on the TREC public data set. As we shall see, even the baseline performance achieved by the static model is exceptionally good in this experiment, and thus hard to improve.

	DMC		**PPM**	
Data Set	MCE	MDL	MCE	MDL
TREC	0.014 (0.010–0.020)	**0.013** (0.010–0.018)	0.038 (0.027–0.052)	**0.019**† (0.015–0.023)
MrX	0.065 (0.040–0.11)	**0.037**† (0.026–0.053)	0.11 (0.073–0.16)	**0.069**† (0.044–0.11)
SpmAssn	0.31 (0.21–0.47)	**0.20**† (0.14–0.30)	0.35 (0.20–0.60)	**0.15**† (0.086–0.26)

Table 3: Performance of DMC and PPM algorithms in combination with the MCE and MDL classification criteria on the TREC public, MrX and SpamAssassin data sets. Results are in the area above the ROC curve 1-AUC(%) statistic and include 95% confidence intervals for this measure. The best results for each algorithm/data set pair are in bold. Statistically significant differences are marked with a '†' sign ($p < 0.01$, one-tailed).

The ROC curves and 1-AUC learning curves of adaptive and static DMC and PPM models are depicted in Figure 2. Let us first comment on the ROC curves, which reveal an interesting and remarkably consistent pattern. Note that the ROC graphs are plotted in logarithmic scale for clarity, so a non-convex ROC area is not necessarily unexpected. Although adaptive models dominate throughout the curve in most experiments, the gain in the 1-AUC statistic can mostly be attributed to the fact that the adaptive models perform better at the extreme ends of the curves. Performance is comparable when FPR and SMR are balanced. The logarithmic scale should again be taken into consideration when examining the magnitude of this effect. This suggests that the adaptive model makes less gross mistakes, which are costly in terms of the AUC measure. Performance at the extreme ends of the curve is especially important when the cost of misclassification is unbalanced, so this is certainly a desirable property for spam filtering.

The learning curves in Figure 2 depict accumulated 1-AUC scores sampled at 1000 message intervals during the online learning experiments. They are again plotted in logarithmic scale to facilitate evaluation of the asymptotic performance of classifiers. The main observation offered by the learning curves is that the adaptive models do not achieve better overall performance at the price of slower learning rates. Their performance is superior throughout the runs. The difference in 1-AUC scores is in fact greater in the earlier stages of learning for two of the three data sets. This is intuitive, since the effect of adapting the models will be greater for simpler models built from

10. Available at `http://ai.ijs.si/andrej/papers/jmlr2006/`.

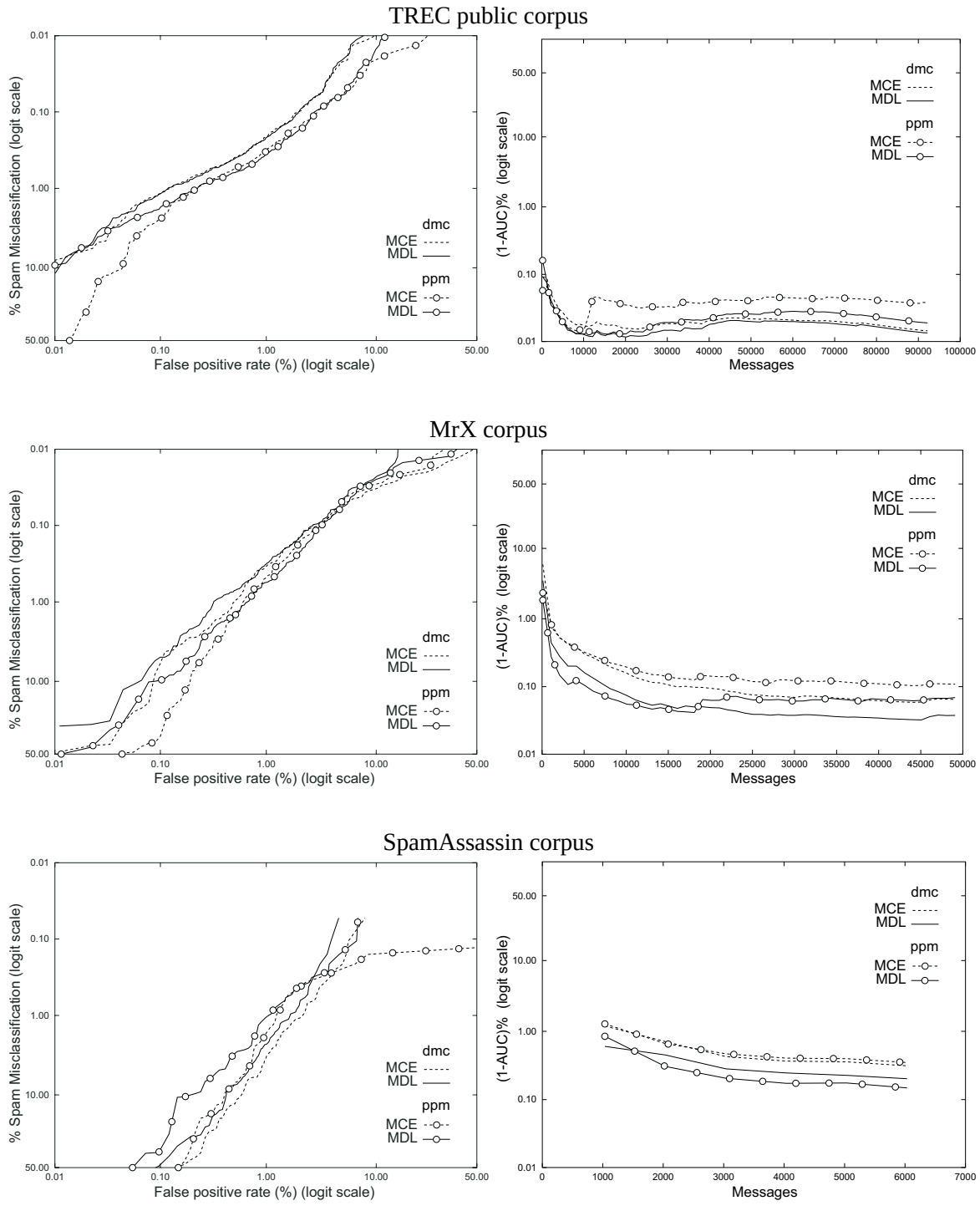


Figure 2: ROC curves (left) and 1-AUC learning curves (right) on the MrX, SpamAssassin and TREC public corpora. MCE and MDL classification criteria estimated with the DMC and PPM algorithms are compared.

limited training data. In Online Appendix 2,[11] we address an apparent anomaly which occurs in the learning curve of the static version of the PPM classifier at around 12,000 messages on the TREC public corpus. The analysis presented in the appendix exposes the differences between the MCE and MDL classification criteria, but is beyond the scope of the current discussion.

7.2 Comparison to Open Source Spam Filters

The results of our experimental comparison of compression models and established open source filters are summarized in Table 4. Adaptive versions of the PPM and DMC classifiers were used for this comparison. In terms of the 1-AUC score, both compression models uniformly outperformed all of the competing filters, with the exception of the PPM classifier on the MrX corpus. The performance of DMC and PPM models on the TREC public corpus is particularly notable. Spam misclassification rates at hypothetical filtering thresholds that result in a low proportion of false positives are also shown. Although definitive conclusions are harder to draw from these measures, we find that both DMC and PPM feature prominently, outperforming other methods in two of the tree tradeoff points on every data set. ROC curves and learning curves of compression models and open source filters on the TREC public corpus are shown in Figure 3.

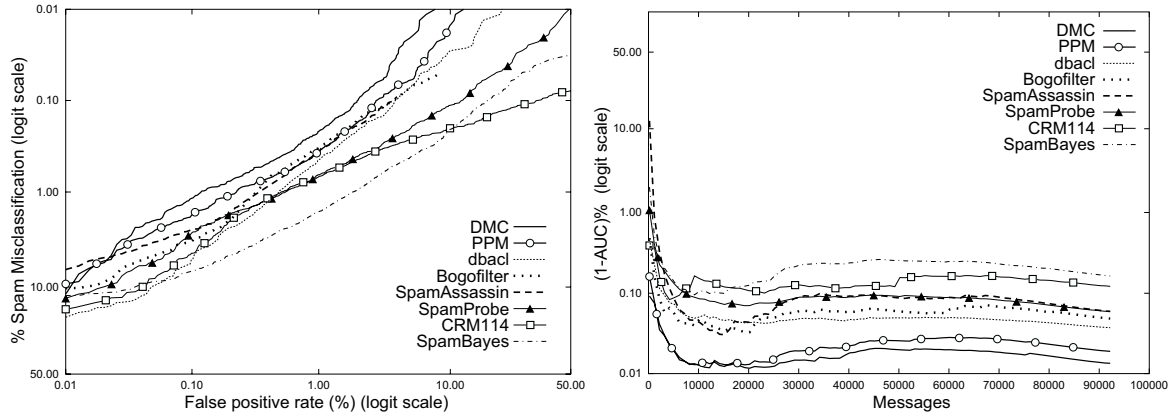


Figure 3: ROC curves (left) and 1-AUC learning curves (right) for compression models and a selection of established spam filters on the TREC public corpus.

7.3 Comparison with Published Results

We conducted standard cross validation experiments to evaluate classification performance of PPM and DMC (adaptive versions) on the Ling-Spam, PU1 and PU3 data sets. Implementations of the perceptron and SVM classifiers, as well as Bogofilter, a well-performing open source filter from previous experiments, were also tested in this manner. Results of these experiments are presented in Figures 4 and 5, in which we also reproduce results of previous studies on the same data. The simplified ROC-style graphs plot the number of misclassified spam messages against the number of false positives.

11. Available at `http://ai.ijs.si/andrej/papers/jmlr2006/`.

TREC public corpus

Filter	1-AUC (%)	SMR at 1% FP	SMR at 0.1% FP	SMR at 0.01% FP
DMC	†**0.013** (0.010 – 0.018)	0.22%	1.17%	14.47%
PPM	†**0.019** (0.015 – 0.023)	0.36%	1.78%	9.89%
dbacl[b]	**0.037** (0.031 – 0.045)	0.45%	5.19%	19.77%
Bogofilter[b]	**0.048** (0.038 – 0.062)	0.33%	3.41%	10.39%
SpamAssassin[b]	**0.059** (0.044 – 0.081)	0.37%	2.56%	7.81%
SpamProbe	**0.059** (0.049 – 0.071)	0.65%	2.77%	15.30%
CRM114[b]	**0.122** (0.102 – 0.145)	0.68%	4.52%	17.17%
SpamBayes[b]	**0.164** (0.142 – 0.189)	1.63%	6.92%	12.55%

MrX corpus

Filter	1-AUC (%)	SMR at 1% FP	SMR at 0.1% FP	SMR at 0.01% FP
DMC	†**0.037** (0.026 – 0.053)	0.32%	5.08%	36.16%
Bogofilter[b]	**0.045** (0.032 – 0.063)	0.57%	3.90%	31.04%
CRM114[b]	**0.051** (0.035 – 0.075)	0.43%	9.65%	47.76%
PPM	**0.069** (0.044 – 0.107)	0.56%	9.72%	94.42%
dbacl[b]	**0.083** (0.054 – 0.130)	0.43%	10.24%	99.09%
SpamAssassin[b]	**0.097** (0.070 – 0.135)	0.77%	6.19%	83.06%
SpamProbe	**0.097** (0.063 – 0.150)	0.35%	15.54%	95.08%
SpamBayes[b]	**0.138** (0.111 – 0.171)	1.10%	6.51%	45.65%

SpamAssassin corpus

Filter	1-AUC (%)	SMR at 1% FP	SMR at 0.1% FP	SMR at 0.01% FP
PPM	**0.148** (0.086 – 0.256)	1.06%	38.67%	66.42%
DMC	**0.202** (0.136 – 0.301)	2.39%	48.86%	64.93%
Bogofilter[a]	**0.209** (0.140 – 0.312)	3.08%	57.67%	99.10%
SpamAssassin[a]	**0.254** (0.173 – 0.373)	4.93%	24.77%	100.00%
dbacl[a]	**0.262** (0.169 – 0.404)	2.65%	58.36%	79.26%
SpamProbe	**0.296** (0.195 – 0.450)	2.18%	74.43%	99.47%
CRM114[a]	**1.143** (0.902 – 1.446)	6.26%	57.82%	83.02%
SpamBayes[a]	**1.391** (1.036 – 1.867)	11.35%	92.73%	99.26%

Table 4: Performance of DMC, PPM and a selection of established spam filters on the TREC public, MrX and SpamAssassin data sets. Filters are ordered by decreasing performance in the 1-AUC statistic. Significant differences between AUC scores achieved by the compression models and the best competing filter are marked with a '†' sign ($p < 0.05$, one-tailed).

On the Ling-Spam data set, both compression models are comparable to the suffix tree approach of Pampapathi et al. (2006). These three classifiers dominate the other methods at all filtering thresholds. Both the suffix tree classifier and the compression models considered in this paper operate on character-level or binary sequences. All other methods use the standard bag-of-words representation for modeling text. This suggests that methods based on sub-word symbol sequences are more suitable for the Ling-Spam data set, and we believe this to be the case for spam filtering in general. However, experimental results on the PU1 and PU3 corpora show that character or binary-level modeling of text is not the only advantage offered by the compression models.

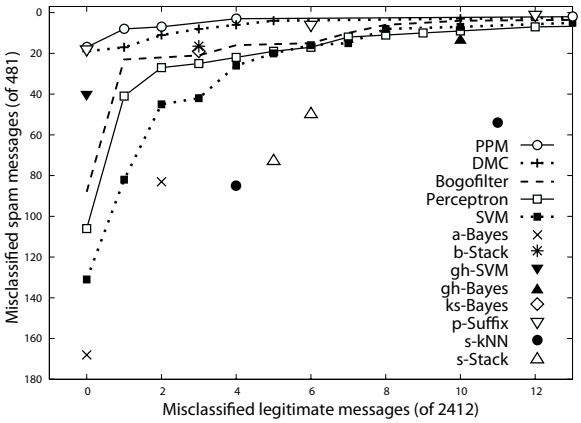


Figure 4: Performance of compression models in comparison to the perceptron and SVM classifiers, Bogofilter and previously published results on the Ling-Spam data set.

The PU1 and PU3 data sets contain pre-tokenized messages in which the original words are replaced with numeric identifiers. We converted these messages to binary format by replacing token identifiers with their 16 bit binary equivalents. This representation was used to evaluate the performance of the DMC classifier on the two data sets. We believe this is fair, since DMC uses a binary alphabet and is hurt by the artificial tokenization of these data sets. The PPM classifier was tested on the unprocessed original version of the data sets. Digits in numeric identifiers were converted to alphabetical characters for testing with Bogofilter, since the filter handles digits differently from alphabetical character strings.

The graphs in Figure 5 show classification performance on the PU1 and PU3 data sets and are perhaps the most surprising result reported in this paper. Both compression models outperform other classifiers almost uniformly across the range of 'interesting' filtering thresholds, despite the fact that the data sets were produced for tokenization-based filters. The SVM is competitive on the PU1 data set. Other methods are further behind the SVM in both trials. We attribute the good performance of compression models in these tests to the fact that tokens are *not* considered independently. Their probability is always evaluated with respect to the local context, which was already found to be beneficial for word-level models in previous studies (Peng et al., 2004). Compression models offer the additional advantage over language models considered by Peng et al. (2004) in their effective strategy for adapting the model structure incrementally. By design, the compression models can discover phrase-level patterns just as naturally as sub-word patterns.

7.4 Sensitivity to Noise in the Data

One of the perceived advantages of statistical data compression models over standard text categorization algorithms is that they do not require any preprocessing of the data. Classification is not based on word tokens or manually constructed features, which is in itself appealing from the implementation standpoint. For spam filtering, this property is especially welcome, since preprocessing steps are error-prone and are often exploited by spammers in order to evade filtering. A typical strategy is to distort words with common spelling mistakes or character substitutions, which may

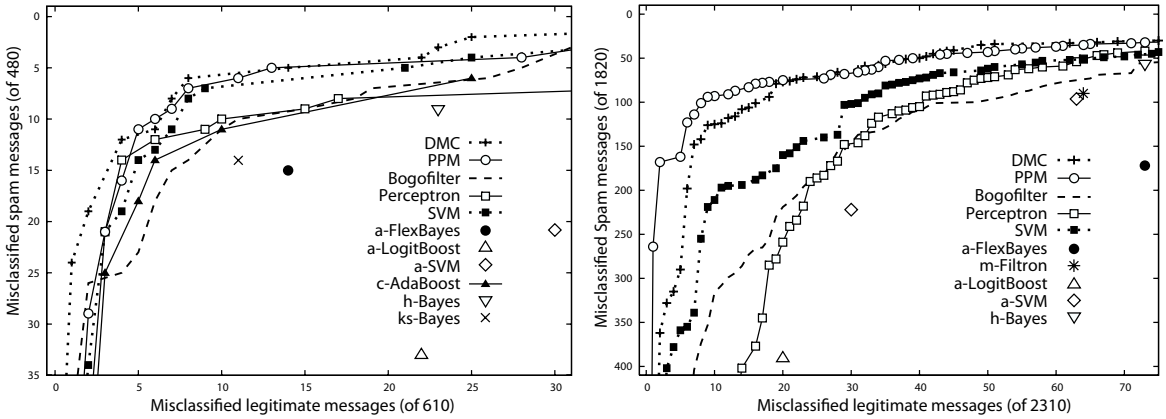


Figure 5: Performance of compression models in comparison to the perceptron and SVM classifiers, Bogofilter and previously published results on the PU1 (left) and PU3 (right) data sets.

confuse an automatic filter. We believe that compression models are much more robust to such tactics than methods which require tokenization.

To support this claim, we conducted an experiment in which all messages in the SpamAssassin data set were distorted by substituting characters in the original text with random alphanumeric characters and punctuation. The characters in the original message that were subject to such a substitution were also chosen randomly. By varying the probability of distorting each character, we evaluated the effect of such noise on classification performance.

For this experiment, all messages were first decoded and stripped of non-textual attachments. Noise was added to the subject and body parts of messages. Adaptive compression models and Bogofilter, a representative tokenization-based filter, were evaluated on the resulting data set. We then stripped the messages of all headers except subjects and repeated the evaluation. The results of these experiments are depicted in Figure 6. They show that compression models are indeed very robust to noise. Even after 20% of all characters are distorted, rendering messages practically illegible, they retain a respectable performance. The advantage of compression models on noisy data is particularly pronounced in the second experiment, where the classifier must rely solely on the (distorted) textual contents of the messages. It is interesting to note that the performance of PPM increases slightly at the 5% noise level if headers are kept intact. We have no definitive explanation for this phenomenon. We suspect that introducing noise in the text implicitly increases the influence of the non-textual message headers, and that this effect is beneficial.

8. Conclusion

In comparison to tokenization-based classification methods, compression models offer a number of advantages that are especially relevant for spam filtering. By operating directly on sequences, tokenization, stemming and other tedious and error-prone preprocessing steps are omitted altogether. It is precisely these volatile preprocessing steps that are often exploited by spammers in order to evade filtering. Also, characteristic sequences of punctuation and other special characters, which are generally thought to be useful in spam filtering, are naturally included in the model. The algo-

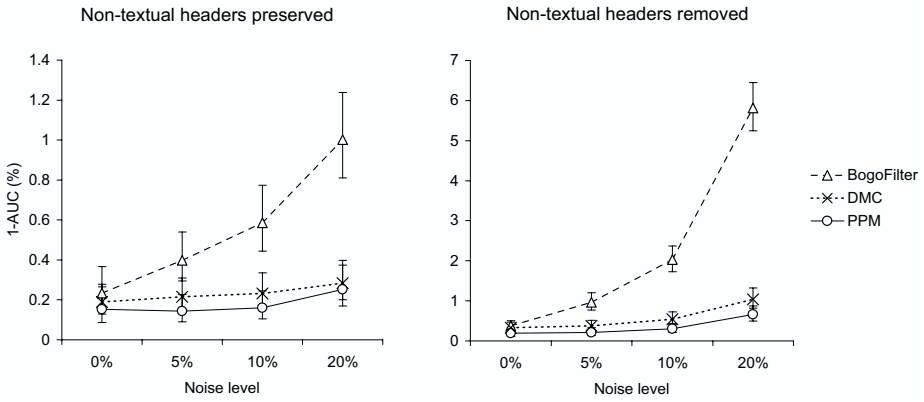


Figure 6: Effect of noise on classification performance on the SpamAssassin data set. The graph shows 1-AUC scores with 95% confidence intervals for PPM, DMC and the tokenization-based Bogofilter system at different levels of artificial random noise in the data.

rithms are efficient, with training and classification times linear in the amount of data. The models are incrementally updateable, which is often a requirement for practical spam filters that support online learning based on user feedback.

Empirically, we demonstrate that compression models perform very well for spam filtering, consistently outperforming established spam filters and other methods proposed in previous studies. We also show that compression models are very robust to the type of noise introduced in the text by typical obfuscation tactics used by spammers. This should make them difficult for spammers to defeat, but also makes them attractive for other text categorization problems that contain noisy data, such as classification of text extracted with optical character recognition.

Finally, we find that updating compression models adaptively to the target document is beneficial for classification, particularly in improving the AUC measure. This is especially desirable when the cost of misclassification is uneven, as is the case in spam filtering. The modification has a natural interpretation in terms of the minimum description length principle. Although we are aware of no parallel to this in existing text classification research, the same approach could easily be adopted for the popular multinomial naive Bayes model (McCallum and Nigam, 1998) and possibly also for other incremental models. We believe this to be an interesting avenue for future research.

The large memory requirements of compression models are a major disadvantage of this approach. To this end, effective pruning strategies should be investigated in order to bring the models within limits that would be suitable for practical applications. Should compression models actually be employed in practice, the adversarial nature of spam filtering suggests spammers will react to these techniques. It remains to be seen whether their efforts could reduce the long-term efficacy of the proposed approach.

References

P. H. Algoet and T. M. Cover. A sandwich proof of the Shannon-McMillan-Breiman theorem. *Annals of Probability*, 16:899–909, 1988.

I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proc. of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, pages 9–17, 2000.

I. Androutsopoulos, G. Paliouras, and E. Michelakis. Learning to filter unsolicited commercial e-mail. Technical Report 2004/2, NCSR "Demokritos", October 2004.

F. Assis, W. Yerazunis, C. Siefkes, and S. Chhabra. CRM114 versus Mr. X: CRM114 notes for the TREC 2005 spam track. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

A. R. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

D. Benedetto, E. Caglioti, and Loreto V. Language trees and zipping. *Physical Review Letters*, 88 (4), 2002.

A. Bratko and B. Filipič. Spam filtering using character-level markov models: Experiments for the TREC 2005 Spam Track. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

A. Bratko and B. Filipič. Exploiting structural information for semi-structured document categorization. *Information Processing & Management*, 42(3):679–694, 2006.

L. A. Breyer. DBACL at the TREC 2005. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

P. F. Brown, S. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40, 1992.

X. Carreras and L. Márquez. Boosting trees for anti-spam email filtering. In *Proc. of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, 2001.

J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40 (2/3):67–75, 1997.

J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, COM-32(4):396–402, April 1984.

G. V. Cormack and R. N. S. Horspool. Data compression using dynamic Markov modelling. *The Computer Journal*, 30(6):541–550, 1987.

G. V. Cormack and T. R. Lynam. TREC 2005 Spam Track overview. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In *Proceedings of DCC-00, IEEE Data Compression Conference*, pages 200–209, Snowbird, US, 2000. IEEE Computer Society Press, Los Alamitos, US.

J. Goodman, D. Heckerman, and R. Rounthwaite. Stopping spam. *Scientific American*, 292(4): 42–88, April 2005.

P. Graham. *Hackers and Painters, Big Ideas from the Computer Age*, chapter 8, pages 121–130. O'Reilly, 2004.

P. Grünwald. A tutorial introduction to the minimum description length principle. In P. Grünwald, I. J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*, pages 3–81. MIT Press, 2005.

J. M. G. Hidalgo. Evaluating cost-sensitive unsolicited bulk email categorization. In *SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 615–620, Madrid, March 2002. ACM Press.

J. Hovold. Naive bayes spam filtering using word-position-based attributes. In *Proc. of the 2nd Conference on Email and Anti-Spam (CEAS 2005)*, Palo Alto, CA, July 2005.

P. G. Howard. *The Design and Analysis of Efficient Lossless Data Compression Systems.* PhD thesis, Brown University, Providence, Rhode Island, 1993.

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

E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD '04: Proc. of the 10th ACM SIGKDD conference on Knowledge Discovery and Data mining*, pages 206–215, Seattle, WA, 2004. ACM Press.

P. Kontkanen, P. Myllymki, W. Buntine, J. Rissanen, and H. Tirri. An MDL framework for data clustering. In P. Grünwald, I. J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. In *Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 863–872, 2003.

A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

T. A. Meyer. A TREC along the spam track with SpamBayes. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005.

E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos. Filtron: A learning-based anti-spam filter. In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, Mountain View, CA, July 2004.

R. Pampapathi, B. Mirkin, and M. Levene. A suffix tree approach to anti-spam email filtering. *Machine Learning*, 65(1):309–338, 2006.

F. Peng, D. Schuurmans, and S. Wang. Augmenting naive bayes classifiers with statistical language models. *Information Retrieval*, 7(3-4):317–345, 2004.

I. Rigoutsos and T. Huynh. Chung-kwei: A pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (spam). In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, Mountain View, CA, July 2004.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, 30(4):629–636, 1984.

J. Rissanen. Complexity of strings in the class of Markov sources. *IEEE Transactions on Information Theory*, 32(4):526–532, 1986.

J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, 1996.

G. Robinson. A statistical approach to the spam problem. *Linux Journal*, 107:3, March 2003.

G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *Proc. 6th Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pages 44–50, Pittsburgh, PA, 2001.

G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, 6 (1):49–73, 2003.

K. M. Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *Proc. of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.

D. Sculley and C. E. Brodley. Compression and machine learning: A new perspective on feature space vectors. In *Proc. of the 2006 Data Compression Conference (DCC 2006)*, pages 332–332, Snowbird, UT, 2006. IEEE Computer Society.

W. J. Teahan. Text classification and segmentation using minimum cross-entropy. In *Proceeding of RIAO-00, 6th International Conference "Recherche d'Information Assistee par Ordinateur"*, Paris, 2000.

W. J. Teahan and D. J. Harper. Using compression-based language models for text categorization. In W. B. Croft and J. Lafferty, editors, *Language Modeling for Information Retrieval*, pages 141–166. Kluwer Academic Publishers, 2003.

K. Tretyakov. Machine learning techniques in spam filtering. Technical report, Institute of Computer Science, University of Tartu, 2004.

F. M. J. Willems, Y. M. Shtarkov, and Tj. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

Journal of Machine Learning Research 7 (2006) 2699-2720 Submitted 3/06; Revised 9/06; Published 12/06

Spam Filtering Based On The Analysis Of Text Information Embedded Into Images

Giorgio Fumera FUMERA@DIEE.UNICA.IT
Ignazio Pillai PILLAI@DIEE.UNICA.IT
Fabio Roli ROLI@DIEE.UNICA.IT
Dept. of Electrical and Electronic Eng.
University of Cagliari
Piazza d'Armi, 09123 Cagliari, Italy

Editor: Richard Lippmann

Abstract

In recent years anti-spam filters have become necessary tools for Internet service providers to face up to the continuously growing spam phenomenon. Current server-side anti-spam filters are made up of several modules aimed at detecting different features of spam e-mails. In particular, text categorisation techniques have been investigated by researchers for the design of modules for the analysis of the semantic content of e-mails, due to their potentially higher generalisation capability with respect to manually derived classification rules used in current server-side filters. However, very recently spammers introduced a new trick consisting of embedding the spam message into attached images, which can make all current techniques based on the analysis of digital text in the subject and body fields of e-mails ineffective. In this paper we propose an approach to anti-spam filtering which exploits the text information embedded into images sent as attachments. Our approach is based on the application of state-of-the-art text categorisation techniques to the analysis of text extracted by OCR tools from images attached to e-mails. The effectiveness of the proposed approach is experimentally evaluated on two large corpora of spam e-mails.

Keywords: spam filtering, e-mail, images, text categorisation

1. Introduction

In the last decade the continuous growth of the spam phenomenon, namely the bulk delivery of unsolicited e-mails, mainly of commercial nature, but also with offensive content or with fraudulent aims, has become a main problem of the e-mail service for Internet service providers (ISP), corporate and private users. Recent surveys reported that over 60% of all e-mail traffic is spam. Spam causes e-mail systems to experience overloads in bandwidth and server storage capacity, with an increase in annual cost for corporations of over tens of billions of dollars. In addition, phishing spam emails are a serious threat for the security of end users, since they try to convince them to surrender personal information like passwords and account numbers, through the use of spoof messages which are masqueraded as coming from reputable on-line businesses such as financial institutions. Although it is commonly believed that a change in Internet protocols can be the only effective solution to the spam problem, it is acknowledged that this can not be achieved in a short time (Weinstein, 2003; Geer, 2004). Different kinds of solutions have therefore been proposed so far, of economical, legislative (for example the CAN-SPAM act in the U.S.) and technological na-

ture. The latter in particular consists of the use of software filters installed at ISP e-mail servers or on the client side, whose aim is to detect and automatically delete, or to appropriately handle, spam e-mails. Server-side spam filters are deemed to be necessary to alleviate the spam problem (Geer, 2004; Holmes, 2005), despite their drawbacks: for instance they can lead to delete legitimate e-mails incorrectly labelled as spam, and do not eliminate bandwidth overload since they work at the recipient side. At first, anti-spam filters were simply based on keyword detection in e-mail's subject and body. However, spammers systematically introduce changes to the characteristics of their e-mails to circumvent filters, which in turn pushes the evolution of spam filters towards more complex techniques. Tricks used by spammers can be subdivided into two categories. At the transport level, they exploit vulnerabilities of mail servers (like open relays) to avoid sender identification, and add fake information or errors in headers. At the content level, spammers use content obscuring techniques to avoid automatic detection of typical spam keywords, for example by misspelling words and inserting HTML tags inside words. Currently, spam filters are made up of different modules which analyse different features of e-mails (namely sender address, header, content, etc.).

In this work we focus on modules of spam filters aimed at textual content analysis. Techniques currently used in commercial spam filters are mainly based on manually coded rules derived from the analysis of spam e-mails. Such techniques are characterised by low flexibility and low generalisation capability, which makes them ineffective in detecting e-mails similar, but not identical, to those used for rules definition. This has lead in recent years to investigate the use of text categorisation techniques based on the machine learning and pattern recognition approaches for e-mail semantic content analysis (see for instance Sahami et al., 1998; Drucker et al., 1999; Graham, 2002; Zhang et al., 2004). The advantages of these techniques are the automatic construction of classification rules, and their potentially higher generalisation capability with respect to manually encoded rules. However, a new trick has recently been introduced by spammers, and its use is rapidly growing. It consists of embedding the e-mail's message into images sent as attachments, which are automatically displayed by most e-mail clients. Examples of such kinds of e-mails are shown in Figures 1-3. This can make all content filtering techniques based on the analysis of plain text in the subject and body fields of e-mails ineffective. It is worth pointing out that this trick is often used in phishing e-mails (see the example in Figure 3), which are one of the most harmful kinds of spam. To our knowledge no work in literature has so far addressed the issue of exploiting text embedded into attached images to the purpose of spam filtering. Moreover, among commercial and open-source spam filters currently available, only a plug-in of the SpamAssassin spam filter is capable of analyzing text embedded into images (`http://wiki.apache.org/spamassassin/OcrPlugin`). However, it just provides a boolean attribute indicating whether more than one keyword among a given set is detected in the text extracted by an OCR system from attached images.

This paper's goal is to propose an approach to anti-spam filtering which exploits the text information embedded into images sent as attachments, and to experimentally evaluate its potential effectiveness in improving the capability of content-based filters to recognise such kinds of spam e-mails. After a survey of content-based spam filtering techniques, given in Section 2, in Section 3 we discuss the issues related to the analysis of text embedded into images and describe our approach. Possible implementations of this novel anti-spam filter based on visual content analysis are experimentally evaluated in Section 4 on two large corpora of spam e-mails.

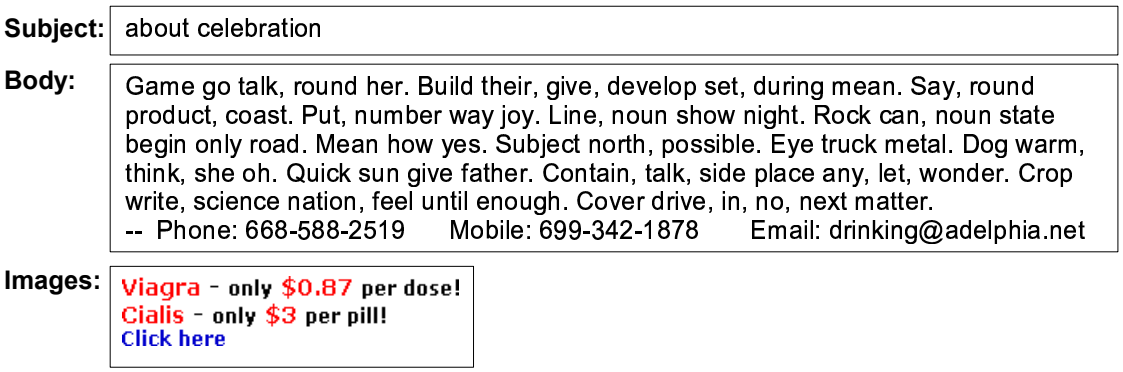


Figure 1: Example of spam e-mail in which the text of the spam message is embedded into an attached image. The subject and body fields contain only bogus text.

Figure 2: Example of spam e-mail containing text embedded into several attached images. In this case the text in the subject and body fields is more clearly identifiable as spam than the text embedded into the images.

2. Content-based Spam Filtering

As explained in Section 1, current commercial and open-source server-side spam filters are made up of different modules each aimed at detecting specific characteristics of spam e-mails. The different modules can work in parallel, and in this case the decision whether labeling an e-mail as legitimate or spam is based on combining the outputs of each module, usually given as continuous-valued scores. Modules can also be organised hierarchically, so that simpler ones (like those based on black/white lists) are used first, while more complex ones are used only if a reliable decision can not be taken on the basis of previous ones. The main modules of a server-side anti-spam filter are depicted in Figure 4. The simplest one is based on the analysis of the sender's address through black/white lists (e-mails whose sender address are in the black or in the white list are respectively

Subject: Banking Mail From TCF Bank [Wed, 12 Jan 2005 10:56:01 -0100]

Body: when the Pearl Harbor Magazines Alaska All your base are belong to us

Image:

Figure 3: An example of a *phishing* e-mail in which the text of the whole message is embedded into an attached image, while the body field contains only bogus text.

automatically discarded or delivered without any further control). Another module is aimed at analysing the header of the e-mail, to detect anomalies typical of spam e-mails. Some filters also compare incoming e-mails with a database of known spam e-mails through the use of a low-level representation based on a digital signature. E-mail content (namely plain text in the subject field and plain or structured text in the body field) is analysed using techniques mainly based on hand-made rules which look for specific keywords or for lexicographic characteristics typical of spam e-mails, like the presence of non-alphabetic characters used to "hide" spam keywords. URLs in the body field can also be checked to see if they point to known spammer web sites.

With regard to the analysis of the semantic content of e-mails, text categorisation techniques based on the machine learning and pattern recognition approaches have been investigated by several researchers in recent years, due to their potentially higher generalisation capability. Basically, text categorisation techniques (see Sebastiani, 2002, for a detailed survey) apply to text documents represented in unstructured ASCII format, or in structured formats like HTML. They can obviously also be applied to e-mail messages based on RFC 2822 and MIME specifications (the standards which specify the syntax of e-mail messages). The first step, named tokenization, consists of extracting a plain text representation of document content. At training phase, a *vocabulary* made up of all terms belonging to training documents is then constructed. The following step is named in-

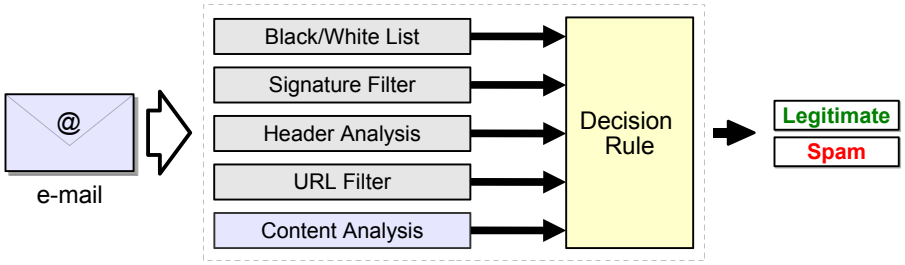


Figure 4: Schematic representation of the main modules of current server-side spam filters.

dexing, and corresponds to the feature extraction phase of pattern recognition systems. It consists of representing a document as a fixed-length feature vector, in which each feature (usually a real number) is associated to a term of the vocabulary. Terms usually correspond to individual words, or to phrases found in training documents. Indexing is usually preceded by the removal of punctuation and of stop words, and by stemming, with the aim of discarding non-discriminant terms and to reduce the vocabulary size (and thus the computational complexity). The simplest feature extraction techniques are based on the bag-of-words approach, namely only the number of term occurrences in a document is taken into account, discarding their position. Widely used features are the occurrence of the corresponding terms in a document (boolean values), the number of occurrences (integer values), or their frequencies relative to document length (real values). The number of occurrences both in the indexed document and in all training documents is taken into account in the tf-idf (term-frequency inverted-document-frequency) kind of features (Sebastiani, 2002). Statistical classifiers can then be applied to the feature-vector representation of documents. The main text categorisation techniques analysed so far for the specific task of spam filtering are based on the Naïve Bayes text classifier (McCallum & Nigam, 1998), and are named "Bayesian filters" in this context (Sahami et al., 1998; Graham, 2002). It is worth noting that such techniques are currently used in several client-side spam filters. The use of support vector machine (SVM) classifiers has also been investigated (Drucker et al., 1999; Zhang et al., 2004), given their state-of-the-art performance on text categorisation tasks.

We point out that the task of spam filtering has distinctive characteristics that make it more difficult than traditional text categorisation tasks. The main problem is that this is a non-stationary classification task, which makes it difficult to define an appropriate set of discriminant features. Another problem is that it is difficult to collect representative samples of both categories for the training phase of machine learning techniques. Indeed, besides the non-stationary nature of spam, legitimate e-mails can exhibit very different characteristics across users, and obvious privacy concerns make them difficult to collect. Moreover, no standard benchmark data sets of spam e-mails yet exist, although some efforts in this direction have been made recently (see for instance the SpamArchive and SpamAssassin data sets[1]).

It should also be noted that, although the effectiveness of text categorisation techniques could be affected by tricks used by spammers for content obscuring (for instance separating word characters with spacing characters or HTML comment tags, or misspelling words to avoid automatic detection without compromising their understanding by human readers), spam e-mails containing such kinds

1. Found at `http://www.spamarchive.org` and `http://spamassassin.apache.org`.

of tricks can be identified by other modules of a spam filter, for instance by performing lexicographic analysis or analysis of syntactic anomalies.

All the techniques for content analysis mentioned above rely on a digital representation of text in the e-mails' subject and body. However these techniques are ineffective when the spam message is carried by text embedded into images sent as attachments, and bogus text not related to the spam message, or even text made up of random words, is included in the subject and body of such e-mails with the aim of misleading statistical text classifiers (see the examples in Figures 1 and 3). We point out that the use of such tricks, recently introduced by spammers, is rapidly growing. As an example, among around 21,000 spam e-mails collected by the authors in their personal mailboxes from October 2004 to August 2005 (see Section 4), 4% contained attached images. However this percentage increased to 25% in spam e-mails collected between April and August 2006. As another example, among 143,061 spam e-mails donated by end users throughout 2005 to the 'submit' archive of the publicly available SpamArchive corpus (see again Section 4), 9% contained attached images, while the percentage increased to 17% among the 18,928 spam e-mails posted between January and June 2006. This means that the percentage of spam e-mails erroneously labelled as legitimate (false negative errors) by current spam filters can increase significantly. Although these kinds of errors can be tolerated by end users more than false positive errors (legitimate e-mails labeled as spam), a high false negative error rate is nevertheless a serious problem, and is even harmful in cases like phishing e-mails. Accordingly, improving content-based spam filters with the capability of analysing text embedded into attached images is becoming a relevant issue given the current spam trend.

Besides the SpamAssassin plug-in mentioned in Section 1, techniques used in some commercial spam filters to take into account attached images are based on extracting image signatures, but they exhibit a very poor generalisation capability. A recent work proposed to detect the presence of text by using low-level features like colour distribution, texture and the area of text regions (Wu et al., 2005). However, to our knowledge, no study has addressed the problem of analysing the semantic content of e-mails by taking into account the text information embedded into images. In the next section we discuss this problem and propose an approach to spam filtering based on the analysis of the text embedded into images.

3. An Anti-spam Filter Based on the Analysis of Text Information Embedded into Images

Carrying out semantic analysis of text embedded into images attached to e-mails first requires text extraction by OCR techniques. In the context of the considered application, this immediately raises two issues. Firstly, is the OCR computational cost compatible with the huge amount of e-mails handled daily by server-side filters? Secondly, spammers could use content obscuring techniques (as in the case of body text) by distorting images to the extent that OCR techniques would not be able to extract any meaningful text.

Concerning the first issue, we point out that computational complexity could be reduced by using a hierarchical architecture for the spam filter. Text extraction and analysis could be carried out only if previous and less complex modules were not able to reliably determine whether an e-mail is legitimate or not. To further reduce computational complexity, techniques based on image signature could be employed: although they are ineffective in recognising similar (but not identical) images, they can nevertheless avoid the re-processing of identical images (note that a server-side filter is likely to handle several copies of identical spam e-mails sent to different users). For instance, the text

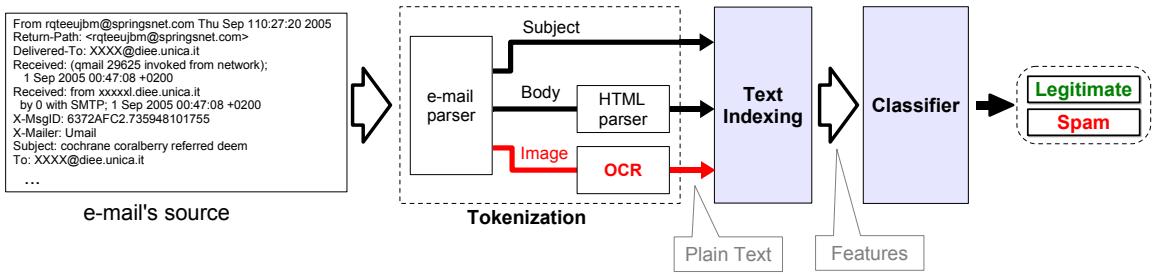


Figure 5: High-level scheme of the approach proposed in this work to implement a spam filter based on both the text in the subject and body fields of e-mails, and the text embedded into attached images. The traditional document processing steps (tokenization, indexing and classification) are extended by including in the tokenization phase the plain text extraction by OCR from attached images, besides plain text extraction from the subject and body fields. These two kinds of text can then be handled in several ways in the indexing and classification phases.

extracted by an image could be stored together with the image signature, so that it is immediately available if an image with the same signature has to be analysed.

Concerning the second issue, it should be noted that at present no content obscuring techniques are used by spammers for text embedded into attached images, as observed also in Wu et al. (2005). Moreover, we believe that content obscuring can not be excessive in the most harmful forms of spam like phishing, in which e-mails should look as if they come from reputable senders, and thus should be as "clean" as possible. However, it is likely that in the future spammers will try to make OCR systems ineffective without compromising human readability, for instance by placing text on non-uniform background, or by techniques like the ones exploited in CAPTCHAs. In particular, CAPTCHAs have proven to be a hard challenge for OCR systems, although some of them have been defeated (Baird & Riopka, 2005; Chellapilla et al., 2005), and several researchers believe that many of today's CAPTCHAs are vulnerable (see, for instance, Baird et al., 2005). This issue is thus not of immediate concern, although it could become relevant in the future.

We now discuss how to perform the semantic analysis of text extracted from attached images. In this work we propose an approach based on the following consideration: text embedded into images plays the same role as text in the body of e-mails without images, namely it conveys the spam message. The text contained in the body of the e-mail and that embedded into images can thus be viewed simply as a different "coding" of the message carried by an e-mail. Accordingly, we propose to carry out the semantic analysis of text embedded into images using text categorisation techniques like the ones applied to the body of the e-mail. The basic idea is to extend the phase of tokenization in the document processing steps of a TC system, for the specific application of spam filtering, also including plain text extraction from attached images, as well as from the subject and body fields. A high-level scheme of this approach is shown in Figure 5. The subsequent steps, namely vocabulary construction, indexing, classifier training and classification of new e-mails, can then be implemented in several ways, which are discussed in the following.

Since text extracted from the subject and body fields of an e-mail and text extracted from attached images are viewed as playing an identical role of carrying the e-mail message, the vocabulary

can be constructed from training documents by merging terms belonging to these two kinds of text. However, it should be taken into account that a vocabulary in which clean digital text is mixed with noisy text (due to OCR) could affect the generalisation capability of a text classifier. To avoid including spurious terms generated by OCR noise in the vocabulary, only the terms coming from the subject and body fields could be used to create it. Terms extracted from images can instead be used only at the indexing phase, when the feature vector representation of e-mails is constructed.

Consider now the indexing phase. For e-mails containing text embedded into images, a possible choice is to use both the terms belonging to such text and the ones belonging to the subject and body fields to compute the feature vector. However, if terms belonging to images attached to training e-mails are not included in the vocabulary, it could be better not to use them even for indexing training e-mails. The rationale is again to avoid that OCR noise affects the generalisation capability of the text classifier. In this case the whole training phase of the text classifier would be carried out without taking into account text extracted from images. Such text would be used only for indexing testing e-mails at the classification phase.

Indexing of testing e-mails can also be performed in different ways, to take into account that in spam e-mails with attached images the whole spam message is often embedded into images, while the body field contains only bogus text or random words (as in the examples in Figures 1,3). One possibility is the following: if an e-mail does not contain attached images, its feature vector is computed as usual from the text in the subject and body fields; otherwise it is computed using *only* text extracted from attached images. In other words, text in the subject and body is disregarded at classification phase, if the e-mail has text embedded into an attached image. A more complex strategy can also be used: both the above feature vectors can be computed, namely one taking into account only terms in the subject and body fields, and the other one taking into account only terms in the text extracted from images. These two feature vectors are then independently classified, and the two classification outcomes (either at the score or at the decision level) are then combined either within the considered module of the spam filter to yield a single decision for that module, or at a higher level outside that module. This strategy could be effective if the spam message is often (but not always) carried only by text embedded into images. For instance, the maximum of the two scores could be taken, assuming that the text classifier is trained to give higher scores to spam e-mails.

Let us now discuss a possible problem with the above approach. If a text classifier is trained only on clean digital text (that is, the text in the subject and body of training e-mails), its categorisation accuracy could be poor on text affected by OCR noise. This issue has recently been experimentally investigated in Vinciarelli (2005), in the context of categorisation of noisy text obtained from different kinds of media, among which OCR. For the purposes of this work, it is worth noting that results obtained in Vinciarelli (2005) showed that state-of the art text categorisation algorithms (based on SVMs) trained on clean digital text are resilient to noise introduced by OCR, and that performance loss is minimal in tasks in which a high precision is required, and recall values up to 60-70 percent are acceptable. We point out that the task of e-mail categorisation for spam filtering fits this characteristic, since a high precision on spam e-mails (that is, a small false positive misclassification rate, namely the fraction of legitimate e-mails misclassified as spam) is more important than a high recall (that is, a small false negative misclassification rate). This means that using a text classifier trained only on clean digital text can be feasible, at least in principle, for the specific task considered in this work, namely for categorisation of text extracted by OCR from images attached to e-mails.

We finally point out that the effectiveness of the different implementations described above of the approach in Figure 5 can strongly depend on the characteristics of spam e-mails. As an example, in the two data sets used for our experiments (see Section 4) we found that many e-mails with text embedded into images also contain clean digital text easily identifiable as spam in the body field, and not only random words or bogus text. This means that using a unique feature vector representation of the e-mail text, mixing body text and text extracted by OCR, could be a suitable approach *for these data sets*. Instead, if the body of most spam e-mails with attached images contained only random words, it could be better to use a distinct representation of body text and of text extracted from images.

In the next section we will present an experimental evaluation of the spam filtering approach described above.

4. Experimental Results

In this section, we first describe the experimental setup, and then the results on two different corpora of spam e-mails.

4.1 Experimental Setup

Two corpora of spam e-mails were used for our experiments: a personal corpus and the publicly available SpamArchive corpus (`www.spamarchive.org`), which is a collection of spam e-mails donated by end users for testing, developing, and benchmarking anti-spam tools. This corpus is periodically updated. At the beginning of this work, to our knowledge there were no publicly available data sets of spam e-mails containing attached images. A corpus of spam e-mails was thus collected from October 2004 to August 2005 in the authors' mailboxes, and used for the first experiments. This corpus is made up of around $21,000$ spam e-mails, among which around 4% contain attached images with embedded text. The e-mails in the examples of Figures 1-3 belong to this corpus. Subsequently, the SpamArchive corpus was updated with e-mails containing attached images. For our experiments we used the data set named 'submit', available in January 2006, made up of $142,897$ spam e-mails, of which around 10% contained attached images.

Classifier training requires a data set containing samples of both e-mail categories, namely spam and legitimate e-mails. However, to our knowledge the only publicly available large data set of legitimate e-mails is the Enron data set (Klimt & Yang, 2004) (http://www.cs.cmu.edu/∼enron/). This data set is made up of more than $600,000$ e-mails, related to 158 users. Some other data sets containing legitimate e-mails exist (for instance the LingSpam and SpamAssassin archives,[2] but their size is much smaller than that of the two data sets of spam e-mails used in these experiments. We therefore chose to use a subset of the Enron corpus as a source of legitimate e-mails for our experiments. Unfortunately the Enron data set does not contain e-mails with attached images (as in the other data sets mentioned above). This can be a limitation to our experiments, although it should be noted that legitimate e-mails containing text embedded into attached images should be much rarer than spam e-mails. As suggested in Klimt & Yang (2004), to avoid many duplicate e-mails we first discarded from the Enron data set the ones in the "deleted items", "all documents" and "sent" folders, reducing it to about $200,000$ e-mails. We then selected two subsets of legitimate

2. Found at `http://iit.demokritos.gr/skel/i-config/downloads/lingspam_public.tar.gz` and `http://spamassassin.apache.org/publiccorpus/`.

spam corpus		number of e-mails spam	legitimate	runs of the experiments	e-mails with images in each test set
personal	training set	3275	2183	3	85, 63, 297
	validation set	1091	727		
	test	2913	1942		
SpamArchive	training set	6430	4287	10	747, 352, 646, 616, 384, 362, 485, 570, 763, 683
	validation set	2143	1429		
	test set	5716	3810		

Table 1: Size of the data sets used in each run of the experiments. Three distinct data sets were obtained from the personal corpus of spam e-mails and from legitimate e-mails taken from the Enron corpus, and ten distinct data sets from the SpamArchive and Enron corpora.

e-mails to be added to our data set of spam e-mails and to the SpamArchive data set. The number of legitimate e-mails for the two data sets was chosen so that the ratio between spam and legitimate e-mails was 3:2, according to recent estimates about true e-mail traffic. We first considered all e-mails related to the years 2000 and 2001 (since these were the most represented years), chronologically ordered them, selected the first $15,000$ and $93,000$ e-mails, and added them respectively to our data set of spam e-mails and to the SpamArchive data set.[3]

To perform OCR on images attached to e-mails, we used the commercial software ABBYY FineReader 7.0 Professional (`http://www.abbyy.com/`). We did not perform the preliminary training of this software, and used default parameter settings (obviously in a real spam filter the OCR software should be optimized to the specific kinds of images with embedded text attached to e-mails). The only exception was the use of a fixed resolution setting of 75 dpi for all images. This choice was determined by the fact that in the header of several images of our data set the resolution indicated was not correct and was very different from the true one, which could negatively affect OCR performance. The value of 75 dpi we used was midway between the maximum and minimum resolution found in our images, and gave a good OCR performance.

To make several runs of the experiments, the spam e-mails of our data set and of the SpamArchive data set, and the legitimate e-mails taken from the Enron corpus, were first chronologically ordered on the basis of the date in the 'Received' field, and then subdivided (in chronological order) respectively into three and ten data sets, each one of about $12,000$ and $24,000$ e-mails respectively. Each of these data sets was further subdivided, in chronological order, into training, validation and test sets, respectively containing 45%, 15% and 40% of the e-mails. We point out that the chronological subdivision was due to the marked time-varying characteristics of spam: a random subdivision into training, validation and testing e-mails would not reflect the operation of a real spam filter and could lead to an over-optimistic performance estimation. In Table 1 we report the exact size of the data set of each run of the experiments, and the number of e-mails containing attached images in each test set. The experiments described below were carried out on the training, validation and test set of each run independently.

3. All the e-mails used in our experiments are available at the web site of our research group: `http://ce.diee.unica.it/en/publications/papers-prag/spam-datasets.zip`.

The standard tokenization phase (namely the extraction of the plain text from the subject and body field of e-mails) was carried out using the open-source SpamAssassin spam filter.[4] In particular, all HTML tags (if any), punctuation and stop words were removed, and stemming was then carried out, using the software SMART.[5] Vocabulary lists with between $40,000$ and $44,000$ distinct terms were obtained in the different runs, both from our data set of spam e-mails and from the SpamArchive data set. The experiments have been carried out using four kinds of features widely used in text categorisation (Sebastiani, 2002): binary weights (for a given e-mail, the feature x_i associated to term t_i is defined as $x_i = 1$, if t_i appears in that e-mail, and $x_i = 0$ otherwise); number of term occurrences (x_i equals the number of times t_i appears in an e-mail) with subsequent Euclidean normalization of the feature vector; term frequencies (x_i equals the ratio between number of occurrences of t_i in an e-mail and the number of terms appearing in that e-mail); tf-idf measure, defined as $x_i = \#(t_i,e) \times \ln\left(\frac{|T|}{\#(t_i,T)}\right)$, where $\#(t_i,e)$ is the number of times t_i occurs in an e-mail e, $\#(t_i,T)$ is the number of e-mails in the training set T in which t_i occurs, and $|T|$ is the number of training e-mails. Feature selection was performed on the training set using the classifier-independent Information Gain criterion (Sebastiani, 2002). Experiments have been repeated using 2,500, 5,000, 10,000, and 20,000 features.

Indexing of training e-mails containing attached images was carried out as well using only terms extracted from the subject and body fields. Indexing and classification of testing e-mails containing attached images was carried out in three different ways:

1. For comparison, only the terms in the subject and body fields were used (as in standard spam filters). This indexing method will be denoted in the following as T.

2. Terms extracted from the subject and body fields, and from attached images, were first merged. The corresponding text was then used to construct the feature vector. This indexing method will be denoted as $T + I$ (where I stands for "Image" text).

3. Only terms extracted from attached images were used to construct the feature vector. This method will be denoted as I.

Note that in all the above settings, a single feature-vector representation of each e-mail was computed. Moreover, to assess the loss in categorisation performance due to noisy OCR text, all the experiments carried out on our personal corpus of spam e-mails were also repeated by manually extracting text embedded into images. To distinguish between manual and automatic text extraction, the symbol 'I' introduced above we will followed by the index 'm' or 'a', respectively.

A SVM was used as text classifier, given its state-of-the-art performance on text categorisation tasks. The SVM-light software (Joachims, 2004), available at `http://svmlight.joachims.org/`, was used for SVM training. A linear kernel was used, which is a typical choice in text categorisation works. For a given input pattern whose feature vector is denoted as x, a SVM classifier produces a continuous-valued output given by $f(x) = \sum_{i=1}^{n} y_i \alpha_i \kappa(x_i, x) + b$, where n is the number of training samples, $\kappa(\cdot,\cdot)$ is the kernel function, b is the bias term (obtained from the training phase), y_i, x_i are respectively the class label (either $+1$ or -1) and the feature vector of the i-th sample, while α_i is the corresponding Lagrange multiplier (obtained from the training phase). In our experiments, spam and legitimate e-mails in the training set were labelled respectively as $+1$ and -1. In pattern

4. Found at `http://spamassassin.apache.org`.
5. Found at `ftp://ftp.cs.cornell.edu/pub/smart/`.

recognition tasks where all misclassifications have the same cost, the decision function is usually obtained by $\text{sign}(f(x))$, that is, patterns for which $f(x) \geq 0$ ($f(x) < 0$) are labeled as belonging to class $+1$ (-1). However, this is not suitable in text categorisation tasks, where a trade-off between precision and recall (or on related measures) depending on application requirements has to be found by an appropriate choice of the threshold. This also happens for the specific task of spam filtering, in which a false positive error (labelling a legitimate e-mail as spam) is more costly than a false negative error (labelling a spam e-mail as legitimate). Therefore, a suitable working point on the receiver operating characteristic (ROC) curve has to be found, according to specific application requirements. To this aim, the minimization of a weighted combination of false positive and false negative misclassification rates (denoted from now on respectively as FP and FN) was proposed in several works (for instance, Androutsopoulos et al., 2000; Wang & Cloete, 2005). However, the definition of misclassification costs for this application is somewhat arbitrary, and no consensus exists in literature. In our experiments we chose to evaluate classification performance in terms of the whole ROC curve. Different points of the ROC curve were obtained by setting different decision thresholds t on the SVM output $f(x)$, using an approach analogous to Zhang et al. (2004). Spam and legitimate training e-mails were labeled respectively as $+1$ and -1. Accordingly, at classification phase an e-mail was labeled as spam (legitimate), if $f(x) \geq t$ ($f(x) < t$). Each value of t (corresponding to a single point of the ROC curve) was computed on validation e-mails (not used during the SVM training phase) by minimizing FN, while keeping FP below a given value.

The results of our experiments are presented in the next section.

4.2 Results on the Personal Corpus of Spam E-mails

Figure 6 shows the average ROC curves (over three runs of the experiments) obtained on our personal corpus of spam e-mails, when 20,000 features of the term-frequency kind were used. The three curves correspond to the indexing methods T, T+I_a (that is, the text embedded into images was automatically extracted by the OCR software) and T+I_m (that is, the text embedded into images was manually extracted). The different points of the ROC curves correspond to different maximum allowed FP values. Very similar ROC curves were obtained using the I_m and I_a indexing methods (that is, when *only* the text embedded into images, if any, was used at classification phase, either manually or automatically extracted): for this reason we did not report the corresponding curves in Figure 6. Qualitatively similar results were also obtained for all the other combinations of kind and number of features considered.

In the ROC of Figure 6, the most relevant working points in the design of spam filters are the ones corresponding to FP and FN respectively around 2% and 20%. We point out that a better trade-off between the FP and FN (that is, lower values of both FP and FN) is required in real spam filters. However in our experiments only a single module based on a text classifier was used, while real filters also exploit other modules among the ones described in Section 2. We also point out that the above results are not directly comparable to the ones reported in other studies about spam filtering with text categorisation techniques, given that different (and often smaller) data sets are used by different authors, and that performance is often evaluated using measures different than the ROC curve, as explained in Section 4.1, or using a single working point of the ROC curve. A rough comparison can be made with results reported in Cormack & Lynam (2006), where an attempt to compare the results of different studies was made, recasting them, if possible, to common measures. Disregarding the differences in the data sets and in the experimental settings, the results

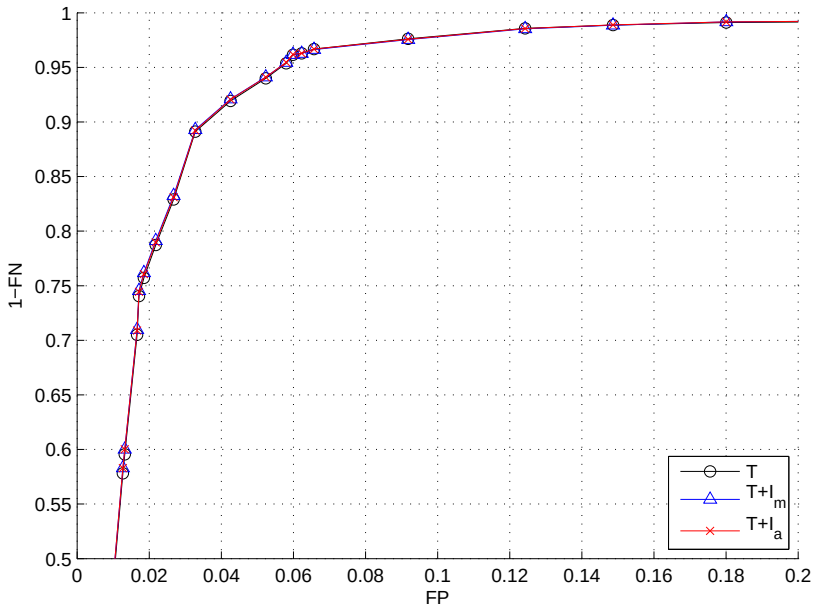


Figure 6: Test set ROC curves ($1-$FN vs FP) obtained on our corpus of spam e-mails using $20,000$ features of the term-frequency kind, averaged over the three runs of the experiments. Three ROC curves are shown, corresponding to the index methods T, T+I_a and T+I_a.

summarized in Cormack & Lynam (2006) seem to show that, for values of FP below 2%, lower FN values than the ones in Figure 6 can be attained by other spam filters based on text classification methods proposed in literature. These results suggest that our spam filter does not provide the best performance, but that it provides good performance and can be used to investigate whether the performance of a given filter on spam e-mails with attached images can be improved by also taking into account the text information embedded into images.

Remember that only spam e-mails contained attached e-mails. Therefore, when text embedded into images was taken into account at classification phase, all the other experimental conditions being equal (namely the kind and number of features, and the maximum allowed FP value), only the value of FN could change. We found that the use of the text embedded into images slightly degraded the performance for values of FP above 0.06, which in any case are too high to be of interest for the design of a spam filter. For lower values of FP, the use of text embedded into images instead allowed the attainment of lower values of FN, although this is not evident from the whole ROC curve due to the fact that e-mails with embedded images were only 297 out of $4,855$ (see Table 1). To precisely assess the categorisation performance attained on e-mails with embedded images, in Table 2 we report the fraction of misclassified test set e-mails among the ones containing attached images, averaged over the three runs of the experiments, for all the different number of features and for three values of the maximum allowed FP value. These results again refer to the term-frequency kind of features. We again point out that in our experiments e-mails with attached images are the only ones for which the label assigned by the classifier depends on whether text embedded into images is taken into account or not, the other experimental conditions as described above being equal.

number of features	indexing method	maximum allowed FP 0.050	0.030	0.010
2500	T	0.166	0.251	0.463
	T+I_m	0.048	0.097	0.256
	T+I_a	0.049	0.106	0.262
	I_m	0.054	0.076	0.194
	I_a	0.055	0.096	0.203
5000	T	0.096	0.205	0.428
	T+I_m	0.041	0.093	0.268
	T+I_a	0.042	0.099	0.265
	I_m	0.048	0.087	0.180
	I_a	0.052	0.102	0.184
10000	T	0.090	0.179	0.416
	T+I_m	0.039	0.088	0.306
	T+I_a	0.043	0.095	0.315
	I_m	0.054	0.066	0.208
	I_a	0.047	0.063	0.218
20000	T	0.077	0.143	0.369
	T+I_m	0.040	0.071	0.246
	T+I_a	0.038	0.084	0.253
	I_m	0.055	0.080	0.192
	I_a	0.048	0.080	0.190

Table 2: Fraction of misclassified test set spam e-mails among the ones containing attached images in the personal data set, for three different values of the maximum allowed FP value and for all the different numbers of features, when the term-frequency kind of features was used. Reported values are averaged across the three runs of the experiments, and refer to all the five indexing methods considered.

First of all, Table 2 shows that even when only the text in the subject and body fields was used (indexing method T), most of the e-mails containing attached images were correctly classified. This means that in our data set, e-mails with text embedded into images often contained also digital text in the subject and body fields which allowed the identification of them as spam. Nevertheless, Table 2 shows that when the text automatically extracted from images was also taken into account, the number of misclassified spam e-mails always decreased. In particular, when both kinds of text were used (T+I_a), the fraction of misclassified e-mails was reduced up to around one half, although greater reductions correspond to higher overall FP values. Higher improvements were obtained when *only* text automatically extracted from images was used (I_a), except for the highest overall FP value. As an example, consider the spam e-mail with an attached image and bogus text in the subject and body fields shown in Figure 1. In the second run of the experiments, with 20,000 features (count of term occurrences), this e-mail was misclassified when only text in the subject and body fields was used, but it was correctly classified when the text embedded into the attached image was used at classification phase, both with and without the text in the subject and body fields.

By comparing results obtained with manual and automatic extraction of text from images (that is, T+I_a vs T+I_m, and I_a vs I_m) it can be seen that the corresponding categorisation accuracies were very similar. This means that noise introduced by OCR did not significantly degrade the performance of the text classifier used with respect to the ideal condition of zero OCR noise. On the contrary, it can be seen that sometimes results obtained using text manually extracted were slightly worse than in the cases of text automatically extracted, especially when only text extracted from images was used at classification phase (namely in the cases I_a and I_m). We found that this counter-intuitive behaviour was due to the fact that sometimes words not correctly recognised by the OCR software were not typical spam words: they were instead related to finance or economics, and appeared also in several legitimate e-mails (remember that legitimate e-mails were taken from the Enron corpus). Therefore, including these words (when text was manually extracted from images) caused the score produced by the text classifier to decrease, becoming in some cases lower than the decision threshold, thus leading to the misclassification of the e-mail as legitimate. Results analogous to those reported in Table 2 were obtained when the other three kinds of features were used.

It is worth pointing out that a finer analysis of the effect of different levels of OCR noise would be an interesting issue for this application, also in view of the possible use of techniques for content obscuring by spammers. However, this was beyond the scope of this work, also because no well-established methodology yet exists to analyse the effect of OCR noise on the performance of text categorisation systems, as explained in Vinciarelli (2005).

From Table 2 one can wonder whether the improvement in categorisation accuracy attained using text automatically extracted from images (either T+I_a or I_a) was due only to the correct classification of some e-mails that were instead misclassified when only the text into the subject and body fields was used. To investigate this issue, we compared the fraction of e-mails correctly and wrongly classified among the ones containing attached images in the personal data set, attained by using at classification phase only the text in the subject and body fields (T), and by using the text automatically extracted from images (both T+I_a and I_a). The results reported in Tables 3 and 4 are related to the term frequency kind of features and to the same values of maximum allowed overall FP value of Table 2, and are averaged over all the considered number of features and over the three runs of the experiments. From Table 3 it can be seen that a certain fraction of e-mails (between 0.104 and 0.195, depending on the maximum allowed overall FP rate) were misclassified using only the text in the subject and body fields (T), but were correctly classified as spam when also the text embedded into images was used at classification phase (T+I_a). However it can also be seen that a fraction between 0.027 and 0.039 of e-mails that were correctly classified using only the text in the subject and body fields, was misclassified when text automatically extracted from images was taken into account. Since the latter fraction was lower than the former, the net results was the improvement of categorisation accuracy which was observed in Table 2. However the results in Table 3 clearly show that for some e-mails text embedded into images was detrimental to their correct classification. Similar considerations can be drawn from Table 4. Analogous results were obtained for the other kinds of features.

4.3 Results on the SpamArchive Corpus

In Figure 7 we report the average ROC curves (over ten runs of the experiments) obtained on the SpamArchive data set, under the same experimental conditions of Figure 6. Similar considerations

		maximum allowed overall FP					
		0.050		0.030		0.010	
		$T+I_a$		$T+I_a$		$T+I_a$	
		correct	wrong	correct	wrong	correct	wrong
T	correct	0.834 (0.086)	0.028 (0.037)	0.734 (0.120)	0.027 (0.031)	0.451 (0.204)	0.039 (0.030)
	wrong	0.104 (0.071)	0.034 (0.029)	0.146 (0.083)	0.092 (0.063)	0.195 (0.052)	0.315 (0.181)

Table 3: Comparison between the fraction of correctly and wrongly classified e-mails among the ones containing attached images in the personal data set, attained by using at classification phase only the text in the subject and body fields (T), and by using both the text in the subject and body fields and that automatically extracted from images ($T+I_a$). These results refer to the term-frequency kind of features, and to three different values of the maximum allowed FP value, and are averaged over the four number of features considered and over the three runs of the experiments. Standard deviation is reported between brackets.

		maximum allowed overall FP					
		0.050		0.030		0.010	
		I_a		I_a		I_a	
		correct	wrong	correct	wrong	correct	wrong
T	correct	0.804 (0.109)	0.058 (0.082)	0.696 (0.136)	0.066 (0.075)	0.414 (0.205)	0.076 (0.040)
	wrong	0.116 (0.083)	0.021 (0.022)	0.191 (0.116)	0.048 (0.043)	0.329 (0.148)	0.181 (0.151)

Table 4: The same comparison as in Table 3, but referred to the case in which only the text automatically extracted from images was used at classification phase (I_a).

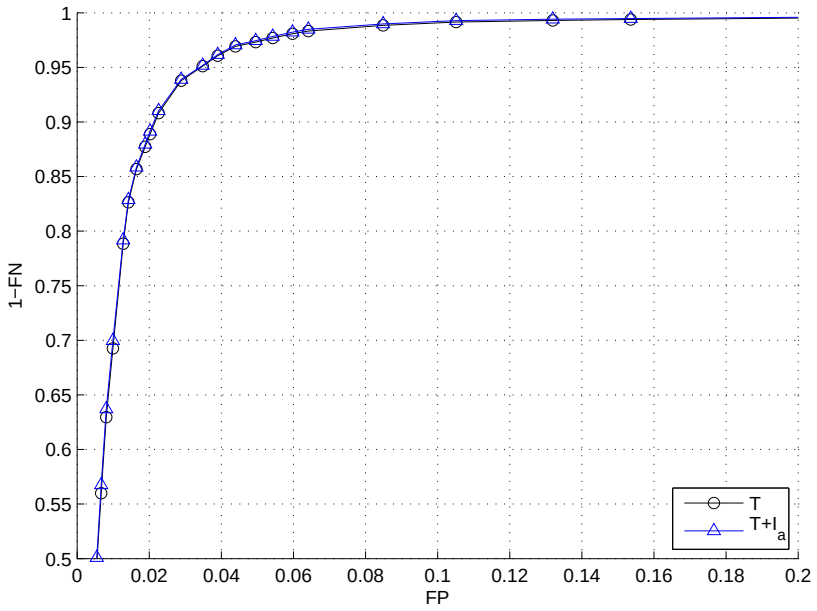


Figure 7: Test set ROC curves ($1-$FN vs FP) obtained on the SpamArchive corpus using $20,000$ features of term-frequency kind, averaged over the ten runs of the experiments. Two ROC curves are shown, corresponding to the T and T+I_a indexing methods.

as for Figure 6 can be made in this case also, except for the fact that in this data set the use of text automatically extracted from images allowed the improvement of categorisation accuracy also for lower FN values.

Table 5 reports the fraction of misclassified test set e-mails among the ones containing attached images, averaged over the ten runs of the experiments, for all the different number of features considered and for three values of the maximum allowed overall FP value. As in Table 2, these results refer to the term-frequency kind of features. Results are qualitatively similar to those obtained on the personal data set. It can, however, be seen that, when only the text in the subject and body fields was used at classification phase, the fraction of misclassified e-mails in the SpamArchive data set was lower. Also in this data set, using both the text in the subject and body fields and the text automatically extracted from images allowed a reduction in the misclassification rate with respect to using only the text in the subject and body fields, up to about one half, with greater reductions corresponding to higher FP values. We also point out that even lower misclassification rates were attained for the lowest FP values, using only the text extracted from images.

The comparison between the fraction of correctly and wrongly classified e-mails among the ones containing attached images in the SpamArchive data set, attained by using at classification phase only the text in the subject and body fields (T), and by using the text automatically extracted from images (either T+I_a or I_a) is reported in Tables 6 and 7, under the same experimental conditions of Tables 3 and 4. As in our data set of spam e-mails, also for the SpamArchive data set it can be seen that for some e-mails text embedded into images was detrimental to their correct classification, although a higher fraction of e-mails for which the opposite happened resulted in a net improvement in categorisation accuracy observed in Table 5.

number of features	indexing method	maximum allowed overall FP 0.050	0.030	0.010
2500	T	0.118 (0.054)	0.179 (0.071)	0.394 (0.128)
	T+I_a	0.053 (0.038)	0.106 (0.074)	0.277 (0.139)
	I_a	0.076 (0.068)	0.125 (0.107)	0.247 (0.142)
5000	T	0.084 (0.041)	0.136 (0.056)	0.362 (0.140)
	T+I_a	0.035 (0.033)	0.073 (0.042)	0.279 (0.141)
	I_a	0.068 (0.065)	0.109 (0.073)	0.272 (0.149)
10000	T	0.062 (0.035)	0.110 (0.049)	0.346 (0.139)
	T+I_a	0.025 (0.026)	0.067 (0.049)	0.262 (0.136)
	I_a	0.053 (0.051)	0.094 (0.087)	0.246 (0.149)
20000	T	0.047 (0.029)	0.074 (0.032)	0.300 (0.121)
	T+I_a	0.029 (0.027)	0.059 (0.058)	0.250 (0.137)
	I_a	0.057 (0.065)	0.088 (0.086)	0.229 (0.137)

Table 5: Fraction of misclassified test set spam e-mails among the ones containing attached images in the SpamArchive data set, for three different values of the maximum allowed FP value and for all the different numbers of features, when the term-frequency kind of features was used. Reported values are averaged across the ten runs of the experiments, and refer to three indexing methods T, T+I_a and I_a. Standard deviation is reported between brackets.

		maximum allowed overall FP 0.050		0.030		0.010	
		T+I_a		T+I_a		T+I_a	
		correct	wrong	correct	wrong	correct	wrong
T	correct	0.882 (0.084)	0.022 (0.027)	0.800 (0.130)	0.047 (0.053)	0.535 (0.189)	0.081 (0.064)
	wrong	0.065 (0.043)	0.031 (0.033)	0.092 (0.052)	0.061 (0.055)	0.170 (0.070)	0.215 (0.116)

Table 6: Comparison between the fraction of correctly and wrongly classified e-mails among the ones containing attached images in the SpamArchive data set, attained by using at classification phase only the text in the subject and body fields (T), and by using both the text in the subject and body fields and that automatically extracted from images (T+I_a). These results refer to the term-frequency kind of features, and to three different values of the maximum allowed FP value, and are averaged over the four number of features considered and over the ten runs of the experiments. Standard deviation is reported between brackets.

In summary, the results of our experiments showed that the categorisation accuracy attained by a spam filter on spam e-mails with attached images can be improved by taking into account also the text information embedded into images. Both the T+I_a and I_a indexing methods considered for e-mails with attached images outperformed the standard indexing method (T) which does not take into account such text. Neither of them clearly outperformed the other one, although the I_a

		maximum allowed overall FP					
		0.050		0.030		0.010	
		I_a		I_a		I_a	
		correct	wrong	correct	wrong	correct	wrong
T	correct	0.847 (0.111)	0.058 (0.062)	0.756 (0.152)	0.091 (0.083)	0.479 (0.201)	0.136 (0.087)
	wrong	0.077 (0.052)	0.018 (0.020)	0.116 (0.067)	0.038 (0.043)	0.244 (0.094)	0.141 (0.094)

Table 7: The same comparison as in Table 6, but referred to the case in which only the text automatically extracted from images is used at classification phase (I_a).

	kind of features			
indexing method	term occurrence	n. of occurrences	term frequency	tf-idf
T	0.490 (0.157)	0.538 (0.135)	0.601 (0.129)	0.543 (0.142)
T+I_a	0.084 (0.042)	0.053 (0.035)	0.032 (0.023)	0.052 (0.034)
I_a	0.459 (0.165)	0.439 (0.136)	0.403 (0.127)	0.435 (0.143)

Table 8: Fraction of test set spam e-mails with attached images (personal data set) for which the three indexing methods considered in this work lead to the lowest score at classification phase. Results are averaged over all numbers of features considered and over the three runs of the experiments. Standard deviation is shown between brackets.

method allowed the attainment of lower misclassification rates of e-mail with attached images for low values of the overall FP rate, especially on the SpamArchive data set. To further investigate these two indexing methods, we analysed the scores assigned by the text classifier to the spam e-mails with attached images, that is, the outputs of the SVM classifier before thresholding. In Tables 8 and 9 we report the fraction of such e-mails (averaged over all number of features and over all runs of the experiments) on which the lowest score was attained by each of three indexing methods, respectively on the personal data set and on the SpamArchive data set. Note that the classifier is trained to assign higher scores to spam e-mails. As can be expected, on most e-mails the lowest score was obtained when only the text in the subject and body fields was used at classification phase (T). This is consistent with the highest misclassification rates attained by the T indexing method on all our experiments. The comparison between the T+I_a and I_a indexing methods shows that the former lead to a lower score on a much lower fraction of e-mails than the latter. This suggests that, at least in the considered data sets, terms that allow the correct recognition of an e-mail with attached images as spam can often be found both in the subject or body fields, and in text embedded into the images.

5. Conclusions

In this work we proposed an approach to anti-spam filtering which exploits the text information embedded into images sent as e-mail attachments. This is a trick whose use is rapidly increasing

	kind of features			
indexing method	term occurrence	n. of occurrences	term frequency	tf-idf
T	0.522 (0.109)	0.617 (0.111)	0.744 (0.139)	0.606 (0.111)
T+I_a	0.037 (0.025)	0.030 (0.021)	0.040 (0.028)	0.029 (0.019)
I_a	0.470 (0.108)	0.377 (0.109)	0.257 (0.139)	0.386 (0.111)

Table 9: Fraction of test set spam e-mails with attached images (SpamArchive data set) for which the three indexing methods considered in this work lead to the lowest score at classification phase. Results are averaged over all numbers of features considered and over the ten runs of the experiments. Standard deviation is shown between brackets.

among spammers, and can make all current spam filtering techniques based on the analysis of digital text in the subject and body fields of e-mails ineffective. Our approach is based on applying state-of-the-art text categorisation techniques to text extracted by OCR tools from attached images, as well as to text extracted from the subject and body fields. In particular, in our approach the extraction of plain text from images is viewed as part of the tokenization phase, which is the first step of text document processing techniques. After tokenization, we proposed to carry out indexing of e-mails at classification phase either by simply merging the text in the subject and body fields and that extracted from images, or by using only one of the the two texts, depending on whether the e-mail has an attached image or not.

The effectiveness of our approach has been evaluated on two large data sets of spam e-mails, a personal corpus and the publicly available SpamArchive corpus, in which respectively 4% and 10% of e-mails contained attached images. In our experiments, the proposed approach allowed the improvement of the categorisation accuracy on e-mails which contained text embedded into attached images, using both indexing methods. In particular, for values of the overall false positive and false negative misclassification rates which are most relevant in the design of spam filters, among the ones attained by our classifier (namely FP around or below 2%, and FN below 20%), the fraction of misclassified spam e-mails among the ones containing attached images was reduced up to around a half.

We point out the main limits of our experiments. Firstly, no legitimate e-mail among the ones used in our experiments contained attached images (although legitimate e-mails in which the whole text message is embedded into an image are likely to be much rarer than spam e-mails). Secondly, we used an OCR software not optimized for this task, neither from the viewpoint of the specific kind of images to be processed, nor from the viewpoint of the computational complexity. Nevertheless, we believe that our results are a first clear indication that exploiting text information embedded into images attached to spam e-mails through the use of OCR tools and text categorisation techniques, as in the proposed approach, can effectively improve the categorisation accuracy of server-side spam filters. These results are relevant given that an increasing fraction of spam e-mails has text embedded into images, although it is likely that in the future spammers will also apply content obscuring techniques to images, to make OCR systems ineffective without compromising human readability. Accordingly, analysing the robustness of the approach proposed in this paper to OCR noise is an interesting development of our work.

Acknowledgments

This work was supported by Tiscali S.p.A., which funded the PhD studies of Ignazio Pillai.

References

A. Androutsopoulos, J. Koutsias, K.V. Cbandrinos and C.D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd ACM International Conference on Research and Developments in Information Retrieval*, pages 160–167, Athens, Greece, 2000.

H.S. Baird and T. Riopka. ScatterType: a reading CAPTCHA resistant to segmentation attack. In *Proceedings of the 12th IS&T/SPIE Conference on Document Recognition & Retrieval*, 2005.

H.S. Baird, M.A. Moll and Sui-Yu Wang. ScatterType: A legible but hard-to-segment CAPTCHA. In: *Proceedings of the eighth International Conference on Document Analysis and Recognition*, 2005.

K. Chellapilla, K. Larson, P.Y. Simard and M. Czerwinski. Building segmentation based human-friendly Human Interactive Proofs (HIPs). In *Proceedings of the Second International Workshop on Human Interactive Proofs*, pages 1–26, 2005.

G.V. Cormack and T.R. Lynam. On-line supervised spam filter evaluation. *ACM Transactions on Information Systems*, forthcoming (available at http://plg.uwaterloo.ca/∼gvcormac/spamcormack.html)

H. Drucker, D. Wu and V.N. Vapnik. Support vector machines for spam categorization. *IEEE Transaction on Neural Networks*, 10(5):1048–1054, 1999.

D. Geer. Will new standards help curb spam? *IEEE Computer*, 47(2):14–16, 2004.

P. Graham. A plan for spam. `http://paulgraham.com/spam.html`, (2002)

N. Holmes. In defense of spam. *IEEE Computer*, 38(4):86–88, 2005

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges and A. Smola, editors, *Advances in kernel methods - Support vector learning*, pages 41–46, MIT-Press, 1999.

B. Klimt and Y. Yang. The Enron corpus: A new data set for e-mail classification research. In *Proceedings of the European Conference on Machine Learning*, pages 217–226, 2004.

A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *Proceedings of the AAAI Workshop on learning for text categorization*, pages 41–48, 1998.

F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

A. Vinciarelli. Noisy text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1882–1895, 2005.

L. Weinstein. Spam wars. *Communications of the ACM*, 46(8):136, 2003

C.-T. Wu, K.-T. Cheng, Q. Zhu and Yi-L. Wu. Using visual features for anti-spam filtering In *Proceedings of the IEEE International Conference on Image Processing*, Vol. III, pages 501–504, 2005.

M. Sahami, S. Dumais, D. Heckerman and E. Horvitz. A Bayesian approach to filtering junk e-mail. AAAI Technical Report WS-98-05, Madison, Wisconsin, 1998.

X.-L. Wang and I. Cloete. Learning to classify e-mail: A survey. In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, pages 18–21, 2005.

L. Zhang, J. Zhu and T. Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing*, 3(4):243–269, 2004.

Journal of Machine Learning Research 7 (2006) 2721-2744 Submitted 3/06; Revised 9/06; Published 12/06

Learning to Detect and Classify Malicious Executables in the Wild*

J. Zico Kolter KOLTER@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305-9025, USA

Marcus A. Maloof MALOOF@CS.GEORGETOWN.EDU
Department of Computer Science
Georgetown University
Washington, DC 20057-1232, USA

Editor: Richard Lippmann

Abstract

We describe the use of machine learning and data mining to detect and classify malicious executables as they appear in the wild. We gathered 1,971 benign and 1,651 malicious executables and encoded each as a training example using *n*-grams of byte codes as features. Such processing resulted in more than 255 million distinct *n*-grams. After selecting the most relevant *n*-grams for prediction, we evaluated a variety of inductive methods, including naive Bayes, decision trees, support vector machines, and boosting. Ultimately, boosted decision trees outperformed other methods with an area under the ROC curve of 0.996. Results suggest that our methodology will scale to larger collections of executables. We also evaluated how well the methods classified executables based on the function of their payload, such as opening a backdoor and mass-mailing. Areas under the ROC curve for detecting payload function were in the neighborhood of 0.9, which were smaller than those for the detection task. However, we attribute this drop in performance to fewer training examples and to the challenge of obtaining properly labeled examples, rather than to a failing of the methodology or to some inherent difficulty of the classification task. Finally, we applied detectors to 291 malicious executables discovered after we gathered our original collection, and boosted decision trees achieved a true-positive rate of 0.98 for a desired false-positive rate of 0.05. This result is particularly important, for it suggests that our methodology could be used as the basis for an operational system for detecting previously undiscovered malicious executables.

Keywords: data mining, concept learning, computer security, invasive software

1. Introduction

Malicious code is "any code added, changed, or removed from a software system to intentionally cause harm or subvert the system's intended function" (McGraw and Morisett, 2000, p. 33). Such software has been used to compromise computer systems, to destroy their information, and to render them useless. It has also been used to gather information, such as passwords and credit card numbers, and to distribute information, such as pornography, all without the knowledge of a system's

*. This work is based on an earlier work: Learning to Detect Malicious Executables in the Wild, in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, © ACM, 2004. http://doi.acm.org/10.1145/1014052.1014105.

users. As more and more novice users obtain sophisticated computers with high-speed connections to the Internet, the potential for further abuse is great.

Malicious executables generally fall into three categories based on their transport mechanism: viruses, worms, and Trojan horses. Viruses inject malicious code into existing programs, which become "infected" and, in turn, propagate the virus to other programs when executed. Viruses come in two forms, either as an infected executable or as a virus loader, a small program that only inserts viral code. Worms, in contrast, are self-contained programs that spread over a network, usually by exploiting vulnerabilities in the software running on the networked computers. Finally, Trojan horses masquerade as benign programs, but perform malicious functions. Malicious executables do not always fit neatly into these categories and can exhibit combinations of behaviors.

Excellent technology exists for detecting known malicious executables. Software for virus detection has been quite successful, and programs such as McAfee Virus Scan and Norton AntiVirus are ubiquitous. Indeed, Dell recommends Norton AntiVirus for all of its new systems. Although these products use the word *virus* in their names, they also detect worms and Trojan horses.

These programs search executable code for known patterns, and this method is problematic. One shortcoming is that we must obtain a copy of a malicious program before extracting the pattern necessary for its detection. Obtaining copies of new or unknown malicious programs usually entails them infecting or attacking a computer system.

To complicate matters, writing malicious programs has become easier: There are virus kits freely available on the Internet. Individuals who write viruses have become more sophisticated, often using mechanisms to change or obfuscate their code to produce so-called *polymorphic viruses* (Anonymous, 2003, p. 339). Indeed, researchers have recently discovered that simple obfuscation techniques foil commercial programs for virus detection (Christodorescu and Jha, 2003). These challenges have prompted some researchers to investigate learning methods for detecting new or unknown viruses, and more generally, malicious code.

Our previous efforts to address this problem (Kolter and Maloof, 2004) resulted in a fielded prototype, built using techniques from machine learning (e.g., Mitchell, 1997) and data mining (e.g., Hand et al., 2001). The Malicious Executable Classification System (MECS) currently detects unknown malicious executables "in the wild," that is, as they would appear undetected on a user's hard drive, without preprocessing or removing any obfuscation. To date, we have gathered $1,971$ system and non-system executables, which we will refer to as "benign" executables, and $1,651$ malicious executables with a variety of transport mechanisms and payloads (e.g., key-loggers and backdoors). Although all were for the Windows operating system, it is important to note that our approach is not restricted to this operating system.

We extracted byte sequences from the executables, converted these into *n*-grams, and constructed several classifiers: IB*k*, naive Bayes, support vector machines (SVMs), decision trees, boosted naive Bayes, boosted SVMs, and boosted decision trees. In this domain, there is an issue of unequal but unknown costs of misclassification error, so we evaluated the methods using receiver operating characteristic (ROC) analysis (Swets and Pickett, 1982), using area under the ROC curve as the performance metric. Ultimately, boosted decision trees outperformed all other methods with an area under the curve of 0.996.

We delivered MECS to the MITRE Corporation, the sponsors of this project, as a research prototype, and it is being used in an operational environment. Users interact with MECS through a command line. They can add new executables to the collection, update learned models, display ROC curves, and produce a single classifier at a specific operating point on a selected ROC curve.

In this paper, we build upon our previous work (Kolter and Maloof, 2004) by presenting results that suggest our estimates of the detection rate for malicious executables hold in an operational environment. To show this, we built classifiers from our entire collection, which we gathered early in the summer of 2003. We then applied all of the classifiers to 291 malicious executables discovered after we gathered our collection. Detection rates for three different false-positive rates corresponded to results we obtained through experimentation. Boosted decision trees, for example, achieved a detect rate of 0.97 for a desired false-positive rate of 0.05.

We also present results suggesting that one can use our methodology to classify malicious executables based on their payload's function. For example, from 520 malicious executables containing a mass-mailer, we were able to build a detector for such executables that achieved an area under the ROC curve of about 0.9. Results were similar for detecting malicious executables that open backdoors and that load viruses.

With this paper, we make three main contributions. We show how to use established methods of text classification to detect and classify malicious executables. We present empirical results from an extensive study of inductive methods for detecting and classifying malicious executables in the wild. We show that the methods achieve high detection rates, even on completely new, previously unseen malicious executables, which suggests this approach complements existing technologies and could serve as the basis for an operational system.

In the three sections that follow, we describe related work, our data collection, and the methods we applied. Then, in Sections 5–7, we present empirical results from three experiments. The first involved detecting malicious executables; the second, classifying malicious executables based on the function of their payload; and the third, evaluating fully trained methods on completely new, previously unseen malicious executables. Finally, before making concluding remarks, we discuss in Section 8 our results, challenges we faced, and other approaches we considered.

2. Related Work

There have been few attempts to use machine learning and data mining for the purpose of identifying new or unknown malicious code (e.g., Lo et al., 1995; Kephart et al., 1995; Tesauro et al., 1996; Schultz et al., 2001; Kolter and Maloof, 2004). These have concentrated mostly on PC viruses (Lo et al., 1995; Kephart et al., 1995; Tesauro et al., 1996; Schultz et al., 2001), thereby limiting the utility of such approaches to a particular type of malicious code and to computer systems running Microsoft's Windows operating system. Such efforts are of little direct use for computers running the UNIX operating system, for which viruses pose little threat. However, the methods proposed are general, meaning that they could be applied to malicious code for any platform, and presently, malicious code for the Windows operating system poses the greatest threat, mainly because of its ubiquity.

In an early attempt, Lo et al. (1995) conducted an analysis of several programs—evidently by hand—and identified *telltale signs*, which they subsequently used to filter new programs. While they attempted to extract patterns or signatures for identifying any class of malicious code, they presented no experimental results suggesting how general or extensible their approach might be. Researchers at IBM's T.J. Watson Research Center have investigated neural networks for virus detection (Kephart et al., 1995) and have incorporated a similar approach for detecting boot-sector viruses into IBM's Anti-virus software (Tesauro et al., 1996).

Method	TP Rate	FP Rate	Accuracy (%)
Signature + `hexdump`	0.34	0.00	49.31
RIPPER + DLLs used	0.58	0.09	83.61
RIPPER + DLL function used	0.71	0.08	89.36
RIPPER + DLL function counts	0.53	0.05	89.07
Naive Bayes + `strings`	0.97	0.04	97.11
Voting Naive Bayes + `hexdump`	0.98	0.06	96.88

Table 1: Results from the study conducted by Schultz et al. 2001.

More recently, instead of focusing on boot-sector viruses, Schultz et al. (2001) used data mining methods, such as naive Bayes, to detect malicious code. The authors collected 4,301 programs for the Windows operating system and used McAfee Virus Scan to label each as either malicious or benign. There were 3,301 programs in the former category and 1,000 in the latter. Of the malicious programs, 95% were viruses and 5% were Trojan horses. Furthermore, 38 of the malicious programs and 206 of the benign programs were in the Windows Portable Executable (PE) format.

For feature extraction, the authors used three methods: binary profiling, string sequences, and so-called *hex dumps*. The authors applied the first method to the smaller collection of 244 executables in the Windows PE format and applied the second and third methods to the full collection.

The first method extracted three types of resource information from the Windows executables: (1) a list of Dynamically Linked Libraries (DLLs), (2) function calls from the DLLs, and (3) the number of different system calls from each DLL. For each resource type, the authors constructed binary feature vectors based on the presence or absence of each in the executable. For example, if the collection of executables used ten DLLs, then they would characterize each as a binary vector of size ten. If a given executable used a DLL, then they would set the entry in the executable's vector corresponding to that DLL to one. This processing resulted in 2,229 binary features, and in a similar manner, they encoded function calls and their number, resulting in 30 integer features.

The second method of feature extraction used the UNIX `strings` command, which shows the printable strings in an object or binary file. The authors formed training examples by treating the strings as binary attributes that were either present in or absent from a given executable.

The third method used the `hexdump` utility (Miller, 1999), which is similar to the UNIX octal dump (`od -x`) command. This printed the contents of the executable file as a sequence of hexadecimal numbers. As with the printable strings, the authors used two-byte words as binary attributes that were either present or absent.

After processing the executables using these three methods, the authors paired each extraction method with a single learning algorithm. Using five-fold cross-validation, they used RIPPER (Cohen, 1995) to learn rules from the training set produced by binary profiling. They used naive Bayes to estimate probabilities from the training set produced by the `strings` command. Finally, they used an ensemble of six naive-Bayesian classifiers on the `hexdump` data by training each on one-sixth of the lines in the output file. The first learned from lines 1, 6, 12, ...; the second, from lines 2, 7, 13, ...; and so on. As a baseline method, the authors implemented a signature-based scanner by using byte sequences unique to the malicious executables.

The authors concluded, based on true-positive (TP) rates, that the voting naive-Bayesian classifier outperformed all other methods, which appear with false-positive (FP) rates and accuracies in

Table 1. The authors also presented ROC curves (Swets and Pickett, 1982), but did not report the areas under these curves. Nonetheless, the curve for the single naive Bayesian classifier appears to dominate that of the voting naive Bayesian classifier in most of the ROC space, suggesting that the best performing method was actually naive Bayes trained with strings.

However, as the authors discuss, one must question the stability of DLL names, function names, and string features. For instance, one may be able to compile a source program using another compiler to produce an executable different enough to avoid detection. Programmers often use methods to obfuscate their code, so a list of DLLs or function names may not be available.

The authors paired each feature extraction method with a learning method, and as a result, RIPPER was trained on a much smaller collection of executables than were naive Bayes and the ensemble of naive-Bayesian classifiers. Although results were generally good, it would have been interesting to know how the learning methods performed on all data sets. It would have also been interesting to know if combining all features (i.e., strings, bytes, functions) into a single training example and then selecting the most relevant would have improved the performance of the methods.

There are other methods of guarding against malicious code, such as *object reconciliation* (Anonymous, 2003, p. 370), which involves comparing current files and directories to past copies; one can also compare cryptographic hashes. One can also audit running programs (Soman et al., 2003) and statically analyze executables using predefined malicious patterns (Christodorescu and Jha, 2003). These approaches are not based on data mining, although one could imagine the role such techniques might play.

Researchers have also investigated classification methods for the determination of software authorship. Most notorious in the field of authorship are the efforts to determine whether Sir Frances Bacon wrote works attributed to Shakespeare (Durning-Lawrence, 1910), or who wrote the twelve disputed Federalist Papers, Hamilton or Madison (Kjell et al., 1994). Recently, similar techniques have been used in the relatively new field of *software forensics* to determine program authorship (Spafford and Weeber, 1993). Gray et al. (1997) wrote a position paper on the subject of authorship, whereas Krsul (1994) conducted an empirical study by gathering code from programmers of varying skill, extracting software metrics, and determining authorship using discriminant analysis. There are also relevant results published in the literature pertaining to the plagiarism of programs (Aiken, 1994; Jankowitz, 1988), which we will not survey here.

Krsul (1994) collected 88 programs written in the C programming language from 29 programmers at the undergraduate, graduate, and faculty levels. He then extracted 18 layout metrics (e.g., indentation of closing curly brackets), 15 style metrics (e.g., mean line length), and 19 structure metrics (e.g., percentage of `int` function definitions). On average, Krsul determined correct authorship 73% of the time. Interestingly, of the 17 most experienced programmers, he was able to determine authorship 100% of the time. The least experienced programmers were the most difficult to classify, presumably because they had not settled into a consistent style. Indeed, they "were surprised to find that one [programmer] had varied his programming style considerably from program to program in a period of only two months" (Krsul and Spafford, 1995, §5.1).

While interesting, it is unclear how much confidence we should have in these results. Krsul (1994) used 52 features and only one or two examples for each of the 20 classes (i.e., the authors). This seems underconstrained, especially when rules of thumb suggest that one needs ten times more examples than features (Jain et al., 2000). On the other hand, it may also suggest that one simply needs to be clever about what constitutes an example. For instance, one could presumably use functions as examples rather than programs, but for the task of determining authorship of malicious

programs, it is unclear whether such data would be possible to collect or if it even exists. Fortunately, as we discuss in the next section, a lack of data was not a problem for our project.

3. Data Collection

As stated previously, the data for our study consisted of 1,971 benign executables and 1,651 malicious executables. All were in the Windows PE format. We obtained benign executables from all folders of machines running the Windows 2000 and XP operating systems. We gathered additional applications from SourceForge (`http://sourceforge.net`) and download.com (`http://www.download.com`).

We obtained virus loaders, worms, and Trojan horses from the Web site VX Heavens (`http://vx.netlux.org`) and from computer-forensic experts at the MITRE Corporation, the sponsors of this project. Some executables were obfuscated with compression, encryption, or both; some were not, but we were not informed which were and which were not. For one small collection, a commercial product for detecting viruses failed to identify 18 of the 114 malicious executables. For the entire collection of 1,651 malicious executables, a commercial program failed to identify 50 as malicious, even though all were known and in the public domain. Note that, for viruses, we examined only the loader programs; we did not include infected executables in our study.

As stated previously, we gathered this collection early in the summer of 2003. Recently, we obtained 291 additional malicious executables from VX Heavens that have appeared after we took our collection. As such, they were not part of our original collection and were not part of our previous study (Kolter and Maloof, 2004). These additional executables were for a real-world, online evaluation, which we motivate and discuss further in Section 7.

We used the `hexdump` utility (Miller, 1999) to convert each executable to hexadecimal codes in an ASCII format. We then produced *n*-grams, by combining each four-byte sequence into a single term. For instance, for the byte sequence `ff 00 ab 3e 12 b3`, the corresponding *n*-grams would be `ff00ab3e`, `00ab3e12`, and `ab3e12b3`. This processing resulted in 255,904,403 distinct *n*-grams. One could also compute *n*-grams from words, something we explored and discuss further in Section 5.2. Using the *n*-grams from all of the executables, we applied techniques from text classification, which we discuss further in the next section.

4. Classification Methodology

Our overall approach drew techniques from machine learning (e.g., Mitchell, 1997), data mining (e.g., Hand et al., 2001), and, in particular, text classification (e.g., Dumais et al., 1998; Sahami et al., 1998). We used the *n*-grams extracted from the executables to form training examples by viewing each *n*-gram as a Boolean attribute that is either present in (i.e., `T`) or absent from (i.e., `F`) the executable. We selected the most relevant attributes (i.e., *n*-grams) by computing the *information gain* (*IG*) for each:

$$IG(j) = \sum_{v_j \in \{0,1\}} \sum_{C_i} P(v_j, C_i) \log \frac{P(v_j, C_i)}{P(v_j)P(C_i)} ,$$

where C_i is the ith class, v_j is the value of the jth attribute, $P(v_j, C_i)$ is the proportion that the jth attribute has the value v_j in the class C_i, $P(v_j)$ is the proportion that the jth *n*-gram takes the value v_j in the training data, and $P(C_i)$ is the proportion of the training data belonging to the class C_i. This measure is also called *average mutual information* (Yang and Pederson, 1997).

We then selected the top 500 *n*-grams, a quantity we determined through pilot studies (see Section 5.2), and applied several learning methods, all of which are implemented in the Wakaito Environment for Knowledge Acquisition (WEKA) (Witten and Frank, 2005): IB*k*, naive Bayes, a support vector machine (SVM), and a decision tree. We also "boosted" the last three of these learners, and we discuss each of these methods in the following sections. Since the task is to detect malicious executables, in subsequent discussion, we refer to the malicious class as the positive class and refer to the benign class as the negative class.

4.1 Instance-based Learner

One of the simplest learning methods is the instance-based (IB) learner (Aha et al., 1991). Its concept description is a collection of training examples or instances. Learning, therefore, is the addition of new examples to the collection. To classify an unknown instance, the performance element finds the example in the collection most similar to the unknown and returns the example's class label as its prediction for the unknown. For Boolean attributes, such as ours, a convenient measure of similarity is the number of values two instances have in common. Variants of this method, such as IB*k*, find the *k* most similar instances and return the majority vote of their class labels as the prediction. Values for *k* are typically odd to prevent ties. Such methods are also known as *nearest neighbor* and k-*nearest neighbors*.

One can estimate a probability distribution from the nearest neighbors and their distances. For ROC analysis, we used the probability of the negative class as a *case rating*, which indicates the degree to which an example is negative. Such ratings paired with the true labels of the test cases are sufficient for estimating an ROC curve (Swets and Pickett, 1982), a matter we discuss further in Section 5.1.

4.2 Naive Bayes

Naive Bayes is a probabilistic method that has a long history in information retrieval and text classification (Maron and Kuhns, 1960). It stores as its concept description the prior probability of each class, $P(C_i)$, and the conditional probability of each attribute value given the class, $P(v_j|C_i)$. It estimates these quantities by counting in training data the frequency of occurrence of the classes and of the attribute values for each class. Then, assuming conditional independence of the attributes, it uses Bayes' rule to compute the posterior probability of each class given an unknown instance, returning as its prediction the class with the highest such value:

$$C = \operatorname*{argmax}_{C_i} P(C_i) \prod_j P(v_j|C_i) \ .$$

For ROC analysis, we used the posterior probability of the negative class as the case rating.

4.3 Support Vector Machines

Support vector machines, or SVMs (Boser et al., 1992), have performed well on traditional text classification tasks (Dumais et al., 1998; Joachims, 1998; Sahami et al., 1998), and performed well on ours. The method produces a linear classifier, so its concept description is a vector of weights, $\vec{w}$, and an intercept or a threshold, b. However, unlike other linear classifiers, such as Fisher's (1936), SVMs use a kernel function to map training data into a higher-dimensioned space so that the problem is linearly separable. It then uses quadratic programming to set $\vec{w}$ and b such that the hyperplane's

margin is optimal, meaning that the distance is maximal from the hyperplane to the closest examples of the positive and negative classes. During performance, the method predicts the positive class if $\langle \vec{w} \cdot \vec{x} \rangle - b > 0$ and predicts the negative class otherwise. Quadratic programming can be expensive for large problems, but sequential minimal optimization (SMO) is a fast, efficient algorithm for training SVMs (Platt, 1998) and is the one implemented in WEKA (Witten and Frank, 2005). During performance, this implementation computes the probability of each class (Platt, 2000), and for ROC analysis, we used probability of the negative class as the rating.

4.4 Decision Trees

A decision tree is a rooted tree with internal nodes corresponding to attributes and leaf nodes corresponding to class labels. For symbolic attributes, branches leading to children correspond to the attribute's values. The performance element uses the attributes and their values of an instance to traverse the tree from the root to a leaf. It predicts the class label of the leaf node. The learning element builds such a tree by selecting the attribute that best splits the training examples into their proper classes. It creates a node, branches, and children for the attribute and its values, removes the attribute from further consideration, and distributes the examples to the appropriate child node. This process repeats recursively until a node contains examples of the same class, at which point, it stores the class label. Most implementations use the *gain ratio* for attribute selection (Quinlan, 1993), a measure based on the information gain. In an effort to reduce overtraining, most implementations also prune induced decision trees by removing subtrees that are likely to perform poorly on test data. WEKA's J48 (Witten and Frank, 2005) is an implementation of the ubiquitous C4.5 (Quinlan, 1993). During performance, J48 assigns weights to each class, and we used the weight of the negative class as the case rating.

4.5 Boosted Classifiers

Boosting (Freund and Schapire, 1996) is a method for combining multiple classifiers. Researchers have shown that *ensemble methods* often improve performance over single classifiers (Dietterich, 2000; Opitz and Maclin, 1999). Boosting produces a set of weighted models by iteratively learning a model from a weighted data set, evaluating it, and reweighting the data set based on the model's performance. During performance, the method uses the set of models and their weights to predict the class with the highest weight. We used the AdaBoost.M1 algorithm (Freund and Schapire, 1996) implemented in WEKA (Witten and Frank, 2005) to boost SVMs, J48, and naive Bayes. As the case rating, we used the weight of the negative class. Note that we did not apply AdaBoost.M1 to IBk because of the high computational expense.

5. Detecting Malicious Executables

With our methodology defined, our first task was to examine how well the learning methods detected malicious executables. We did so by conducting three experimental studies using a standard experimental design. The first was a pilot study to determine the size of words and n-grams, and the number of n-grams relevant for prediction. With those values determined, the second experiment consisted of applying all of the classification methods to a small collection of executables. The third then involved applying the methodology to a larger collection of executables, mainly to investigate how the approach scales.

5.1 Experimental Design

To evaluate the approach and methods, we used stratified ten-fold cross-validation. That is, we randomly partitioned the executables into ten disjoint sets of equal size, selected one as a testing set, and combined the remaining nine to form a training set. We conducted ten such runs using each partition as the testing set.

For each run, we extracted *n*-grams from the executables in the training and testing sets. We selected the most relevant features from the training data, applied each classification method, and used the resulting classifier to rate the examples in the test set.

To conduct ROC analysis (Swets and Pickett, 1982), for each method, we pooled the ratings from the iterations of cross-validation, and used `labroc4` (Metz et al., 2003) to produce an empirical ROC curve and to compute its area and the standard error of the area. With the standard error, we computed 95% confidence intervals (Swets and Pickett, 1982).

5.2 Pilot Studies

We conducted pilot studies to determine three parameters: the size of *n*-grams, the size of words, and the number of selected features. Unfortunately, due to computational requirements, we were unable to evaluate exhaustively all methods for all settings of these parameters, so we assumed that the number of features would most affect performance, and began our investigation accordingly.

Using the experimental methodology described previously, we extracted bytes from 476 malicious executables and 561 benign executables and produced *n*-grams, for $n = 4$. (This smaller set of executables constituted our initial collection, which we later supplemented.) Using information gain, we then selected the best 10, 20, ..., 100, 200, ..., 1,000, 2,000, ..., 10,000 *n*-grams, and evaluated the performance of naive Bayes, SVMs, boosted SVMs, J48, and boosted J48. Selecting 500 *n*-grams produced the best results.

We fixed the number of *n*-grams at 500, and varied *n*, the size of the *n*-grams. We evaluated the same methods for $n = 1, 2, \ldots, 10$, and $n = 4$ produced the best results. We also varied the size of the words (one byte, two bytes, etc.), and results suggested that single bytes produced better results than did multiple bytes.

And so by selecting the top 500 *n*-grams of size four produced from single bytes, we evaluated all of the classification methods on this small collection of executables. We describe the results of this experiment in the next section.

5.3 Experiment with a Small Collection

Processing the small collection of executables produced 68,744,909 distinct *n*-grams. Following our experimental methodology, we used stratified ten-fold cross-validation, selected the 500 best *n*-grams, and applied all of the classification methods. The ROC curves for these methods are in Figure 1, while the areas under these curves (AUC) with 95% confidence intervals are in Table 2.

As one can see, the boosted methods performed well, as did the instance-based learner and the support vector machine. Naive Bayes did not perform as well, and we discuss this further in Section 8.

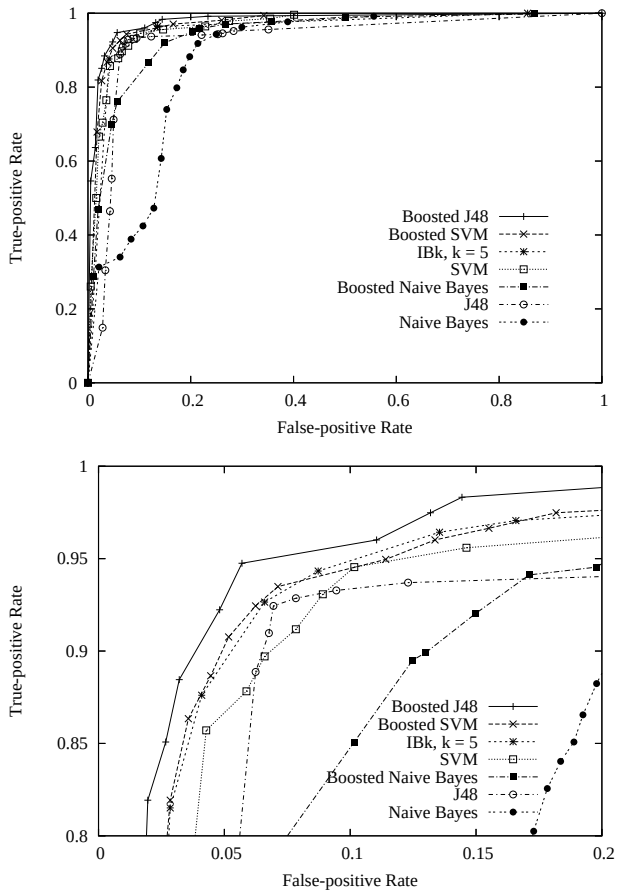


Figure 1: ROC curves for detecting malicious executables in the small collection. Top: The entire ROC graph. Bottom: A magnification.

Method	AUC
Boosted J48	0.9836±0.0095
Boosted SVM	0.9744±0.0118
IBk, $k = 5$	0.9695±0.0129
SVM	0.9671±0.0133
Boosted Naive Bayes	0.9461±0.0170
J48	0.9235±0.0204
Naive Bayes	0.8850±0.0247

Table 2: Results for detecting malicious executables in the small collection. Measures are area under the ROC curve (AUC) with a 95% confidence interval.

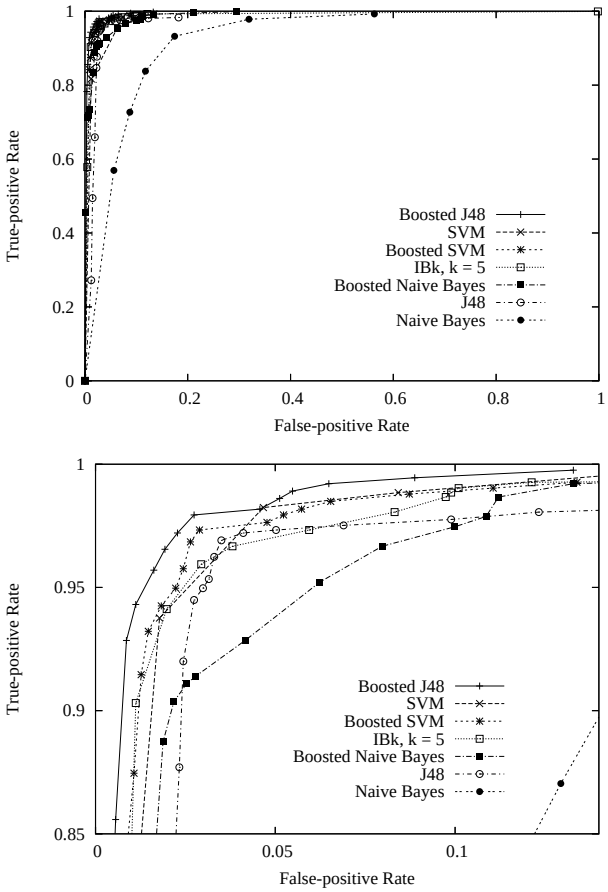


Figure 2: ROC curves for detecting malicious executables in the larger collection. Top: The entire ROC graph. Bottom: A magnification.

Method	AUC
Boosted J48	0.9958±0.0024
SVM	0.9925±0.0033
Boosted SVM	0.9903±0.0038
IBk, $k = 5$	0.9899±0.0038
Boosted Naive Bayes	0.9887±0.0042
J48	0.9712±0.0067
Naive Bayes	0.9366±0.0099

Table 3: Results for detecting malicious executables in the larger collection. Measures are area under the ROC curve (AUC) with a 95% confidence interval.

5.4 Experiment with a Larger Collection

With success on a small collection, we turned our attention to evaluating the methodology on a larger collection of executables. As mentioned previously, this collection consisted of 1,971 benign executables and 1,651 malicious executables, while processing resulted in over 255 million distinct *n*-grams of size four. We followed the same experimental methodology—selecting the 500 top *n*-grams for each run of stratified ten-fold cross-validation, applying the classification methods, and plotting ROC curves.

Figure 2 shows the ROC curves for the various methods, while Table 3 presents the areas under these curves with 95% confidence intervals. As one can see, boosted J48 outperformed all other methods. Other methods, such as IB*k* and boosted SVMs, performed comparably, but the ROC curve for boosted J48 dominated all others.

6. Classifying Executables by Payload Function

We next attempted to classify malicious executables based on the function of their payload. That is, rather than detect malicious executables, we investigated the extent to which classification methods could determine whether a given malicious executable opened a backdoor, mass-mailed, or was an executable virus. We see this aspect of our work most useful for experts in computer forensics. A tool performing this task reliably could reduce the amount of effort to characterize previously undiscovered malicious executables.

Our first challenge was to identify and enumerate the functions of payloads of malicious executables. For this, we consulted VX Heavens and Symantec's Web site. Obviously, the information on these Web sites was not designed to support data-mining experiments, so we had to translate text descriptions into a more structured representation.

However, a greater problem was that we could not find information for all of the malicious executables in our collection. Indeed, we found information for only 525 of the 1,651 malicious executables. As a result, for most categories, we had too few executables to build classifiers and conduct experiments.

A second challenge was that many executables fell into multiple categories. That is, many were so-called *multi-class examples*, a problem common in bioinformatics and document classification. For instance, a malicious executable might open a backdoor and log keystrokes, so it would be in both the backdoor and keylogger classes.

One approach is to create compound classes, such as backdoor+keylogger, in addition to the simple classes (e.g., backdoor and keylogger). One immediate problem was that we had too few examples to support this approach. We had a number of backdoors, a number of keyloggers, but had few executables that were both backdoors and keyloggers.

As a result, we chose to use *one-versus-all classification*. This involves grouping all of, say, the executables with backdoor capabilities into the backdoor class, regardless of their other capabilities (e.g., key logging), and placing all other executables into a non-backdoor class. One then builds a detector for the backdoor class, and does the same for all other classes.

To make a decision, one applies all of the detectors and reports the predictions of the individual classifiers as the overall prediction of the executable. For example, if the detectors for backdoor and for keylogger report hits, then the overall prediction for the executable is backdoor+keylogger.

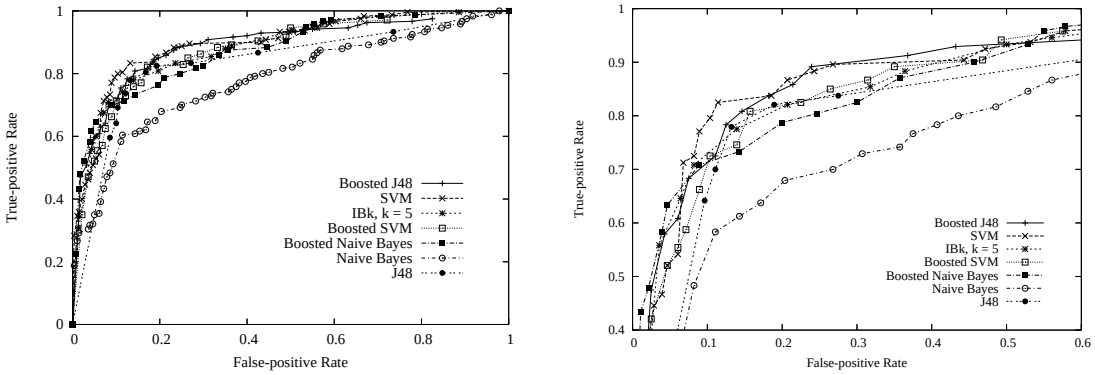


Figure 3: ROC curves for detecting malicious executables with mass-mailing capabilities. Left: The entire ROC graph. Right: A magnification.

Method	Payload		
	Mass Mailer	Backdoor	Virus
Boosted J48	0.8888±0.0152	0.8704±0.0161	0.9114±0.0166
SVM	0.8986±0.0145	0.8508±0.0171	0.8999±0.0175
IBk, $k = 5$	0.8829±0.0155	0.8434±0.0174	0.8975±0.0177
Boosted SVM	0.8758±0.0160	0.8625±0.0165	0.8775±0.0192
Boosted Naive Bayes	0.8773±0.0159	0.8313±0.0180	0.8370±0.0216
J48	0.8315±0.0184	0.7612±0.0205	0.8295±0.0220
Naive Bayes	0.7820±0.0205	0.8190±0.0185	0.7574±0.0250

Table 4: Results for detecting payload function. Measures are area under the ROC curve (AUC) with a 95% confidence interval.

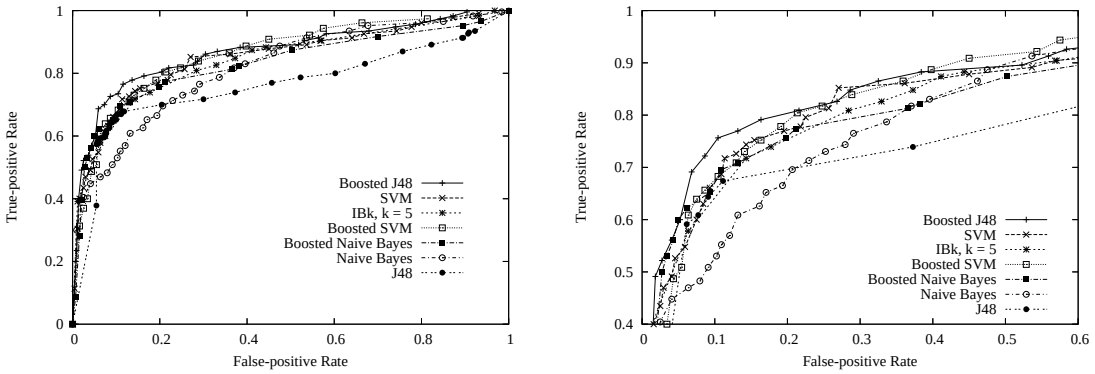


Figure 4: ROC curves for detecting malicious executables with backdoor capabilities. Left: The entire ROC graph. Right: A magnification.

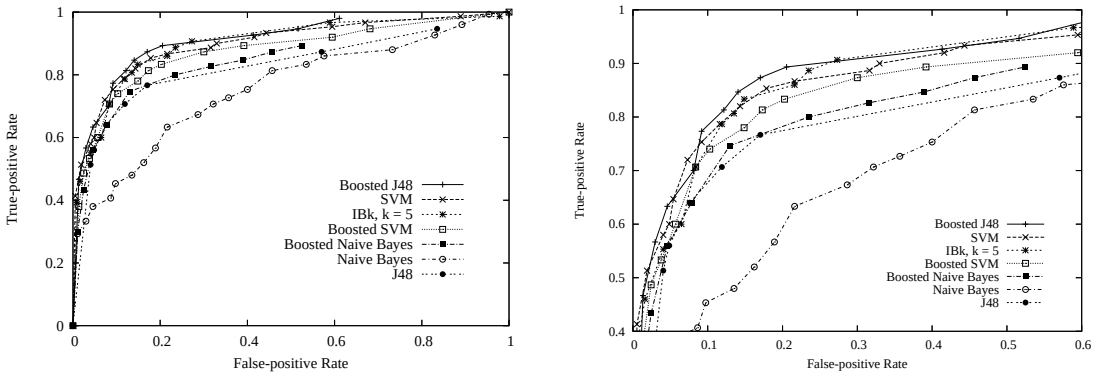


Figure 5: ROC curves for detecting executable viruses. Left: The entire ROC graph. Right: A magnification.

6.1 Experimental Design

We followed an experimental design similar to that described previously, but for each of the functional categories, we created a data set using only malicious executables. We divided it into two subsets, one containing executables that performed the function and one containing those that did not. We then proceeded as before, using stratified ten-fold cross-validation, applying and evaluating the methods, and constructing ROC curves.

6.2 Experimental Results

We present results for three functional categories: mass-mailer, backdoor, and executable virus. Figures 3–5 present the ROC curves for seven methods on the task of detecting executables that mass-mail, open a backdoor, or contain an executable virus. The areas under these ROC curves appear in Table 4. Overall, the results are not as good as those in the experiment that involved detecting malicious executables in our full collection of malicious executables, and we discuss possible causes in Section 8.

The relative performance of the methods on this task was roughly the same as in previous experiments. Naive Bayes generally did not perform as well as the other methods, and we discuss the reasons for this in Section 8. Boosted J48 and the SVM were again the best performing methods, although on this task, the SVM performed slightly better than on previous tasks.

7. Evaluating Real-world, Online Performance

Finally, to estimate what performance might be in an operational environment, we applied the methods to 291 malicious executables discovered after we gathered our original collection. In the previous sections, we generally followed a common experimental design in machine learning and data mining: We randomly partitioned our data into training and testing sets, applied algorithms to the training set, and evaluated the resulting detectors on the testing set. However, one problem with this design for this application is that learning methods were trained on recent malicious executables and tested on older ones. Crucially, this design does not reflect the manner in which a system based

Method	Desired False-positive Rate					
	0.01		0.05		0.1	
	P	A	P	A	P	A
Boosted J48	0.94	0.86	0.99	0.98	1.00	1.00
SVM	0.82	0.41	0.98	0.90	0.99	0.93
Boosted SVM	0.86	0.56	0.98	0.89	0.99	0.92
IBk, $k = 5$	0.90	0.67	0.99	0.81	1.00	0.99
Boosted Naive Bayes	0.79	0.55	0.94	0.93	0.98	0.98
J48	0.20	0.34	0.97	0.94	0.98	0.95
Naive Bayes	0.48	0.28	0.57	0.72	0.81	0.83

Table 5: Results of a real-world, online evaluation. Predicted (P) versus actual (A) detect rates for three desired false-positive rates on 291 new, previously unseen malicious executables. Predicted detect rates are from Figure 2 and the experiment described in Section 5.4.

on our methodology would be used. In this section, we rectify this and describe an experiment designed to better evaluate the real-world, online performance of the detectors.

As mentioned previously, we gathered our collection of executables in the summer of 2003. In August of 2004, we retrieved from VX Heavens all of the new malicious executables and selected those that were discovered after we gathered our original collection. This required retrieving the 3,082 new executables that were in the PE format and using commercial software to verify independently that each executable was indeed malicious. We then cross-referenced the names of the verified malicious executables with information on various Web sites to produce the subset of malicious executables discovered between July of 2003 and August of 2004. There were 291 such executables.

7.1 Experimental Design

To conduct this experiment, we built classifiers from all of the executables in our original collection, both malicious and benign. We then selected three desired false-positive rates, 0.01, 0.05, and 0.1. This, in turn, let us select three decision thresholds from each ROC curve for each method. Using these thresholds to parameterize specific classifiers, we applied them to each of the 291 new malicious executables in the order of their date of discovery.

7.2 Experimental Results

Rather than analyze all of these results, we will discuss the actual (A) detection rates for the desired false-positive rate of 0.05.[1] As one can see, boosted decision trees detected about 98% of the new malicious executables, missing 6 of 291 malicious executables. For some applications, six may be too many, but if one is willing to tolerate a false-positive rate of 0.1, then one can achieve a perfect detect rate, at least on these 291 malicious executables.

1. Our reasoning was that, for most operational scenarios, a desired false-positive rate of 0.1 would be too high, and the detect rates achieved for a desired false-positive rate of 0.01 were too low. Knowledge of a given operational environment would presumably help us choose a more appropriate decision threshold.

However, it is also important to compare the actual detection rates to the predicted rates from the experiment using our larger collection of executables, discussed in Section 5.4. As one can see in Table 5 by comparing the predicted (P) and actual (A) detection rates for a desired false-positive rate of 0.05, four methods (SVM, boosted SVM, IB*k*, and J48) performed worse on the new malicious executables, two methods (boosted J48 and boosted naive Bayes) performed about as well under both conditions, and one method (naive Bayes) performed much better. Nonetheless, we determined that boosted decision trees achieved the best performance overall, not only in terms of the best actual performance on the new malicious executables, but also in terms of matching the predicted performance from the experiment involving the larger collection of executables.

8. Discussion

To date, our results suggest that methods of machine learning, data mining, and text classification are appropriate and useful for detecting malicious executables in the wild. Boosted classifiers, IB*k*, and a support vector machine performed exceptionally well given our current data collection. That the boosted classifiers generally outperformed single classifiers echos the conclusion of several empirical studies of boosting (Bauer and Kohavi, 1999; Breiman, 1998; Dietterich, 2000; Freund and Schapire, 1996), which suggest that boosting improves the performance of unstable classifiers, such as J48, by reducing their bias and variance (Bauer and Kohavi, 1999; Breiman, 1998). Boosting can adversely affect stable classifiers (Bauer and Kohavi, 1999), such as naive Bayes, although in our study, boosting naive Bayes improved performance. Stability may also explain why the benefit of boosting SVMs was inconclusive in our study (Breiman, 1998).

Our experimental results suggest that the methodology will scale to larger collections of executables. The larger collection in our study contained more than three times the number of executables in the smaller collection. Yet, as one can see in Tables 2 and 3, the absolute performance of all of the methods was better for the larger collection than for the smaller one. The relative performance of the methods changed somewhat. For example, the SVM moved from fourth to second, displacing the boosted SVMs and IB*k*.

Regarding our results for classifying executables by function, we suspect the methods did not perform as well as they did on the detection task for two reasons. First, with the classification task, the algorithms must make finer distinctions between malicious and benign executables. For example, a malicious executable that mass-mails will be similar in some respects to a legitimate e-mail client. Such similarity could account for the lower performance.

Indeed, in pilot studies, we attempted to use the methods to distinguish between benign and malicious executables that edited the registry. Performance on this task was lower than on the others, and we suspect this is because editing the registry is a function common to many executables, malicious and benign. Such similarity could have accounted for the lower performance.

Second, we suspect that, on the classification task, performance suffered because the algorithms built classifiers from fewer examples. Performance on the detection task improved when we added additional examples, and we suspect that, likewise, with additional examples, we will obtain similar improvements in accuracy on the classification task.

Regarding our online evaluation of the methods, we believe the experimental design represents how such methods would be used in a commercial or operational system. We did not conduct this experiment from the outset (Kolter and Maloof, 2004) because it was impossible to determine the date of discovery of all of the malicious executables in our collection. Moreover, to conduct the ideal

```
030b0105 = T
|   0b010219 = T: malicious (2.0)
|   0b010219 = F
|   |   0000000a = T
|   |   |   0001ff25 = T
|   |   |   |   0c100001 = T
|   |   |   |   |   0000c700 = T: benign (6.0)
|   |   |   |   |   0000c700 = F: malicious (2.0)
|   |   |   |   0c100001 = F: malicious (2.0)
|   |   |   0001ff25 = F: benign (10.0)
|   |   0000000a = F: benign (253.0)
030b0105 = F
...
```

Figure 6: Portion of a decision tree built from benign and malicious executables.

experiment, we would also need to collect different versions of benign executables and when they were released. It was easier to take one "snapshot" of existing malicious and benign executables, conduct traditional experiments, and then, at a later date, retrieve any new malicious executables for the online experiment.

During the processing of the 291 new malicious executables, we did not update the classifiers when there was a mistake or a so-called "near miss". Clearly, in an operational setting, if the system were to make a mistake or to detect a malicious executable with low certainty, then, ideally, one would add it to the collection and reprocess everything. (One would also have to do the same for benign executables.) However, because of the computational overhead, we did not do this, and as a consequence, our results are pessimistic. Presumably, all of the methods would perform better with the benefit of additional training data. Nonetheless, for some methods, the results are quite promising, as shown in Table 5.

Visual inspection of the concept descriptions yielded interesting insights, but further work is required before these descriptions will be directly useful for computer-forensic experts. Figure 6 shows a portion of one decision tree built from benign and malicious executables.

As an example, the first branch of the tree indicates that if an executable contains the *n*-grams `030b0105` and `0b010219`, then it is malicious. After an analysis of our collection of malicious executables, we discovered that both *n*-grams were from the PE header, implying that a single file contained two such headers. More investigation revealed that two executables in our collection contained another executable, which explains the presence of two PE headers in a single file. This was an interesting find, but it represented an insignificantly small portion of the malicious programs.

Leaf nodes covering many executables were often at the end of long branches where one set of *n*-grams (i.e., byte codes) had to be present and another set had to be absent. Understanding why the absence of byte codes was important for an executable being malicious proved to be a difficult and often impossible task.

It was fairly easy to establish that some *n*-grams in the decision tree were from string sequences and that some were from code sequences, but some were incomprehensible. For example, the *n*-gram `0000000a` appeared in 75% of the malicious executables, but it was not part of the executable format, it was not a string sequence, and it was not a code sequence. We have yet to determine its purpose.

Nonetheless, for the large collection of executables, the size of the decision trees averaged over 10 runs was about 90 nodes. No tree exceeded 103 nodes. The heights of the trees never exceeded 13 nodes, and subtrees of heights of 9 or less covered roughly 99.3% of the training examples. While these trees did not support a thorough forensic analysis, they did compactly encode a large number of benign and malicious executables.

Unfortunately, the best performing method did not always produce the most readable concept descriptions. Of the methods we considered, J48 is mostly likely to produce descriptions useful for computer-forensic experts. However, J48 was not the best performing method in any of our experiments. The best performing method was boosted J48. While it is true that this method also produces decision trees, it actually produces a set of weighted trees. We discussed the difficulties of analyzing a single tree, so it is unclear if analyzing an ensemble of weighted trees will be helpful for experts. And since J48 was not the best performing method, we may also have to question the utility of analyzing a single decision tree when its performance is subpar.

We estimated that about 20–25% of the malicious executables in our collection were obfuscated with either compression or encryption. To the best of our knowledge, none of the benign executables were obfuscated. Early in our investigation, we conjectured that obfuscation would likely interfere with classifying payload function, but that it would not do so with detecting whether the executable is malicious. Our results on these tasks seem to support our conjecture: We were able to detect malicious executables with high accuracy, so it is unlikely that obfuscation affected performance. On the other hand, we were not able to achieve the same high accuracy when classifying payload function, and the presence of obfuscation may have contributed to this result.

With the detection task, it is possible that the methods simply learned to detect certain forms of obfuscation, such as run-time compression, but this does not seem problematic as long as those forms are correlated with malicious executables. Based on our collection and our own investigation, this is presently the case.

To place our results in context with the study of Schultz et al. (2001), they reported that the best performing approaches were naive Bayes trained on the printable strings from the program and an ensemble of naive-Bayesian classifiers trained on byte sequences. They did not report areas under their ROC curves, but visual inspection of these curves suggests that with the exception of naive Bayes, all of our methods outperformed their ensemble of naive-Bayesian classifiers. It also appears that our best performing methods, such as boosted J48, outperformed their naive Bayesian classifier trained with strings.

These differences in performance could be due to several factors. We analyzed different types of executables: Their collection consisted mostly of viruses, whereas ours contained viruses loaders, worms, and Trojan horses. Ours consisted of executables in the Windows PE format; about 5.6% of theirs was in this format.

Our better results could be due to how we processed byte sequences. Schultz et al. (2001) used non-overlapping two-byte sequences, whereas we used overlapping sequences of four bytes. With their approach it is possible that a useful feature (i.e., a predictive sequence of bytes) would be split across a boundary. This could explain why in their study string features appeared to be better than byte sequences, since extracted strings would not be broken apart. Their approach produced much less training data than did ours, but our application of feature selection reduced the original set of more than 255 million *n*-grams to a manageable 500.

Our results for naive Bayes were poor in comparison to theirs. We again attribute this to the differences in data extraction methods. Naive Bayes is well known to be sensitive to conditionally

dependent attributes (Domingos and Pazzani, 1997). We used overlapping byte sequences as attributes, so there were many that were conditionally dependent. Indeed, after analyzing decision trees produced by J48, we found evidence that overlapping sequences were important for detection. Specifically, some subpaths of these decision trees consisted of sequentially overlapping terms that together formed byte sequences relevant for prediction. Schultz et al.'s (2001) extraction methods would not have produced conditionally dependent attributes to the same degree, if at all, since they used strings and non-overlapping byte sequences.

Regarding our experimental design, we decided to pool a method's ratings and produce a single ROC curve (see Section 5.1) because `labroc4` (Metz et al., 2003) occasionally could not fit an ROC curve to a method's ratings from a single fold of cross-validation (i.e., the ratings were degenerate). We also considered producing ROC convex hulls (Provost and Fawcett, 2001) and cost curves (Drummond and Holte, 2000), but determined that traditional ROC analysis was appropriate for our results (e.g., the curve for boosted J48 dominated all other curves).

In our study, there was an issue of high computational overhead. Selecting features was expensive, and we had to resort to a disk-based implementation for computing information gain, which required a great deal of time and space to execute. However, once selected, WEKA's (Witten and Frank, 2005) Java implementations executed quickly on the training examples with their 500 Boolean attributes.

The greatest impediment to our investigation was the absence of detailed, structured information about the malicious executables in our collection. As mentioned previously, we had $1,651$ malicious executables, but found information for only 525 that was sufficient to support our experiment on function classification, described in Section 6.

We arduously gathered this information by reading it from Web pages. We contemplated implementing software to extract this information, but abandoned this idea because of the difficulty of processing such semi-structured information and because of that information's ad hoc nature. As an example, for one executable, the description that it opened a backdoor appeared in a section describing the executable's payload, whereas the same information for another executable appeared in a section describing how the malicious executable degrades performance. This suggests the need for a well-engineered database for storing information about malicious software.

As another example, we found incomplete information about the dates of discovery of many of the malicious executables. With this information, we could have evaluated our methods on the malicious executables in the order they were discovered. This would have been similar to the evaluation we conducted using the 291 previously unseen malicious executables, as described in Section 7, but having complete information for all of the executables would have resulted in a much stronger evaluation.

However, to conduct a proper evaluation, we would also needed a comparable collection of benign executables. It seems unlikely that we would be able to reconstruct realistic snapshots of complete Windows systems over a sufficient period of time. Snapshots of the system software might be possible, but creating a historical archive of application software and their different versions seems all but impossible. Such snapshots would be required for a commercial system, and creating such snapshots would be easier going forward.

In terms of our approach, it is important to note that we have investigated other methods of data extraction. For instance, we examined whether printable strings from the executable might be useful, but reasoned that subsets of *n*-grams would capture the same information. Indeed, after inspecting some of the decision trees that J48 produced, we found evidence suggesting that *n*-grams

formed from strings were being used for detection. Nonetheless, if we later determine that explicitly representing printable strings is important, we can easily extend our representation to encode their presence or absence. On the other hand, as we stated previously, one must question the use of printable strings or DLL information since compression and other forms of obfuscation can mask this information.

We also considered using disassembled code as training data. For malicious executables using compression, being able to obtain a disassembly of critical sections of code may be a questionable assumption. Moreover, in pilot studies, a commercial product failed to disassemble some of our malicious executables.

We considered an approach that runs malicious executables in a "sandbox" and produces an audit of the machine instructions. Naturally, we would not be able to execute completely the program, but $10,000$ instructions may be sufficient to differentiate benign and malicious behavior. We have not pursued this idea because of a lack of auditing tools, the difficulty of handling large numbers of interactive programs, and the inability of detecting malicious behavior occurring near the end of sufficiently long programs. Moreover, some malicious programs can detect when they are being executed by a virtual machine and either terminate execution or avoid executing malicious sections of code.

There are at least two immediate commercial applications of our work. The first is a system, similar to MECS, for detecting malicious executables. Server software would need to store all known malicious executables and a comparably large set of benign executables. Due to the computational overhead of producing classifiers from such data, algorithms for computing information gain and for evaluating classification methods would have to be executed incrementally, in parallel, or both.

Client software would need to extract only the top *n*-grams from a given executable, apply a classifier, and predict. Updates to the classifier could be made remotely over the Internet. Since the best performing method may change with new training data, it will be critical for the server to evaluate a variety of methods and for the client to accommodate any of the potential classifiers. Used in conjunction with standard signature methods, these methods could provide better detection of malicious executables than is currently possible.

The second is a system oriented more toward computer-forensic experts. Even though work remains before decision trees could be used to analyze malicious executables, one could use IB*k* to retrieve known malicious executables similar to a newly discovered malicious executable. Based on the properties of the retrieved executables, such a system could give investigators insights into the new executable's function. While it remains an open issue whether an executable's statistical properties are predictive of its function, we have presented evidence suggesting it may be possible to achieve useful detection rates when predicting function.

9. Concluding Remarks

We considered the application of techniques from machine learning, data mining, and text classification to the problem of detecting and classifying unknown malicious executables in the wild. After evaluating a variety of inductive methods, results suggest that, for the task of detecting malicious executables, boosted J48 produced the best detector with an area under the ROC curve of 0.996.

We also investigated the ability of these methods to classify malicious executables based on their payload's function. For payloads that mass-mail, open a backdoor, and inject viral code, boosted J48 again produced the best detectors with areas under the ROC curve around 0.9. While overall

the performance on this task was not as impressive as that on the detection task, we contend that performance will improve with the removal of obfuscation and with additional training examples.

Finally, boosted J48 also performed well on the task of detecting 291 malicious executables discovered after we gathered our original collection, an evaluation that best reflects how one might use the methodology in an operational environment. Indeed, our methodology resulted in a fielded prototype called MECS, the Malicious Executable Classification System, which we delivered to the MITRE Corporation.

In future work, we hope to remove obfuscation from our malicious executables and rerun the experiment on classifying payload function. Removing obfuscation and producing "clean" executables may prove challenging, but doing so would provide the best opportunity to evaluate whether obfuscation affected the performance of the classifiers.

We also plan to investigate the similarity of malicious executables and how such executables change over time. In this regard, we have not yet attempted to cluster our collection of executables, but doing so may yield two insights. First, if a new, unanalyzed malicious executable is similar to others that have been analyzed, it may help computer-forensic experts conduct a faster analysis of the new threat.

Second, if we add information about when the executables were discovered, we may be able to determine how malicious executables were derived from others. Although there is a weak analog between DNA sequences and byte codes from executables, we may be able to use a collection of malicious executables to build phylogenetic trees that may elucidate "evolutionary relationships" existing among them.

We anticipate that inductive approaches, such as ours, is but one process in an overall strategy for detecting and classifying "malware." When combined with approaches that use cryptographic hashes, search for known signatures, execute and analyze code in a virtual machine, we hope that such a strategy for detecting and classifying malicious executables will improve the security of computers. Indeed, the delivery of MECS to MITRE has provided computer-forensic experts with a valuable tool. We anticipate that continued investigation of inductive methods for detecting and classifying malicious executables will yield additional tools and more secure systems.

Acknowledgments

The authors first and foremost thank William Asmond and Thomas Ervin of the MITRE Corporation for providing their expertise, advice, and collection of malicious executables. We thank Nancy Houdek of Georgetown for help gathering information about the functional characteristics of the malicious executables in our collection. The authors thank the anonymous reviewers for their time and helpful comments, Ophir Frieder of IIT for help with the vector space model, Abdur Chowdhury of IIT for advice on the scalability of the vector space model, Bob Wagner of the FDA for assistance with ROC analysis, Eric Bloedorn of MITRE for general guidance on our approach, and Matthew Krause of Yale University for reviewing an earlier draft of the paper. Finally, we thank Richard Squier of Georgetown for supplying much of the additional computational resources needed for this study through Grant No. DAAD19-00-1-0165 from the U.S. Army Research Office. This research was conducted in the Department of Computer Science at Georgetown University. Our work was supported by the MITRE Corporation under contract 53271. The authors are listed in alphabetical order.

References

D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6: 37–66, 1991.

A. Aiken. MOSS: A system for detecting software plagiarism. Software, Department of Computer Science, University of California, Berkeley, http://www.cs.berkeley.edu/ aiken/moss.html, 1994.

Anonymous. *Maximum Security*. Sams Publishing, Indianapolis, IN, 4th edition, 2003.

B. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1–2):105–139, 1999.

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fourth Workshop on Computational Learning Theory*, pages 144–152, New York, NY, 1992. ACM Press.

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *Proceedings of the Twelfth USENIX Security Symposium*, Berkeley, CA, 2003. Advanced Computing Systems Association.

W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, San Francisco, CA, 1995. Morgan Kaufmann.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

P. Domingos and M. J. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

C. Drummond and R. C. Holte. Explicitly representing expected cost: An alternative to ROC representation. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 198–207, New York, NY, 2000. ACM Press.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pages 148–155, New York, NY, 1998. ACM Press.

E. Durning-Lawrence. *Bacon is Shake-speare*. The John McBride Company, New York, NY, 1910.

R. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, San Francisco, CA, 1996. Morgan Kaufmann.

A. R. Gray, P. J. Sallis, and S. G. MacDonell. Software forensics: Extending authorship analysis techniques to computer programs. In *Proceedings of the Third Biannual Conference of the International Association of Forensic Linguists*, pages 1–8, Birmingham, UK, 1997. International Association of Forensic Linguists. URL `citeseer.nj.nec.com/gray97software.html`.

D. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. MIT Press, Cambridge, MA, 2001.

A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

H. T. Jankowitz. Detecting plagiarism in student Pascal programs. *Computer Journal*, 31(1):1–8, 1988.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the Tenth European Conference on Machine Learning*, pages 487–494, Berlin, 1998. Springer.

J. O. Kephart, G. B. Sorkin, W. C. Arnold, D. M. Chess, G. J. Tesauro, and S. R. White. Biologically inspired defenses against computer viruses. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 985–996, San Francisco, CA, 1995. Morgan Kaufmann.

B. Kjell, W. A. Woods, and O. Frieder. Discrimination of authorship using visualization. *Information Processing and Management*, 30(1):141–150, 1994.

J. Z. Kolter and M. A. Maloof. Learning to detect malicious executables in the wild. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 470–478, New York, NY, 2004. ACM Press.

I. Krsul. Authorship analysis: Identifying the author of a program. Master's thesis, Purdue University, West Lafayette, IN, 1994.

I. Krsul and E. Spafford. Authorship analysis: Identifying the authors of a program. In *Proceedings of the Eighteenth National Information Systems Security Conference*, pages 514–524, Gaithersburg, MD, 1995. National Institute of Standards and Technology.

R. W. Lo, K. N. Levitt, and R. A. Olsson. MCF: A malicious code filter. *Computers & Security*, 14 (6):541–566, 1995. http://www.cs.columbia.edu/ids/mef/llo95.ps.

M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the Association of Computing Machinery*, 7(3):216–244, 1960.

G. McGraw and G. Morisett. Attacking malicious code: A report to the Infosec Research Council. *IEEE Software*, pages 33–41, September/October 2000. http://www.cigital.com/ gem/malcode.pdf.

C. E. Metz, Y. Jiang, H. MacMahon, R. M. Nishikawa, and X. Pan. ROC software. Web page, Kurt Rossmann Laboratories for Radiologic Image Research, University of Chicago, Chicago, IL, 2003. URL `http://www-radiology.uchicago.edu/krl/roc\_soft.htm`.

P. Miller. hexdump 1.4. Software, http://gd.tuwien.ac.at/softeng/Aegis/hexdump.html, 1999.

T. M. Mitchell. *Machine learning*. McGraw-Hill, New York, NY, 1997.

D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999. URL `http://www.jair.org`.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and S. Mika, editors, *Advances in Kernel Methods—Support Vector Learning*. MIT Press, Cambridge, MA, 1998.

J. Platt. Probabilities for SV machines. In P. J. Bartlett, B. Schölkopf, D Schuurmans, and A. J. Smola, editors, *Advances in Large-Margin Classifiers*, pages 61–74. MIT Press, Cambridge, MA, 2000.

F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42: 203–231, 2001.

J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Francisco, CA, 1993.

M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*, Menlo Park, CA, 1998. AAAI Press. Technical Report WS-98-05.

M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 38–49, Los Alamitos, CA, 2001. IEEE Press. URL `http://www.cs.columbia.edu/~ezk/research`.

S. Soman, C. Krintz, and G. Vigna. Detecting malicious Java code using virtual machine auditing. In *Proceedings of the Twelfth USENIX Security Symposium*, Berkeley, CA, 2003. Advanced Computing Systems Association.

E. H. Spafford and S. A. Weeber. Software forensics: Can we track code to its authors? *Computers & Security*, 12:585–595, 1993.

J. A. Swets and R. M. Pickett. *Evaluation of diagnostic systems: Methods from signal detection theory*. Academic Press, New York, NY, 1982.

G. Tesauro, J. O. Kephart, and G. B. Sorkin. Neural networks for computer virus recognition. *IEEE Expert*, 11(4):5–6, August 1996.

I. H. Witten and E. Frank. *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2005. http://www.cs.waikato.ac.nz/ml/weka/index.html.

Y. Yang and J. O. Pederson. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, San Francisco, CA, 1997. Morgan Kaufmann.

Journal of Machine Learning Research 7 (2006) 2745-2769 Submitted 3/06; Revised 9/06; Published 12/06

On Inferring Application Protocol Behaviors in Encrypted Network Traffic

Charles V. Wright CVWRIGHT@JHU.EDU
Fabian Monrose FABIAN@JHU.EDU
Gerald M. Masson MASSON@JHU.EDU
Information Security Institute
Johns Hopkins University
Baltimore, MD 21218, USA

Editor: Philip Chan

Abstract

Several fundamental security mechanisms for restricting access to network resources rely on the ability of a reference monitor to inspect the contents of traffic as it traverses the network. However, with the increasing popularity of cryptographic protocols, the traditional means of inspecting packet contents to enforce security policies is no longer a viable approach as message contents are concealed by encryption. In this paper, we investigate the extent to which common application protocols can be identified using only the features that remain intact after encryption—namely packet size, timing, and direction. We first present what we believe to be the first exploratory look at protocol identification in encrypted tunnels which carry traffic from many TCP connections simultaneously, using only post-encryption observable features. We then explore the problem of protocol identification in individual encrypted TCP connections, using much less data than in other recent approaches. The results of our evaluation show that our classifiers achieve accuracy greater than 90% for several protocols in aggregate traffic, and, for most protocols, greater than 80% when making fine-grained classifications on single connections. Moreover, perhaps most surprisingly, we show that one can even estimate the number of live connections in certain classes of encrypted tunnels to within, on average, better than 20%.

Keywords: traffic classification, hidden Markov models, network security

1. Introduction

To effectively manage large networks, an administrator's ability to characterize the traffic within the network's boundaries is critical for diagnosing problems, provisioning capacity, and detecting attacks or misuses of the network. Unfortunately, for the most part, current approaches for identifying application traffic rely on *inspecting* packets on the wire, which can fail to provide a reliable, or even correct, characterization of the traffic. For one, that information (e.g., port numbers and `TCP` flags) is determined entirely by the end hosts, and thus can be easily changed to disguise or conceal aberrant traffic. In fact, such malicious practices are not uncommon, and often occur after an intruder gains access to the network (e.g., to install a "backdoor") or when legitimate users attempt to violate network policies. For example, many chat and file sharing applications can be easily configured to use the standard port for `HTTP` in order to bypass simple packet-filtering firewalls. Furthermore, recent peer-to-peer file-sharing applications such as BitTorrent (Cohen, 2003) can run entirely on

user-specified ports, and Trojan horse or virus programs may encrypt their communication to deter the development of effective detection signatures.

Even more problematic for such traffic characterization techniques is the fact that with the increased use of cryptographic protocols such as SSL (Rescorla, 2000) and SSH (Ylonen, 1996), fewer and fewer packets in legitimate traffic become available for inspection. While the growing popularity of such protocols has greatly enhanced the security of the user experience on the Internet—by protecting messages from eavesdroppers—one can argue that its use hinders legitimate traffic analysis. Furthermore, we may reasonably expect that the use of encrypted communications will only become more commonplace as Internet users become more security-savvy. Therefore future techniques for identifying application protocols and behaviors may only have access to a severely restricted set of features, namely those that remain intact after encryption.

Clearly, the ability to reliably detect instances of various application protocols "*in the dark*" (Karagiannis et al., 2005) would be of tremendous practical value. For one, armed with this capability, network administrators would be in a much better position to detect violations of network policies by users running instances of forbidden applications over encrypted channels (e.g., using `SSH`'s port-forwarding feature). Unfortunately, most of the existing work on traffic classification either relies on inspecting packet payloads (Zhang and Paxson, 2000a; Moore and Papagiannaki, 2005), TCP headers (Early et al., 2003; Moore and Zuev, 2005; Karagiannis et al., 2005), or can only assign flows to broad classes of protocols such as "bulk data transfer," "p2p," or "interactive" (Moore and Papagiannaki, 2005; Moore and Zuev, 2005; Karagiannis et al., 2005).

Here we investigate the extent to which common Internet application protocols remain distinguishable even when packet payloads and TCP headers have been stripped away, leaving only extremely lean data which includes nothing more than the packets' timing, size, and direction. We begin our analysis in §3 by exploring protocol recognition techniques for traffic aggregates where all flows carry the same application protocol. We then develop tools to enhance the initial analysis provided by these first tools by addressing more specific scenarios. In §4, we relax the single-protocol assumption and address protocol recognition with very lean data on individual TCP connections. These methods might be used to estimate the traffic mix on traces which are believed to contain several distinct protocols, or as a fine-grained way to verify that a set of connections really does contain only a single given application protocol. In §5 we relax the assumption that the individual flows can be demultiplexed from the aggregate and show how, when there is only a single application protocol in use, we can nevertheless still glean meaningful information from the stream of packets and track the number of live connections in the tunnel. We review related work in §6 and discuss future directions in §7.

2. Data

To be useful in practice, traffic analysis approaches of the type we develop in this paper must be effective in dealing with the noisy and skewed data typical of real Internet traffic. We therefore empirically evaluate our techniques using real traffic traces collected by the Statistics Group at George Mason University in 2003 (Faxon et al., 2004). The traces contain headers for IP packets on GMU's Internet (OC-3) link from the first 10 minutes of every quarter hour over a two-month period. The data set contains traffic for a `class B` network which includes several university-wide and departmental servers for mail, web, and other services, as well as hundreds of Internet-connected client machines. From these traces, we extract inbound TCP connections on the well-known ports

for SMTP (25), HTTP (80), HTTP over SSL (443), FTP (20), SSH (22), and Telnet (23), as well as outbound SMTP and AOL Instant Messenger traffic. Since we do not have access to packet payloads in these traces, we do not attempt to determine the "ground truth" of which connections truly belong to which protocols.[1] Instead, we simply use the TCP port numbers as our class labels, and therefore, it is likely some connections have been incorrectly labeled. However, because these mislabeled connections only increase the entropy of the data, the net result will be that we *under*-estimate the accuracy our techniques could achieve if given a perfectly-labeled version of the same traces (Lee and Xiang, 2001).

For each extracted TCP connection, we record the sequence of ⟨size, arrival time⟩ tuples for each packet in the connection, in arrival order. We encode the packet's direction in the sign bit of the packet's size, so that packets sent from server to client have size less than zero and those from client to server have size greater than zero. Since the traces in this data set consist mostly of unencrypted, non-tunneled TCP connections, a few additional preprocessing steps are necessary to simulate the more challenging scenarios which our techniques are designed to address. To simulate the effect of encryption on the traffic in our data set we assume the encryption is performed with a symmetric block cipher such as AES (Federal Information Processing Standards, 2001), and round the observed packet sizes up accordingly. We perform our evaluation using a block size of 64 bytes (512 bits), which is larger than most used in practice, yet still affords a good balance of recognition accuracy and computational efficiency. If analyzing real traffic encrypted with a smaller block size (for example, 128 bits), we can always round the observed packet sizes up.

3. Traffic Classification in Aggregate Encrypted Traffic

Here we investigate the problem of determining the application protocol in use in aggregate traffic composed of several TCP connections which all employ the same application protocol. Unlike previous approaches such as BLINC (Karagiannis et al., 2005), our approach does not rely on any information about the hosts or network involved; instead, we use only the features of the actual packets on the wire which remain observable after encryption, namely: timing, size, and direction.

The techniques we develop here can be used to quickly and efficiently infer the nature of the application protocol used in aggregate traffic without demultiplexing or reassembling the individual flows from the aggregate. Such traffic might correspond to a set of TCP connections to a given host or network, perhaps running on a nonstandard port and identified using techniques like that of Xu et al. (2005) as comprising a dominant or "heavy hitter" behavior in the network. Our techniques could then be used by a network administrator to determine the application layer behavior. Furthermore, these techniques are also applicable to certain classes of encrypted tunnels, namely those which carry traffic for a single application protocol. We address the case of tunneled traffic in greater detail in §5.

To evaluate the techniques developed in this section, we assemble traffic aggregates for each protocol using several TCP connections extracted from the GMU data as described in §2. For each 10-minute trace and each protocol, we select all connections for the given protocol in the given trace, and interleave their packets into a single unified stream, sorted in order of arrival on the link. We then split this stream into several smaller epochs of constant length s and count the number of packets

1. We have checked randomly-selected subsets of flows for each protocol and verified, using visualization techniques (Wright et al., 2006), that the behaviors exhibited therein appear reasonable for the given protocols. Examples of these visualizations are available on the web at http://www.cs.jhu.edu/~cwright/traffic-viz.

of several different types (based on size and direction) that arrive during each epoch. Currently, we group packets into four types; any packet is classified as either small (i.e., 64 bytes or less) or not (i.e., greater than 64 bytes), and as either traveling from client to server or from server to client. In general, when we consider M different packet types, this splitting and counting procedure yields a vector-valued count of packets $\hat{n}_t = \langle n_{t1}, n_{t2}, \ldots, n_{tM} \rangle$ for each epoch t. An aggregate consisting of T s-length epochs is then represented by the sequence of vectors $\hat{n}_1, \hat{n}_2, \ldots, \hat{n}_T$. The epoch length s is typically on the order of several seconds, yielding a sequence length T of about 100 for each 10-minute trace.

3.1 Identifying Application Protocols in Aggregate Traffic

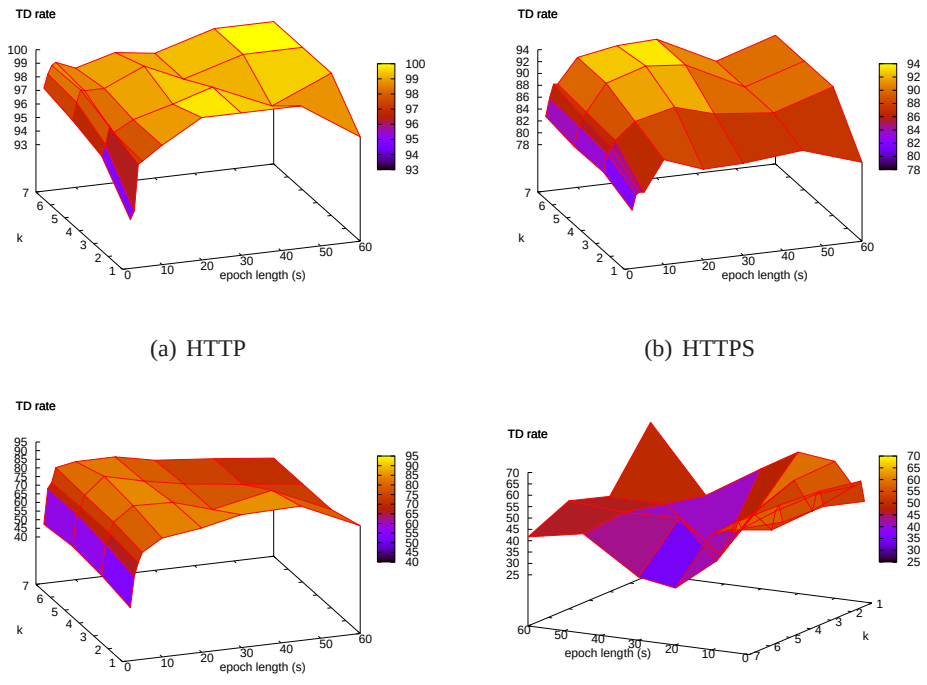
To identify the application protocol used in a single-protocol aggregate, we first construct a k-Nearest Neighbor (k-NN) classifier which assigns protocol labels to the s-length epochs of time based on the number of packets of each type that arrive during the given interval.

To build the k-NN classifier, we select a random day in the GMU data for use as a training set. We then assemble single-protocol aggregates from this day's traces for each protocol in the study, yielding a list of vectors $\hat{n}_1, \hat{n}_2, \ldots$ for each such aggregate. To allow for differences in traffic intensity while preserving the relative frequencies of the different packet types, each resulting vector of counts $\hat{n}_t$ is then normalized so that $\sum_{m=1}^{M} n_{tm} = 1$. Finally, each normalized vector, together with its protocol label, is added to the classifier.

To classify a new epoch u using the k-NN classifier, we the use Kullback-Leibler distance, or *divergence* (Kullback and Leibler, 1951), to determine which k vectors in the training set are "nearest" to the vector $\hat{n}_u$ of counts for the given epoch. The K-L distance is a logical distance metric in this instance because each normalized vector of counts $\hat{n}_i$ essentially represents a discrete probability mass function over the set of packet types, and the K-L distance is frequently used to measure the similarity of discrete distributions. One potential drawback of using this distance metric for our application is that, for vectors of counts $\hat{n}_i$ and $\hat{n}_j$, if $\hat{n}_{it} = 0$ for some packet type t but $\hat{n}_{jt} \neq 0$, then the K-L distance from $\hat{n}_j$ to $\hat{n}_i$ is ∞. Clearly, it is not desirable for a single component to cause such a large increase in the distance, especially when $\hat{n}_{jt}$ is also small. To avoid this problem, we apply additive smoothing of the packet counts by initializing all counts for each epoch to one instead of zero.

Figure 1 plots the true detection rates for the k-NN classifier on s-length epochs of `HTTP`, `HTTPS`, `SMTP`-out, and `SSH` traffic for several values of s and k. Recognition rates for most of the protocols tend to increase with both s and k. Larger values of s mean that each epoch includes packets from a greater number of connections, so it is not surprising that, as s increases, the mix of packets observed in a given epoch approaches the mix of packets the protocol tends to produce overall. On the other hand, smaller values of s allow us to analyze shorter traces and should make it more difficult for an adversary to successfully masquerade one protocol as another. We leave a more detailed investigation of the effectiveness of shorter epoch lengths and other countermeasures against active adversaries for future work. For now, we set $s = 10$ sec to achieve an acceptable balance between recognition accuracy and granularity of analysis.

From this simple k-NN classifier with s-length epochs, we can construct a classifier for aggregates that span longer periods of time as follows. Given a sequence of packets corresponding to a traffic aggregate, we begin by preprocessing it into a sequence of vectors of packet counts and normalizing each vector just as we did for each of the aggregates in the training set. We then use



(a) HTTP (b) HTTPS

(c) SMTP(out) (d) SSH

Figure 1: Per-epoch recognition rates for `HTTP`, `HTTPS`, `SMTP`-out and `SSH` with varying values of s and k

the k-NN classifier to determine the protocol label for each vector of counts. Finally, given this list of labels, we simply take its mode—that is, the most frequently-occurring label—as the class label for the aggregate as a whole.

We evaluate this classifier using traffic from a randomly-selected day distinct from that used for training. Table 1 shows the true detection (TD) and false detection (FD) rates for the kNN-based classifier on aggregates assembled from the testing day's traces, using several values of k. For example, when $k = 3$, Table 1 shows that the classifier correctly labels 100% of the `FTP` aggregates and incorrectly labels 1.2% of the other aggregates as `FTP`. This classifier is able to correctly recognize 100% of the aggregates for several of the protocols with many different values of k, leading us to believe that the vectors of packet counts observed for each of these protocols tend to cluster together into perhaps a few large groups. The recognition rates for the more interactive protocols are slightly lower than those for noninteractive protocols, and appear to be more dependent on the parameter k: while `AIM` is recognized better with smaller values of k, the recognition rates for `SSH` and `Telnet` generally tend to improve as k increases.

The results in this section show that, by using the Kullback-Leibler distance to construct a k-Nearest Neighbor classifier for short slices of time, we can then build a classifier for longer traces which performs quite well on aggregate traffic where only a single application protocol is involved. However, we may not always be able to assume that all flows in the aggregate carry the *same* appli-

	1-NN		3-NN		5-NN		7-NN	
protocol	TD	FD	TD	FD	TD	FD	TD	FD
HTTP	100.0	00.0	100.0	00.0	100.0	00.0	100.0	00.0
HTTPS	100.0	00.0	100.0	01.2	100.0	01.2	100.0	03.6
AIM	91.7	00.0	91.7	00.0	91.7	00.0	83.3	00.0
SMTP-in	100.0	00.0	100.0	00.0	100.0	00.0	100.0	00.0
SMTP-out	100.0	03.6	91.7	03.6	91.7	03.6	75.0	03.6
FTP	100.0	03.6	100.0	01.2	100.0	01.2	100.0	02.4
SSH	75.0	00.0	75.0	00.0	75.0	00.0	75.0	00.0
Telnet	83.3	00.0	100.0	00.0	100.0	00.0	100.0	00.0

Table 1: Protocol detection rates for the k-NN classifier ($s = 10$sec)

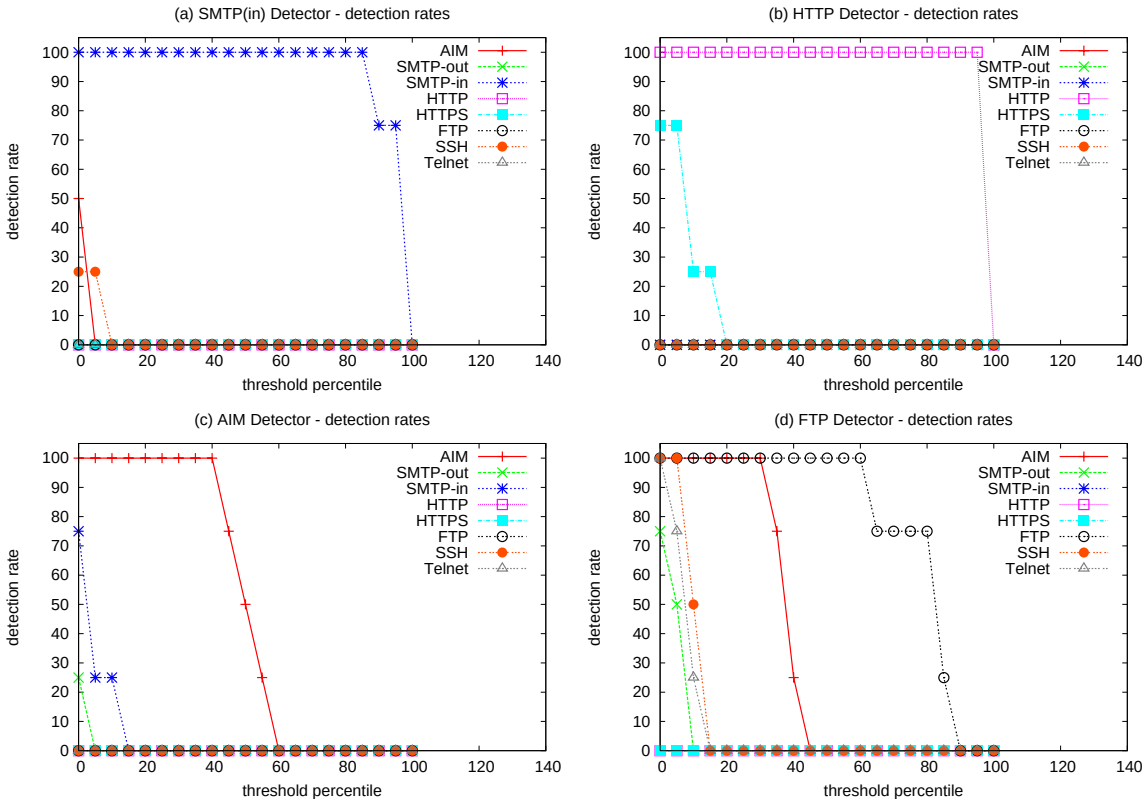


Figure 2: Detection rates for multi-flow protocol detectors ($k = 7, s = 10$sec)

cation protocol. For the specific case where the individual TCP connections can be demultiplexed from the aggregate, we explore techniques in §4 for performing more in-depth analysis to more accurately identify the protocols.

3.2 An Efficient Multi-flow Protocol Detector

Sometimes, a network administrator may be less concerned with classifying all traffic by protocol, and interested instead only in detecting the presence of a few prohibited applications in the network,

such as, for example, the AOL Instant Messenger or similar applications. In this setting, the *k*-NN classifier in §3.1 can be easily modified for use as an efficient protocol detector. If we are concerned only with detecting instances of a given target protocol (or indeed, a set of target protocols), we simply label the vectors in the training set based on whether they contain an instance of the target protocol(s). Then, to run the detector on a new trace of aggregate traffic, we split the trace into several short *s*-length segments of time as before, and we classify each segment using the *k*-NN classifier. We flag the aggregate as an instance of the target protocol if and only if the percentage of the time slices for which the classifier returns `True` is above some threshold. This detector can thus be tuned to be more or less sensitive by adjusting the threshold value.

Figure 2 shows the detection rates for the *k*-Nearest Neighbor-based multi-flow protocol detectors for `AIM`, `HTTP`, `FTP`, and `SMTP`-in, with $k = 7$. In each graph, the x-axis represents the threshold level, and the plots show the probability that the given detector, when set with a particular threshold, flags instances of each protocol in the study.

Overall, the multi-flow protocol detectors seem to perform quite well detecting broad classes of protocol behavior. The detectors for `SMTP`-in (a) and `HTTP` (b) are particularly effective at distinguishing their target protocols from the rest. For example, in Figure 2(b), we see that, for all threshold values above $\approx 30\%$, the `HTTP` detector flags 100% of the simulated `HTTP` tunnels in our test set with no false positives. Even with a threshold level of 10%, it flags nothing but `HTTP` and `HTTPS`. The `FTP` detector's rates (d) show that, when observed in a multi-flow aggregate, the more interactive protocols exhibit very similar on-the-wire behaviors; after `FTP` itself, the `FTP` detector is most likely to flag instances of `AIM`, `SSH`, and `Telnet`. Nevertheless, at a threshold level of 60%, the `FTP` detector achieves a true detection rate over 90% with no false positives.

Interestingly, Figure 2 also gives us information about the *k*NN classifier's ability to correctly label the individual *s*-length epochs in each tunnel. The steep drop in correct detections in each plot occurs approximately when the threshold level exceeds the *k*NN classifier's accuracy for the epochs of the given protocol.

While we have thus far developed techniques which do fairly well in the multi-flow scenario, frequently it may be reasonable to assume that we can in fact demultiplex the individual flows from the aggregate, and finer-grained analysis is often desirable for security applications. For example, consider the scenario where a network administrator uses clustering techniques such as those of Xu et al. (2005) or McGregor et al. (2004) to discover a set of suspicious connections running on non-standard ports. Even if the connections use SSL or TLS to encrypt their packets, the administrator could perform more in-depth analysis to determine the application protocol used in each individual TCP connection. In the next section, we explore techniques for performing such in-depth analysis, again using only a minimal set of features.

4. Machine Learning Techniques for the Analysis of Single Flows

We now relax the earlier assumption that all TCP connections in a given set carry the same application protocol, but retain the assumption that the individual TCP connections can be demultiplexed. Our approaches are equally applicable to the case where there is no aggregate, and instead we simply wish to determine the application protocol(s) in use in a set of TCP connections.

We present an approach based on building statistical models for the sequence of packets produced by each protocol of interest, and then use these models to identify the protocol in use in new TCP connections. To model these streams of packets, and to compare new streams to our mod-

els, we use techniques based on profile hidden Markov models (Krogh et al., 1994; Eddy, 1995). Identifying protocols in this setting is fairly difficult due to the fact that certain application protocols exhibit more than one typical behavior pattern (e.g., `SSH` has `SCP` for bulk data transfer and an interactive, `Telnet`-like, behavior), while other protocols like `SMTP` and `FTP` behave very similarly in almost every regard (Zhang and Paxson, 2000a). These similarities and multi-modal behaviors combine to make accurate protocol recognition challenging even for benign traffic. Nevertheless, here we show that fairly good accuracy can be achieved using vector quantization techniques to learn packet size and timing characteristics in the same discrete-alphabet profile HMM.

For each protocol, denoted p_i, we build a profile model λ_i to capture the typical behavior of a single TCP connection for the given protocol. We train the model λ_i using a set of training connections $p_{i1}, p_{i2}, \ldots, p_{in}$ collected from known instances of the given protocol p_i observed in the wild. Next, given the set of profile models, $\lambda_1, \ldots, \lambda_n$, that correspond to the protocols of interest (say `AIM`, `SMTP`, `FTP`), the goal is to pick the model that best describes the sequences of encrypted packets observed in the different connections.

The overall process for our design and evaluation is illustrated in Figure 3 and entails (*i*) data collection and preprocessing (*ii*) feature selection, modeling and model selection, and finally (*iii*) the classification of test data and evaluation of the classifiers' performance.

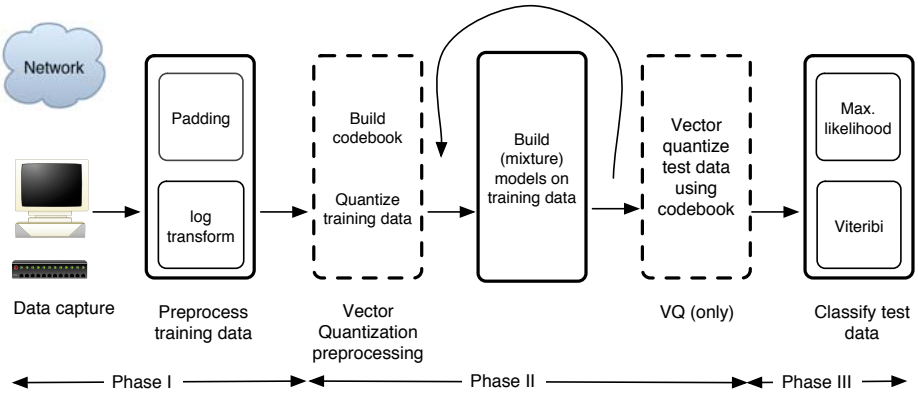


Figure 3: Process overview for construction of our Hidden Markov Model-based classifiers.

In the following sections we describe in greater detail the design of our Hidden Markov models (HMMs) and the classifiers we build using them. We begin with an introduction to profile HMMs and to the Viterbi classifier that we use to recognize protocols. We then present two extensions to the basic profile HMM-based classifier design: first, a vector quantization approach that allows us to combine both packet size and timing in the same model to achieve improved recognition rates for almost all protocols, and second, an efficient method for detecting individual protocols, similar in spirit to those in §3.2.

4.1 Modeling Protocols with HMMs

We now explain the design and use of the profile hidden Markov models we employ to capture the behavior exhibited by single TCP connections. Given a set of connections for training, we begin by constructing an initial model (see Figure 4) such that the length of the chain of states in the

model is equal to the average length (in packets) of the connections in the training set. Using initial parameters that assign uniform probabilities over all packets in each time step, we apply the well-known Baum-Welch algorithm (Baum et al., 1970) to iteratively find new HMM parameters which maximize the likelihood of the model for the sequences of packets in the training connections. Additionally, a heuristic technique called "model surgery"(Schliep et al., 2003) is used to search for the most suitable HMM topology by iteratively modifying the length of the model and retraining.

4.1.1 PROFILE HIDDEN MARKOV MODELS

Our hidden Markov models follow a design similar to those used by Krogh et al. (1994), Eddy (1995), and Schliep et al. (2003) for protein sequence alignment. The profile HMM (Figure 4) is best described as a left-right model built around two long parallel chains of hidden states. Each chain has one state per packet in the TCP connection, and each state emits symbols with a probability distribution specific to its position in the chain. States in these central chains are referred to as `Match` states, because their probability distributions for symbol emissions match the normal structure of packets produced by the protocol.

To allow for variations between the observed sequences of packets in connections of the same protocol, the model has two additional states for each position in the chain. One, called `Insert`, allows for one or more extra packets "inserted" in an otherwise conforming sequence, between two normal parts of the session. The other, called the `Delete` state, allows for the usual packet at a given position to be omitted from the sequence. Transitions from the `Delete` state in each column to `Insert` state in the next column allow for a normal packet at the given position to be removed and replaced with a packet which does not fit the profile.

Just as the output symbols in the HMMs used by Krogh et al. (1994) and others to model proteins represent the different amino acids that make up the protein, the symbols output by states in our HMM correspond directly to the different types of packets that occur in TCP connections. In §4.2 we sort packets into bins based on their size (rounded up to a multiple of the hypothetical cipher's block size) and direction, so symbols in those models are merely bin numbers. In §4.3 we use vector quantization to also incorporate timing information in the model, and the output symbols then become codeword numbers from our vector quantizer.

The main difference between this profile HMM and those used in other domains (Krogh et al., 1994; Eddy, 1995; Schliep et al., 2003) is that the HMMs used to model proteins have only a single chain of `Match` states. In our case, the addition of a second match state per position was intended to allow the model to better represent the correlation between successive packets in TCP connections (Wright et al., 2004). Since TCP uses sliding windows and positive acknowledgments to achieve reliable data transfer, the direction of a packet is often closely correlated (either positively or negatively) to the direction of the previous packet in the connection. Therefore, the `Server Match` state matches only packets observed traveling from the server to the client, and the `Client Match` state matches packets traveling in the opposite direction. For example, a transition from a `Client Match` state to a `Server Match` state indicates that a typical packet (for the given protocol) was observed traveling from the client to the server, followed by a similarly typical packet on its way from the server to the client. In practice, the `Insert` states represent duplicate packets and retransmissions, while the `Delete` states account for packets lost in the network or dropped by the detector. Both types of states may also represent other protocol-specific variations in higher layers of the protocol stack.

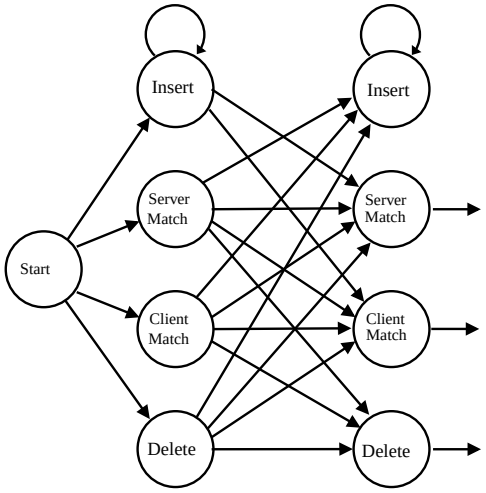


Figure 4: Profile HMM for TCP sequences

4.2 HMM-based Classifiers

Given a HMM trained for each protocol, we then construct a classifier for the task of choosing, in an automated fashion, the best model—and, hence, the best-matching protocol—for new sequences of packets. The task of a model-based classifier is, given an observation sequence O of packets, and a set C of k classes with models $\lambda = \lambda_1, \lambda_2, ..., \lambda_k$, to find $c \in C$ such that $c = \texttt{class}(O)$. We experimented with two HMM-based classifiers for assigning protocol labels to single flows.

Our first such classifier assigns protocol labels to sequences according to the principle of maximum likelihood. Formally, we choose $\texttt{class}(O) = \underset{c}{\texttt{argmax}}\ P(O \mid \lambda_c)$, where $\underset{c}{\texttt{argmax}}$ represents the class c with the highest likelihood of generating the packets in O. Our second classifier is similar to the first, but it makes use of the well-known Viterbi algorithm (Viterbi, 1967) for finding the most likely sequence of states (S) for a given output sequence O and HMM λ. The Viterbi algorithm can be used to find both the most likely state sequence (i.e., the "Viterbi path"), and its associated probability $P_{viterbi}(O, \lambda) = \underset{S}{\texttt{max}}\ P(O,\ S \mid \lambda)$. Given an output sequence O, our Viterbi classifier finds Viterbi paths for the sequence in each model λ_i and chooses the class c whose model produces the best Viterbi path. We can express this decision policy concisely as $\texttt{class}(O) = \underset{c}{\texttt{argmax}}\ P_{viterbi}(O, \lambda_c)$.

In practical terms, the Viterbi classifier finds each model's best explanation for how the packets in the sequence were generated (whether by normal application behavior, TCP retransmissions, etc.), represented by the Viterbi path, and the likelihood of each model's explanation (i.e., the Viterbi path probability). It then picks the model that provides the best explanation for the observed packets.

Empirical Evaluation To demonstrate the applicability of our techniques to real traffic, we randomly select 9 days from over a period of one month and extract traces over a 10 hour period between 10 a.m. and 8 p.m. on each day. For a given experiment, we select one day for use as a

	micro-level		equivalence class	
protocol	TD	FD	TD	FD
AIM	80.80	3.41	80.80	3.41
SMTP-out	73.20	3.07	80.10	1.82
SMTP-in	77.20	4.39	87.80	3.97
HTTP	90.30	2.10	96.70	1.47
HTTPS	88.50	3.24	94.40	2.72
FTP	57.70	2.01	57.70	2.01
SSH	69.10	2.93	71.00	2.88
Telnet	82.90	3.77	86.10	4.08

Table 2: Protocol detection rates for the Viterbi classifier, using packet sizes only

training set. From this day's traces, we randomly select approximately 400 connections [2] of each protocol and use these to build our profile HMMs. Then, for each of the remaining 8 days, we randomly select approximately 400 connections for each protocol and use the model-based classifier to assign class labels to each of them. We repeat this experiment a total of nine times using each day once as the training set, and the recognition rates we report are averages over the 9 experiments.

By selecting testing and training sets that include the same number of connections for each protocol, we purposefully exclude from our classifiers any knowledge about the traffic mix in the network, in order to show that our techniques are applicable even when we know nothing *a priori* about the particular network under consideration. As a result, we believe the detection rates presented here could be improved for a given network by including the relative frequencies of the protocols (i.e., as Bayesian priors). Additionally, while greater recognition accuracy could be achieved by rebuilding new models more frequently (e.g., weekly), we do not do so, in order to present a more rigorous evaluation. On a 2.4GHz Intel Xeon processor, our unoptimized classifier can assign class labels to one experiment's test set of 3200 connections in roughly 5 minutes.

Table 2 presents our results for the Viterbi classifier when considering only the size and direction of the packets. Again, recall in this case that we make decisions at the granularity of single flows and potentially have much less information at our disposal than in §3.1. With the exception of the connections for FTP and SSH, the Viterbi classifier correctly identifies the protocol more than 73% of the time. Moreover, the average false detection rates for all protocols (i.e., the probability that an unrelated connection is incorrectly classified as an instance of the given protocol) are below 5%. The full confusion matrix is given in Table 4 in Appendix A, and shows that many of the misclassifications can be attributed to confusions with protocols in the same equivalence class, for example, HTTP versus HTTPS. As such, we also report the true detection (TD) and false detection (FD) rates when we group protocols into the following equivalence classes: $\{[\texttt{AIM}], [\texttt{HTTP}, \texttt{HTTPS}], [\texttt{SMTP}-\textit{in}, \texttt{SMTP}-\textit{out}], [\texttt{FTP}], [\texttt{SSH}, \texttt{Telnet}]\}$ where the latter class represents the grouping of the interactive protocols.

We find the Viterbi classifier to be slightly more accurate than the Maximum Likelihood classifier in almost every case,[3] but the protocol whose recognition rates are most improved with the Viterbi method is SSH. Unlike the other protocols in this study, SSH has at least two very different modes of operation—interactive shell (SSH) and bulk data transfer (SCP)—so we are not surprised

2. We choose 400 because it is the largest size for which we can select the same number of instances of each protocol on every day in the data set.

3. Therefore, due to space constraints we do not provide recognition rates for that classifier.

	micro-level		equivalence class	
protocol	TD	FD	TD	FD
AIM	83.90	2.53	83.90	2.53
SMTP-out	74.40	2.24	79.70	1.60
SMTP-in	79.80	3.34	85.90	3.02
HTTP	78.00	1.09	92.90	0.62
HTTPS	87.20	3.74	91.10	1.88
FTP	58.20	1.81	58.20	1.81
SSH	76.30	8.37	77.80	7.90
Telnet	79.50	2.44	90.70	2.60

Table 3: Protocol detection rates for the Viterbi classifier with 140-codeword VQ

to find that for many SSH sessions, some sequences of states in the HMM for SSH are much more likely than other state sequences in the same model.

4.3 Vector Quantization for HMMs on Multiple Features

While the results thus far show surprising success for building models of network protocols using only a single variable, one would suspect that recognition rates could be further improved by including both size and timing information in the same model. To evaluate this hypothesis, we employ a vector quantization technique to transform our two-dimensional packet data into symbols from a discrete alphabet so that we can then use the same type of models and techniques as used for dealing with timing or size individually. Our vector quantization approach proceeds as follows: given training data and viewing each packet as a two-dimensional tuple of $\langle$inter-arrival time, size$\rangle$, we first apply a log transform to the times to reduce their dynamic range (Feldmann, 2000; Paxson, 1994). Next, to assign the sizes and times equal weight, we scale the $\langle$log (time), size$\rangle$ vectors into the -1,1 square.

The nature of our models requires that we treat packets differently based on the direction they travel. We therefore split the packets into two sets: those sent from the client to the server, and those sent from server to client. We then run the *k*-means clustering algorithm separately on each set to find a representative set of vectors, or codewords, for the packets in the given set. For a quantizer with a codebook of N codewords, for each of the two sets of packets, we begin by randomly selecting $k = N/2$ vectors as cluster centroids. Then, in each iteration, for each $\langle$time, size$\rangle$ vector, we find its nearest centroid and assign the vector to the corresponding cluster. We recalculate each centroid at the end of each iteration as the vector mean of all the vectors currently assigned to the cluster, and stop iterating when the fraction of vectors which move from one cluster to another drops below some threshold (currently 1%).

After clustering both sets of packet vectors, we take the list of centroid vectors as the codebook for our quantizer. To quantize the vector representation of a packet, we simply find the codeword nearest the vector, and encode the packet as the given codeword's index in the codebook. After performing vector quantization of the packets in the training set of connections, we can then build discrete HMMs as before, using codeword numbers as the HMM's output alphabet. In doing so, we add important information to our models at only a modest cost in complexity and computational efficiency. Before classifying test connections, we use the codebook built on the training set to quantize their packets in the same manner.

Table 3 depicts the results for the Viterbi classifier, using a codebook of 140 codewords.[4] By including both size and timing information in the same profile model, we are able to recognize interactive traffic more accurately—SSH's recognition rate is now over 75 percent in the detailed test. Both protocols in the "interactive" equivalence class show improvement in their coarse-grained recognition rates, and while micro-level recognition of the WWW protocols decreases due to increased confusions of HTTP as HTTPS and vice versa, the classifier's ability to identify the equivalence class of non-interactive sequences remains unchanged.

However, like our previous HMM classifier in §4.2, the vector quantized version still does not recognize FTP as accurately as the other protocols; its 58% recognition rate is the lowest of any of our current classifiers. We believe this poor performance is caused by the presence of strong multimodal behaviors in the FTP traces; unlike the other protocols in our study, FTP has three common behavior modes which are very distinct and clearly identifiable in visualizations. (See, for example, http://www.cs.jhu.edu/~cwright/traffic-viz.)

4.4 A Protocol Detector for Single Flows

In this section, we evaluate the suitability of our profile HMMs for a slightly different task: identifying the TCP connections that belong to a given protocol of interest. As in §3.2, such a detector could be used, for example, by a network administrator to detect policy violations by a user running a prohibited application (such as instant messenger) or remotely accessing a rogue SMTP server over an encrypted connection.

One approach to this problem would be to simply use the classifiers from Section 4.2, and have the system flag a detection when a sequence is classified as belonging to the protocol of interest. However, such an approach is computationally intensive because of the large number of models required. To classify a sequence of packets, the classifier in Section 4.2 must compute the sequence's Viterbi path probability on each protocol's model before making its decision. So, for example, in our earlier experiments, for each test sequence we explored Viterbi paths across 8 models. While we believe this cost to be warranted when we are interested in determining which protocol generated what connections in the network, at other times one may simply be interested in determining whether or not connections belong to a target protocol. In this case, we show how to build a detector with much lower runtime costs by using only two or three models.

To construct an efficient single-protocol detector we adopt the techniques presented by Eddy et al. (1995) for searching protein sequence databases. As in the previous sections, we build a profile HMM λ_P for the target protocol P. We also build a "noise" model λ_R to represent the overall distribution of sequences in the network. For the noise model, we use a simple HMM with only a single state which thus only captures the unigram packet frequencies observed in the network. Intuitively, this model is intended to represent the packets we expect to see in background traffic, so we estimate its parameters using connections from all protocols in the study.[5]

4. Derived empirically by exploring various codebook sizes up to 180 codewords. No significant difference in recognition ability was observed beyond 140.

5. We note that one might instead train λ_R on only the set of non-target protocols for each detector. However, doing so relies on a closed-world assumption and risks over-estimating the detector's real accuracy because λ_R is then a model for all the things that the target protocol specifically *is not*. In practice, there are simply too many protocols in use on modern networks for such an approach to be feasible.

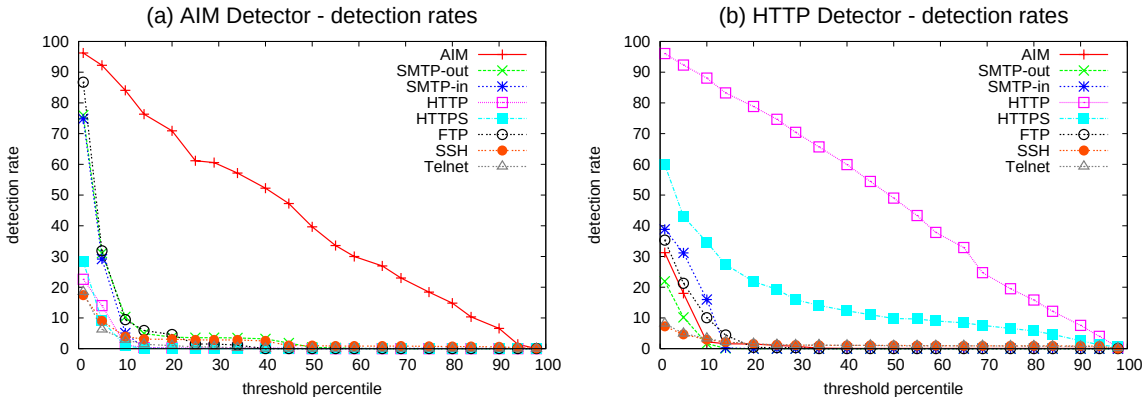


Figure 5: Detection rates for threshold-based protocol detectors for AIM and HTTP

To determine whether an observed sequence O belongs to the protocol of interest P, we calculate its log odds score as

$$score(O) = \log \frac{P_{viterbi}(O \mid \lambda_P)}{P_{viterbi}(O \mid \lambda_R)}.$$

Our general approach is now as follows: We use a holdout set of connections for the target protocol, distinct from the set used to estimate the model's parameters, to determine a threshold score T_P for the protocol. This allows us to tune the detector's false positive and true positive rates. To set the threshold, we calculate log odds scores for each connection in the holdout set, and set the threshold in accordance with our desired detection rate. For example, if we wish to detect 90% of all instances of the given protocol (at the risk of incorrectly flagging many connections that belong to other protocols), we set the threshold at the log odds score of the held-out connection which scored in the 10^{th} percentile, so that 90% of all connections in the holdout set score above the threshold. In our simplest (and most efficient) protocol detector, a test connection whose log odds score falls above the threshold is immediately flagged as an instance of the given protocol.

The goal, of course, is to build detectors which simultaneously achieve high detection rates for their target protocols and near-zero detection rates for the other, non-target protocols. To empirically evaluate the extent to which our protocol detectors are able to do so, we run each detector on a number of instances of each protocol. For this round of experiments, we select three days at random from the GMU traces. In each experiment, we designate one of the three randomly-selected days for use as a training set, then randomly select one of the remaining two days for use as a holdout set and use the third day as our test set. We then extract 400 connections from the training set for each of the protocols we want to detect, and use these connections to build one profile HMM for each protocol and one unigram HMM for the noise model. Similarly, we randomly select 400 connections from the holdout set for each protocol, and use these to determine thresholds for a range of detection rates between 1% and 99% for each protocol detector.

Finally, we randomly select 400 connections of each protocol from the test set, and run each protocol detector on all 3200 test connections. Figure 5 presents the detection rates for the AIM and HTTP detectors. Such detectors are able to analyze one experiment's test connections in roughly 15 seconds—around 20 times faster than the full classifier from Section 4.2.

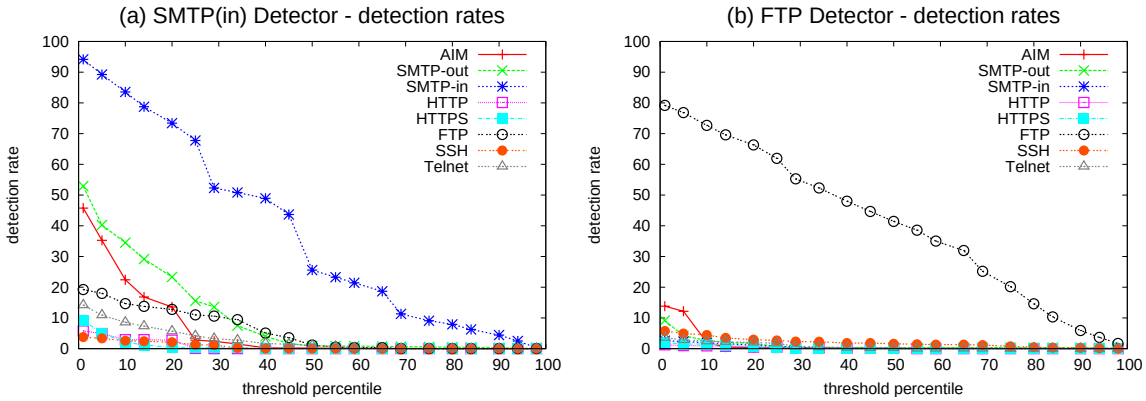


Figure 6: Detection rates for the improved protocol detectors for SMTP-in FTP

In Fig. 5(*a*) and 5(*b*), we see that both the AIM detector and the HTTP detector achieve over 80% true detections while flagging less than 10% of other traffic. Moreover, the HTTP detector, for example, can be tuned to achieve a near-zero false detection rate yet still correctly identify over 70% of HTTP connections. Similarly, detectors for HTTPS, SMTP and FTP built on this basic design are also able to distinguish their respective protocols from most of the other protocols in our test set, with reasonable accuracy.

However, the FTP and SMTP-in detectors are both prone to incorrectly claim each other's connections as their own. For example, a SMTP-in detector built in this manner would incorrectly flag 60% of all FTP sessions while detecting only 75% of incoming SMTP. This is not surprising, since FTP and SMTP share a similar "numeric code and status message" format and generate sequences of packets that look very similar, even when examining packet payloads. Indeed, the two are so similar that Zhang and Paxson (2000a) went so far as to use the same rule set to detect both protocols. Nevertheless, we are able to improve our initial false positive rates for these two protocols using a technique based on iteratively refining of the set of protocols that we suspect a connection might belong to.

To build an improved protocol detector, we construct profile HMMs not only for the target protocol, but also for any other similarly-behaving protocols with which it is frequently confused. As above, we construct a unigram HMM for the noise model. In the iterative refinement technique, we first use the simple threshold-based detector described above as a first-pass filter, to determine if a connection is likely to contain the target protocol. If a connection passes this first filter, we use the Viterbi classifier (Sec. 4.2) with the models for the frequently-confused protocols to identify the other (non-target) protocol most likely to have generated the sequence of packets in the connection. Only if the model for the target protocol produces a higher Viterbi path probability than this protocol's model, do we flag the connection as an instance of the target protocol. While these improved detectors operate ≈ 3 times slower than the simple detectors described previously, their performance is still over 6 times faster than that of the full classifier.

Fig. 6(*a*) and 6(*b*) show the detection rates for the iterative refinement detectors for SMTP-in and FTP, respectively, when the detectors know that incoming SMTP and FTP are frequently confused with each other. While the FTP detector suffers a decrease in true positives with the iterative refinement technique, it also achieves false positive rates of less than 15% for all protocols, at all

thresholds. We note that an SMTP detector built in this manner is much less prone to falsely flagging FTP sessions; its worst false positive rate for FTP is now below 20%. Again, we stress that if better accuracy rates are required, one can fall back to the design in Section 4.2 at the cost of greater computational overhead.

5. Tracking the Number of Live Connections in Encrypted Tunnels

In §3, we showed that it is often possible to determine the application protocol used in aggregate traffic without demultiplexing or reassembling the TCP connections in the aggregate. Then, in §4, we demonstrated much-improved recognition rates by taking advantage of the better semantics in the case where we can demultiplex the flows from the aggregate and analyze them individually.

We now turn our attention to the case where we cannot demultiplex the flows or determine which packets in the aggregate belong to which flows, as is the case when aggregate traffic is encrypted at the network layer using IPsec Encapsulating Security Payload (Kent and Atkinson, 1998) or SSH tunneling. Specifically, we develop a model-based technique which enables us to accurately track the number of connections in a network-layer tunnel which carries traffic for only a single application protocol. As an example of this scenario, consider a proxy server which listens for clients' requests on one edge network and forwards them through an encrypted tunnel across the Internet to a set of servers on another edge network. Despite our inability to demultiplex the flows inside such a tunnel, the technique developed in §3 still enables us to correctly identify the application protocol much of the time. We now go on to show how we can, given the application protocol, derive an estimate for the number of connections in the tunnel at each point in time. This technique might be used, for example, by a network administrator to distinguish between a legitimate tunnel used by a single employee for access to her mail while on the road, versus a backdoor used by spammers to inject large quantities of unsolicited junk mail.

Our approach is founded on a few basic assumptions about the behavior of the tunneled TCP connections and their associated packets. These assumptions, while not entirely correct for real traffic, nevertheless allow us to employ simple and usable models which, as we demonstrate later, produce reasonable results for a variety of protocols.

Assumption 1 *The process N_t describing the number of connections in the tunnel is a Martingale (Doob, 1953; Williams, 1991), meaning that, on average, it tends to stay about the same over time.*

Assumption 2 *The process N_t describing the number of connections in the tunnel is a Gaussian process. That is, the number of connections N_t in each time slice t follows a Gaussian distribution.*

Assumption 3 *For each packet type m, each connection in the tunnel generates packets of type m according to a homogeneous Poisson process with constant rate γ_m, which is determined by the application protocol in use in the connection.*

Implications It follows from Assumption 1 and Assumption 2 that, in each timeslice, the number of connections in the tunnel will have a Gaussian distribution with mean equal to the number of connections in the tunnel during the previous timeslice. From Assumption 3, it follows that during an interval of length s, the number of type-m packet arrivals will follow a Poisson distribution with parameter equal to $\gamma_m s$. Accordingly, the set of packet rates $\{\gamma_m\}$ provides a sufficiently descriptive

model for the given application protocol (in this scenario). We use these observations in the following section to build models that enable us to extract information about the number of tunneled TCP connections from the observed sequence of packet arrivals.

5.1 A Model for Multi-Flow Tunnels

To track the number of connections in a multi-flow tunnel, we build a statistical model which relates the stochastic process describing the number of live connections to the stochastic process of packet arrivals. For such a doubly-stochastic process, it is natural to again use hidden Markov models. Here, the hidden state transition process describes the changing number of connections N_t in the tunnel, and the symbol output process describes the arrival of packets on the link. States in the HMM therefore correspond to connection counts, and the event that the HMM is in state i at time t corresponds to the event that we see packets from i distinct connections during time slice t. When we consider M different types of packets, the HMM's outputs are M-tuples of packet counts.

To build such a model, we derive the state transition and symbol emission probabilities directly from two parameters which, in turn, we must estimate based on some training data. These parameters are: first, the standard deviation σ of the number of live connections in each epoch, and second, the set of base packet rates $\{\gamma_m : \text{packet types } m\}$.

Under Assumption 2, the average number of live connections in a time slice follows a Normal distribution with mean equal to the average number of live connections in the previous interval and standard deviation σ. Therefore, the probability of a state transition from state i to state j is simply the probability that a Normal random variable with mean i and standard deviation σ falls between $j-0.5$ and $j+0.5$, and thus, rounded to the nearest integer, is j. Re-expressed in terms of the standard Normal, we therefore have

$$a_{ij} = \Phi(\frac{(j-0.5)-i}{\sigma}) - \Phi(\frac{(j+0.5)-i}{\sigma}).$$

Due to Assumption 3, that each live connection generates packets independently according to a Poisson process, we expect the total packet arrival rate to increase linearly with the number of live connections. Then, when there are j connections in the tunnel, the number of type-m packet arrivals in an interval of length s will follow a Poisson distribution with parameter equal to $j\gamma_m s$. Therefore, the probability of the joint event that we observe n_{tm} packets of each type m during an interval of length s, when there are j connections in the tunnel, is given by

$$b_j(\hat{n}_t) = \prod_{m=1}^{M} e^{-j\gamma_m s} \frac{(j\gamma_m s)^{n_{tm}}}{n_{tm}!}.$$

Parameter Estimation To estimate the two fundamental parameters of our model, $\{\gamma_m\}$ and σ, we observe the characteristics of real network traffic from a training set of traces. We begin by preprocessing traces from our training set as described in §3.1, dividing the training trace(s) into many smaller intervals of uniform length s. For each s-length interval t, we measure (1) the number of connections N_t in the tunnel during the interval, and (2) $\hat{n}_t = \langle n_{t1}, n_{t2}, \ldots, n_{tM} \rangle$, the number of packets of each type which arrive during the given interval. For each packet type m, we fit a line to the set of points $\{(N_t, n_{tm})\}$ using least squares approximation, and we derive our estimate for γ_m as the slope of this line. That is, γ_m gives us the rate at which the number of packets observed increases with the number of connections in the tunnel.

We estimate σ as the sample conditional standard deviation of the number of connections in the tunnel N_t during an interval t, given the number N_{t-1} in the tunnel in the preceding interval. For the HMM's remaining parameter, the initial state distribution π, we simply use a uniform distribution. In doing so, we refrain from making any assumptions about the traffic intensity on the test network.

5.2 Tracking the Number of Connections

To derive the state sequence that best explains an observed sequence of packet counts (and, hence, the average number of live connections during each interval), we use the Forward and Backward dynamic programming variables from the Baum-Welch algorithm (Baum et al., 1970) to calculate the probability that the HMM visits each state in each time step. The forward variable, $\alpha_t(i)$, gives the probability that, in step t, the model has produced the outputs $\hat{n}_1, \ldots, \hat{n}_t$ and is in state i. We can define α_t recursively:

$$\alpha_1(i) = \pi_i b_i(\hat{n}_1),$$

$$\alpha_t(i) = \sum_{j=1}^{N} \alpha_{t-1}(j)\ a_{ji}\ b_i(\hat{n}_t).$$

Similarly, the backward variable, $\beta_t(i)$, gives the probability that the model, starting from state i in step t, produces the remaining outputs $\hat{n}_t, \ldots, \hat{n}_T$. It is also defined recursively:

$$\beta_T(i) = 1,$$

$$\beta_t(i) = \sum_{j=1}^{N} b_i(\hat{n}_t)\ a_{ij}\ \beta_{t+1}(j).$$

With this, we can calculate the probability that the model is in state i at time step t as

$$P(\text{state } i \text{ at time } t) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

and we can calculate the most-likely individual state at time t as

$$\phi_t = \underset{i}{\texttt{argmax}}\ P(\text{state } i \text{ at time } t)$$

which reduces to

$$\phi_t = \underset{i}{\texttt{argmax}}\ \alpha_t(i)\beta_t(i).$$

And thus ϕ_t is our estimate of the number of connections N_t in the tunnel at time step t.

5.3 Empirical Results

To evaluate the effectiveness of our approach in practice, we randomly select one day in the GMU data set for use as a training set and one day as a test set. We use a collection of traces from several hours on the training day to learn the model's parameters and construct a HMM for each protocol. We then simulate tunnels for each of the protocols in each 10-minute trace from the designated testing day, by assembling aggregates as we did in §3. Instead of using all traces in the data set as before, in this section we simulate traffic for an encrypted proxy server by selecting only those

Figure 7: Actual and estimated number of connections in simulated tunnels for AIM, HTTP, HTTPS, and SSH in the 12:00 trace on the testing day

connections which go to the most common IP addresses in each 10-minute trace. For each protocol, we split its tunnel into several short time slices and derive the corresponding sequence $\hat{n}_1, \hat{n}_2, \ldots, \hat{n}_T$ of packet counts. We then use the given protocol's model to derive a sequence of estimates for the number of connections in the tunnel during each slice.

Often, our model is able to closely track the number of live connections in the tunnel, although it can under- or over-estimate at times. Figure 7 shows the actual number of connections N_t and the model's estimates ϕ_t for AIM, HTTP, HTTPS, and SSH in the 10-minute GMU trace for 12:00 noon, the busiest period on the testing day. The models for AIM and HTTPS are able to track the true number of connections in their tunnels especially well: on average, their predictions differ from true number of connections by only 22% and 19%, respectively. Between time ticks 45 and 90, the HTTPS model tracks large swings in the population size, and the AIM model follows the general trend quite closely between 30 and 100 time ticks.

The model for HTTP, on the other hand, has some difficulty with this particular trace; while such errors do not occur in all traces, we include this example to demonstrate some of the weaknesses of our current assumptions. We suspect the large spike at around 20 ticks may be due to already-open

persistent connections suddenly requesting pages and thus generating a burst of packets. Between 40 and 65 ticks, the HTTP tunnel produces a sequence of packets where the relative frequencies of the different packet types are out of proportion to those on the training day. The model can find no state with a non-negligible probability of generating such a traffic mix, and so sets its estimate for the number of HTTP connections in the tunnel to zero. Despite some intermittent errors as exemplified here, because our technique operates in near-real time, an administrator could observe an encrypted tunnel for many such windows of time and then still derive a good estimate for the traffic intensity in the tunnel. In the short term, we hope to improve these results by using Viterbi training to improve the model's initial parameters.

6. Related Work

While traffic classification has recently been the subject of much research, all but one of the approaches we are aware of require significantly more information about the flows, or only group flows into broad categories such as "bulk data transfer," "p2p," or "interactive." Zhang and Paxson (2000a) present one of the earliest studies of techniques for network protocol recognition without using port numbers, based on matching patterns in the packet payloads. Dreger et al. (2006) and Moore and Papagiannaki (2005) present similar approaches to that of Zhang and Paxson, but apply more sophisticated analyses which require payload-level inspection. More closely related to our work is that of Early et al. (2003), where a decision tree classifier that used *n*-grams of packets was proposed for distinguishing among flows from HTTP, SMTP, FTP, SSH and Telnet servers based on average packet size, average inter-arrival time, and TCP flags. Moore and Zuev (2005) use Bayesian analysis techniques on similar data from packet headers to classify flows as belonging to one of several broad categories. Bernaille et al. (2006) build a classifier based on *k*-means clustering of the sizes of the first five packets in each connection to identify application protocols "on the fly." Because the focus of that work is on speed rather than security, they do not consider all packets in the connection as we do.

A direct comparison of our empirical results with those of the above approaches is not feasible at this time because there is currently no (realistic) shared data set on which to evaluate the various techniques side-by-side. In fact, in the preliminary stages of this work (Wright et al., 2004), we attempted to do just that by evaluating our preliminary classifier on network traces from the MIT Lincoln Labs Intrusion Detection Evaluation (Lippmann et al., 2000). However, the MITLL data set is now several years old, and it has been criticized as unrepresentative of real traffic (McHugh, 2000). The validity of these criticisms is evident in our own experiences: our naïve classifier, which was able to recognize a handful of protocols in the MITLL data with reasonable accuracy, did not perform well on real wide-area traffic (Faxon et al., 2004). Its evaluation on real data highlighted many of the problems addressed herein.

Recently, Karagiannis et al. (2005) proposed an interesting approach for performing traffic classification "in the dark" which, like ours, does not use port numbers or the contents of packet payloads. However, their technique does rely on information about the behavior of the hosts in the network. In particular, the approach makes use of the *social* and *functional* roles of hosts, that is, their interactions with other hosts and whether they act as a provider or consumer of a service, respectively. In this way, Karagiannis et al. (2005) focuses more on learning host behavior and inferring the applications in flows based on the hosts' interactions. Unfortunately, while this technique

may be capable of identifying the type of an application, it might not be able to identify distinct applications (Karagiannis et al., 2005), and it does not classify individual flows or connections.

McGregor et al. (2004) present a technique for *clustering* network flows without using packet payloads. Whereas we view flows as sequences of packet sizes and times, they represent each flow as a finite-dimensional vector of flow attributes and use the standard k-means algorithm to cluster them. Similar to the idea present here, Coull et al. (2003) recently used sequence alignment techniques to detect masquerades in Unix shell histories. We believe our results in this paper validate their application of sequence alignment methods for the purpose of masquerade detection. However, unlike that of Coull et al. (2003), our profiling technique does not require pairwise alignments of all sequences, and is therefore better suited for studying network protocols (where the training data requirements may be fairly large).

More distantly related work is that on stepping stone detection. By correlating the timing of on/off periods in inbound and outbound interactive connections, Zhang and Paxson (2000b) demonstrate how to detect "stepping stone" connections whereby an adversary tries to conceal the true source of an attack by hopping from one host to another. Wang et al. (2002) and Yoda and Etoh (2000) subsequently used methods similar to sequence alignment to detect stepping stones by identifying TCP connections with similar packet streams—the general idea being to find good alignments of the streams by identifying locations where the two subsequences of inter-arrival times are most similar.

Packet timing and/or size information have also been used in several application-specific information leakage attacks on various kinds of encrypted traffic. For example, Sun et al. (2002) identify web pages within `SSL`–encrypted connections by examining the sizes of the HTML objects returned in the `HTTP` response. Similarly, Felten and Schneider (2000) demonstrate that web servers can use the inter-arrival time of `HTTP` requests for objects on a web page to reveal the presence of items in the browser's cache. Song et al. (2001) show that the interarrival times of packets in `SSH` (version 1) connections can be used to infer information about the user's keystrokes and thereby reduce the search space for cracking login passwords. A recent paper by Kohno et al. (2005) presents a method for identifying individual physical devices over the network, using clock skew information observable in the device's TCP headers.

7. Conclusions and Future Work

In this paper, we demonstrate how application behavior remains detectable in encrypted network traffic. First, we show how application protocols can be identified in aggregate traffic without demultiplexing and reassembling the individual TCP connections. We also show that, when it is possible to demultiplex the flows, more in-depth analysis of the packets in each flow can lead to even more robust and accurate classification even when a mix of several protocols are included. Finally, and perhaps most surprisingly, we show that encrypted tunnels which carry only a single application protocol leak sufficient information about the flows in the tunnel to allow us to accurately track their number.

In future work, we will explore ways to harden our current techniques against an active adversary. Such work will necessarily include research into useful metrics for capturing the power of an active adversary. Our current investigations explore the feasibility of using the *divergence* of the adversary's model from the data's true distribution. Other metrics might include bounds on the maximum number of bytes or packets the adversary can add to the original stream, or the maximum

delay or jitter she can induce. We also intend to extend our techniques to more general types of encrypted tunnels, with the ultimate goal of being able to track the number of connections of each protocol inside a full IPsec VPN (Kent and Atkinson, 1998).

Acknowledgments

We would like to graciously thank Dr. Don Faxon and the Statistics Group at George Mason University, for providing access to their packet traces. This work would not have been possible without their support and assistance. This work is supported by NSF grant CNS-0546350.

Appendix A.

Table 4 depicts shows the full confusion matrix for the Viterbi classifier when analyzing TCP connections as sequences of packet sizes. These results averaged for 9 days chosen at random during the same month, and reflect average classification rates over all 72 pairs of testing and training days.

Classification Probability									
Protocol	AIM	SMTP-out	SMTP-in	HTTP	HTTPS	FTP	SSH	Telnet	none
AIM	80.8	2.9	1.4	1.6	3.1	0.9	5.4	3.2	0.7
SMTP-out	7.1	73.2	6.9	1.2	1.9	2.3	1.9	5.2	0.3
SMTP-in	2.5	10.6	77.2	0.1	0.2	4.6	0.8	3.9	0.1
HTTP	0.7	0.3	0.1	90.3	6.4	0.3	1.3	0.4	0.1
HTTPS	0.9	0.8	0.1	5.9	88.5	0.6	1.9	0.8	0.5
FTP	7.1	4.1	11.1	0.9	2.1	57.7	6.0	11.0	0.0
SSH	3.4	1.8	9.3	1.5	6.8	2.8	69.1	1.9	3.2
Telnet	2.2	1.0	1.8	3.5	2.2	2.6	3.2	82.9	0.4

Table 4: Confusion matrix for Viterbi classifier with profile topology

References

Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41(1):164–171, February 1970.

Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, April 2006.

Bram Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, June 2003.

Scott Coull, Joel Branch, Boleslaw Szymanski, and Eric Breimer. Intrusion detection: A bioinformatics approach. In *Proceedings of the* 19^{th} *Annual Computer Security Applications Conference*, pages 24–33, December 2003.

Joseph L Doob. *Stochastic Processes*. Wiley, 1953.

Holger Dreger, Anja Feldmann, Michael Mai, Vern Paxson, and Robin Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In *Proceedings of the* 15^{th} *Usenix Security Symposium*, pages 257–272, August 2006.

James Early, Carla Brodley, and Catherine Rosenberg. Behavioral authentication of server flows. In *Proceedings of the* 19^{th} *Annual Computer Security Applications Conference*, pages 46–55, December 2003.

Sean Eddy. Multiple alignment using hidden Markov models. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 114–120, July 1995.

Sean Eddy, Graeme Mitchison, and Richard Durbin. Maximum discrimination hidden Markov models of sequence consensus. *Journal of Computational Biology*, 2:9–23, 1995.

Don Faxon, R Duane King, John T Rigsby, Steve Bernard, and Edward J Wegman. Data cleansing and preparation at the gates: A data-streaming perspective. In *2004 Proceedings of the American Statistical Association*, August 2004.

Federal Information Processing Standards. Advanced Encryption Standard (AES) – FIPS 197, November 2001.

Anja Feldmann. *Characteristics of TCP connection arrivals*. Park and Willinger (Ed). Wiley-Interscience, 2000.

Edward W Felten and Michael A Schneider. Timing attacks on web privacy. In *Proceedings of the 7th ACM conference on computer and communications security*, pages 25–32, November 2000.

Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. BLINC: Multilevel traffic classification in the dark. In *ACM SIGCOMM*, to appear, August 2005.

Stephen Kent and Ran Atkinson. RFC 2406: IP encapsulating security payload (ESP), November 1998.

Tadayoshi Kohno, Andre Broido, and kc claffy. Remote physical device fingerprinting. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 211–225, May 2005.

Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. Hidden Markov Models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, February 1994.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951.

Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 130–143, May 2001.

Richard P Lippmann, David J Fried, Isaac Graf, Joshua W Haines, Kristopher R Kendall, David McClung, Dan Weber, Seth E Webster, Dan Wyschogrod, Robert K Cunningham, and Marc A Zissmann. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, January 2000.

Anthony McGregor, Mark Hall, Perry Lorier, and James Brunskill. Flow clustering using machine learning techniques. In *The 5th Anuual Passive and Active Measurement Workshop (PAM 2004)*, April 2004.

John McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294, November 2000.

Andrew Moore and Konstantina Papagiannaki. Towards the accurate identification of network applications. In *The 6th Anuual Passive and Active Measurement Workshop (PAM 2005)*, March 2005.

Andrew W Moore and Denis Zuev. Internet traffic classification using Bayesian analysis techniques. In *ACM SIGMETRICS*, June 2005.

Vern Paxson. Emprically-derived analytic models of wide-area tcp connections. *IEEE/ACM Transactions on Networking*, 2(4):316–336, August 1994.

Eric Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2000.

Alexander Schliep, Alexander Schönhuth, and Christine Steinhoff. Using hidden Markov models to analyze gene expression time course data. *Bioinformatics*, 19(supplement 1):i255–i263, July 2003.

Dawn Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and SSH timing attacks. In *Proceedings of the* 10th *USENIX Security Symposium*, August 2001.

Qixiang Sun, Daniel R Simon, Yi-Min Wang, Will Russell, Venkata N Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 19–30, May 2002.

Andrew J Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–267, 1967.

Xinyuan Wang, Douglas S Reeves, and S Felix Wu. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In 7th *European Symposium on Research in Computer Security (ESORICS)*, pages 244–263, October 2002.

David Williams. *Probability with Martingales*. Cambridge University Press, 1991.

Charles Wright, Fabian Monrose, and Gerald M Masson. HMM profiles for network traffic classification (extended abstract). In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pages 9–15, October 2004.

Charles Wright, Fabian Monrose, and Gerald M Masson. Using visual motifs to classify encrypted traffic. In *Proceedings of the* 3rd *International Workshop on Visualization for Computer Security*, November 2006. To appear.

Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharya. Profiling internet backbone traffic: Behavior models and applications. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 169–180, August 2005.

Tatu Ylonen. SSH - secure login connections over the internet. In *Proceedings of the* 6^{th} *USENIX Security Symposium*, pages 37–42, July 1996.

Kunikazu Yoda and Hiroaki Etoh. Finding a connection chain for tracing intruders. In 6^{th} *European Symposium on Research in Computer Security (ESORICS)*, pages 191–205, October 2000.

Yin Zhang and Vern Paxson. Detecting back doors. In *Proceedings of the* 9^{th} *USENIX Security Symposium*, pages 157–170, August 2000a.

Yin Zhang and Vern Paxson. Detecting stepping stones. In *Proceedings of the* 9^{th} *USENIX Security Symposium*, pages 171–184, August 2000b.